Lecture 9

CME 295: Transformers & Large Language Models

Afshine Amidi & Shervine Amidi

Stanford University

ICME

Transformers & Large Language Models

Recap

Beyond Transformer-based LLMs

Diffusion LLMs

Closing thoughts

Rewinding the quarter...

Lecture 1

Transformers

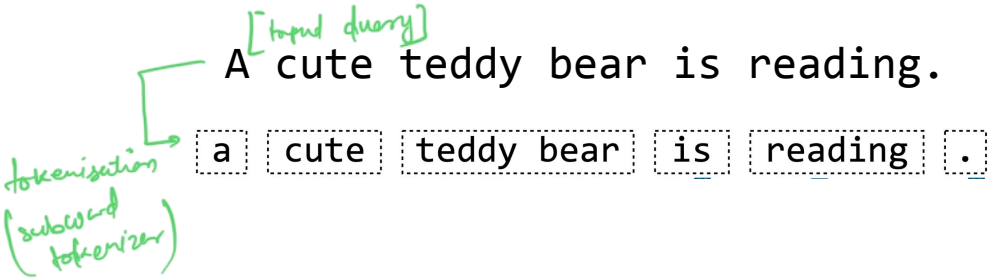
Rewinding the quarter...

Lecture 1

```
A cute teddy bear is reading.
```

Rewinding the quarter...

Lecture 1

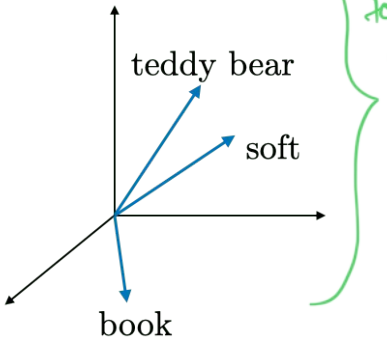


Rewinding the quarter...

Lecture 1

A cute teddy bear is reading.

a | cute | teddy bear | is | reading | .



token representation/ embedding representation (dividing input text into atomic units i.e. tokens)

[word2vec - algorithm for token representation]

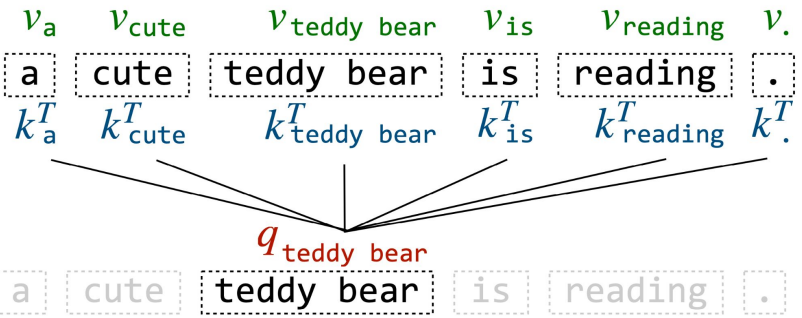
used for learning for representing (predicting center word/ predicting word)

Rewinding the quarter...

Lecture 1

A cute teddy bear is reading.

a | cute | teddy bear | is | reading | .



all tokens attends all and formulated into the form of query, key, value.

once token representation being specified; token representation can be used for learning the prediction and relationship btw represented for next token prediction which initially started with RNN but with having certain limits, concept of "self attention" being introduced.

Rewinding the quarter...

Lecture 1

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

metric formulation of calculating attention value of attending one token to other

"Attention Is All You Need", Vaswani et al., 2017.

Rewinding the quarter...

Lecture 1

A cute teddy bear is reading.

a | cute | teddy bear | is | reading | .

v_a v_{cute} $v_{teddy\ bear}$ v_{is} $v_{reading}$ $v_.$

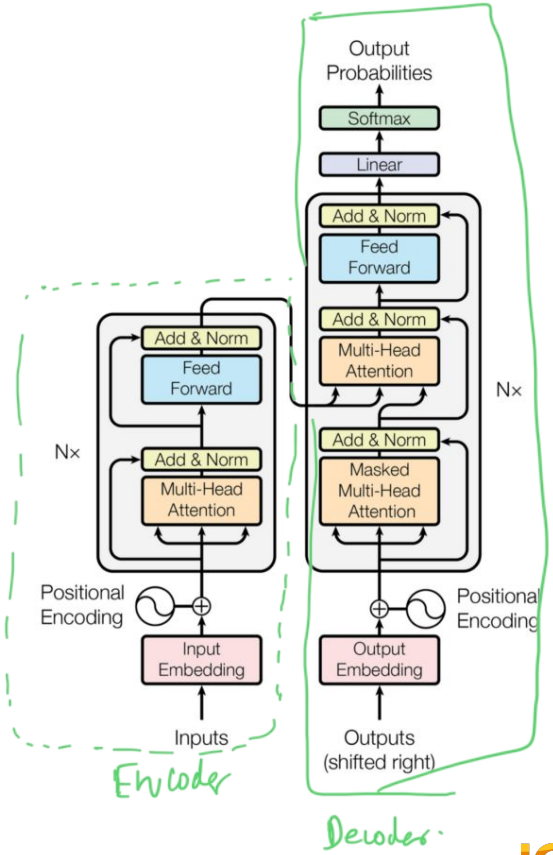
a | cute | teddy bear | is | reading | .

k^T_a k^T_{cute} $k^T_{teddy\ bear}$ k^T_{is} $k^T_{reading}$ $k^T_.$

$q_{teddy\ bear}$

a | cute | teddy bear | is | reading | .

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



"Attention Is All You Need", Vaswani et al., 2017.

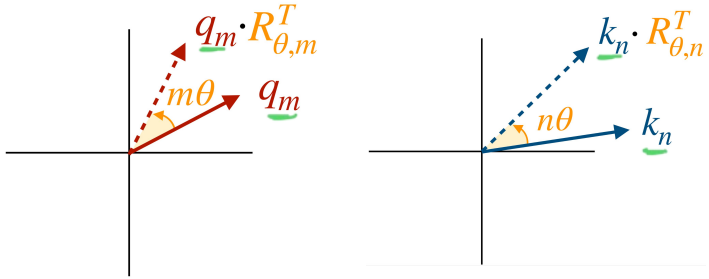
Rewinding the quarter...

Lecture 1 | Lecture 2

Transformer-based models & tricks

Rewinding the quarter...

Lecture 1 | Lecture 2



$$R_{\theta,m} = \begin{pmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{pmatrix}$$

introduced position embedding for Transfer paper.

"rotary position embedding" gives relative position btw token

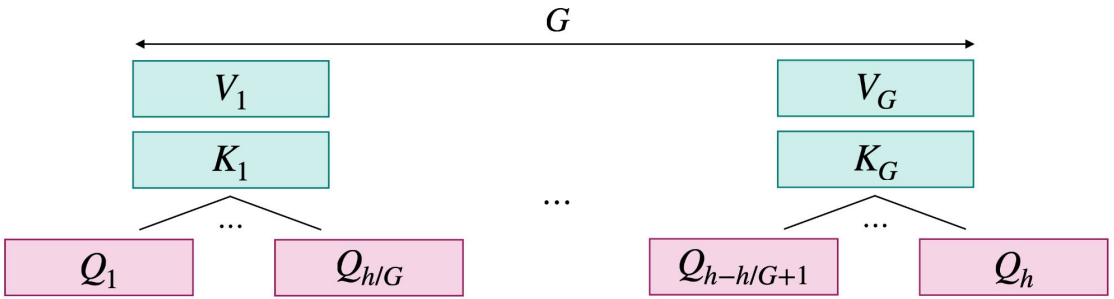
it says how far tokens are in self-attention mechanism.

"RoFormer: Enhanced Transformer with Rotary Position Embeddings", Su et al., 2021.
Figure from "Super Study Guide: Transformers & Large Language Models", Amidi et al., 2024.

Rewinding the quarter...

Lecture 1 | Lecture 2

$$R_{\theta,m} = \begin{pmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{pmatrix}$$



in the case of multi-head attention mechanism, groupings of projection metric"
ie Key and value will be shared with query in grouped"

"GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints", Ainslie et al., 2023.
Figure from "Super Study Guide: Transformers & Large Language Models", Amidi et al., 2024.

Rewinding the quarter...

Lecture 1 | Lecture 2

$$R_{\theta,m} = \begin{pmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{pmatrix}$$

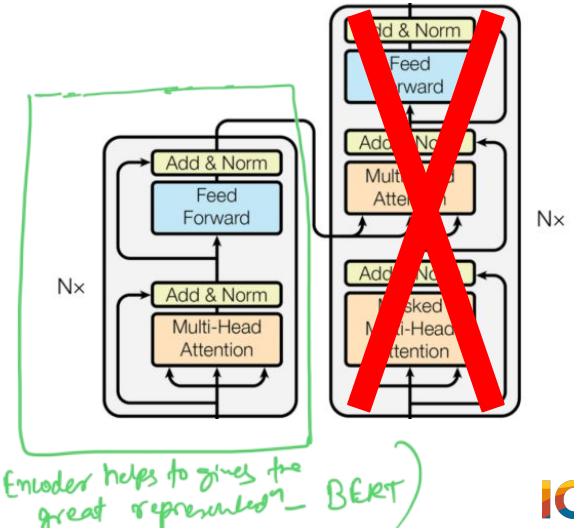


Figure adapted from "Attention Is All You Need", Vaswani et al., 2017.

Rewinding the quarter...

only decoder based architecture - GPT

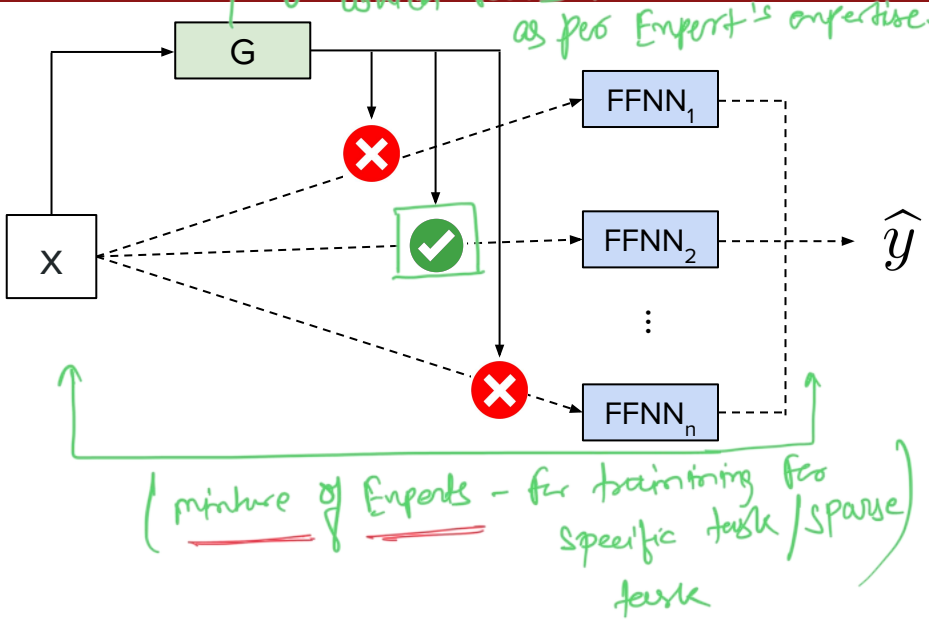
Lecture 1 | Lecture 2 | Lecture 3

Large Language Models

Rewinding the quarter...

Lecture 1 | Lecture 2 | Lecture 3

decoder only transformer model.



gating mechanism which routes to correct FFN as per Expert's expertise.

(mixture of Experts - for training for specific tasks / sparse tasks)

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

Rewinding the quarter...

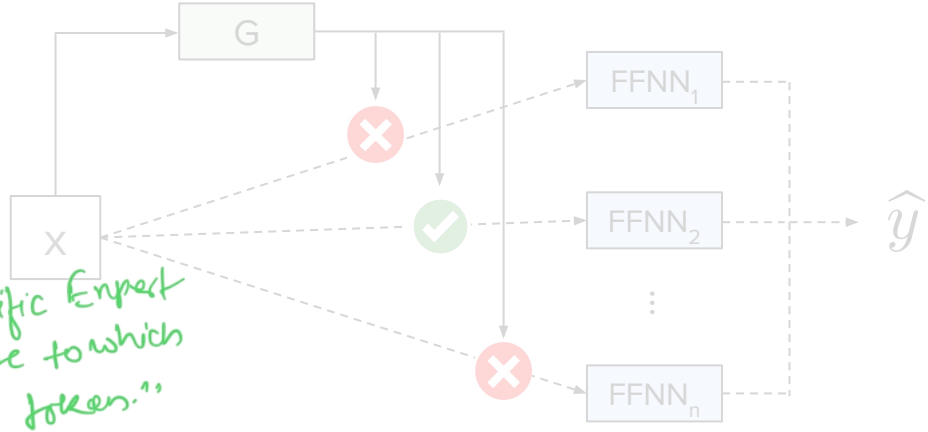
Lecture 1 | Lecture 2 | **Lecture 3**

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)
```

"Which specific Expert will route to which specific tokens."



"Mixtral of Experts", Jiang et al., 2024.

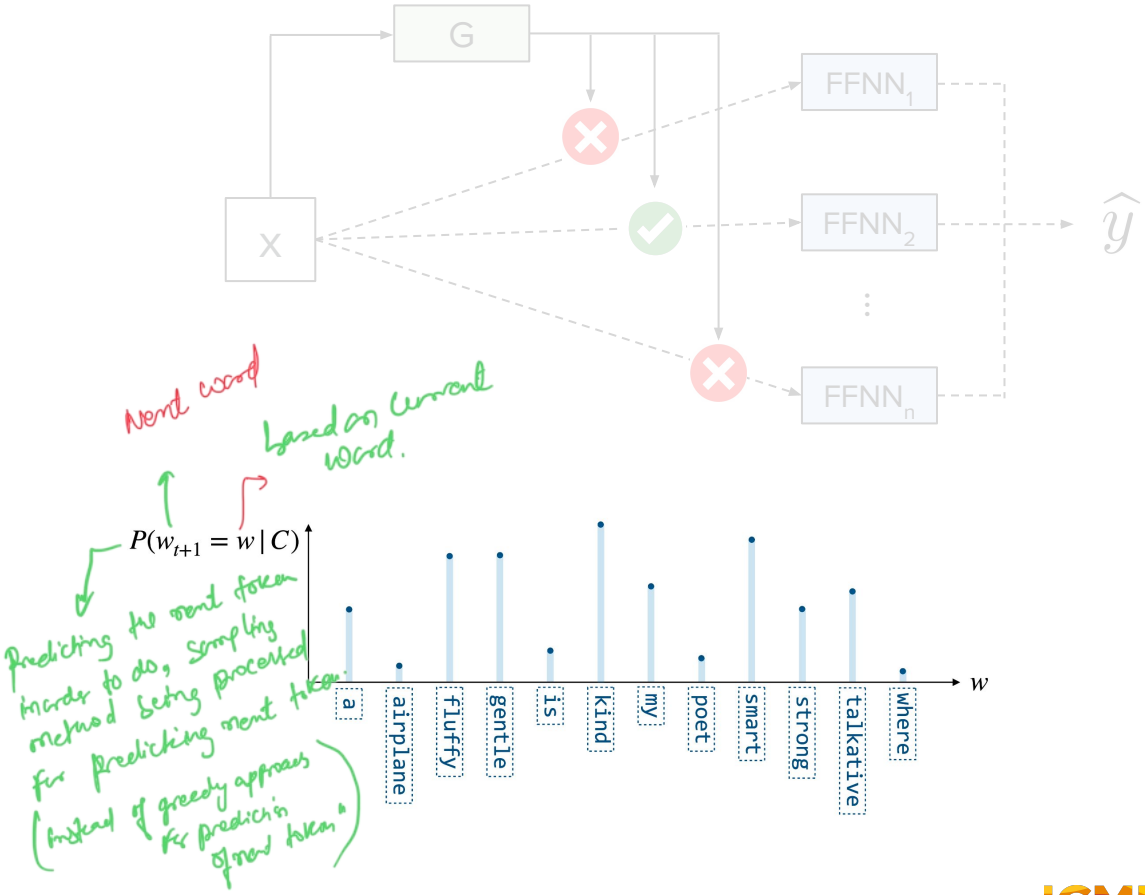
Rewinding the quarter...

Lecture 1 | Lecture 2 | Lecture 3

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)
```



Rewinding the quarter...

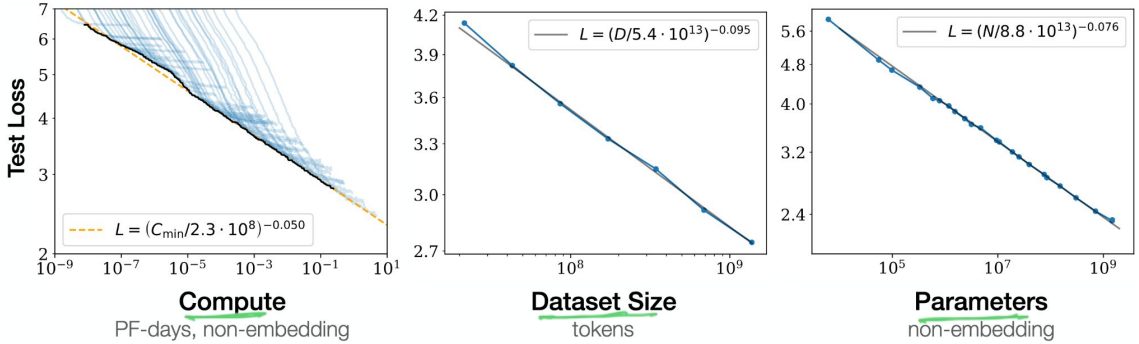
Lecture 1 | Lecture 2 | Lecture 3 | Lecture 4

LLM training

Rewinding the quarter...

(depends on # of parameters, data size, model size)

Lecture 1 | Lecture 2 | Lecture 3 | Lecture 4 → "The bigger your model is better the performance"



"Scaling Laws for Neural Language Models", Kaplan et al., 2020.

Rewinding the quarter...

Lecture 1 | Lecture 2 | Lecture 3 | Lecture 4

Parameters	FLOPs	FLOPs (in *Gopher* unit)	Tokens
400 Million	1.92e+19	1/29,968	8.0 Billion
1 Billion	1.21e+20	1/4,761	20.2 Billion
10 Billion	1.23e+22	1/46	205.1 Billion
67 Billion	5.76e+23	1	1.5 Trillion
175 Billion	3.85e+24	6.7	3.7 Trillion
280 Billion	9.90e+24	17.2	5.9 Trillion
520 Billion	3.43e+25	59.5	11.0 Trillion
1 Trillion	1.27e+26	221.3	21.2 Trillion
10 Trillion	1.30e+28	22515.9	216.2 Trillion

Studied relationship b/w size of model and no of tokens/dataset being used for training.

So, optimal model will be considered as in LLM

# of tokens used for training ≈ 20 * # of parameter in model

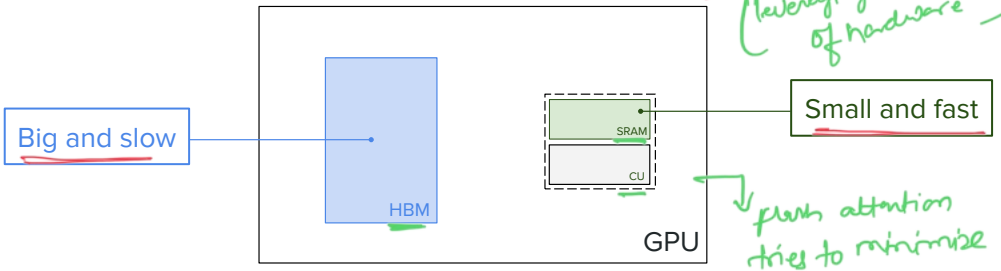
" rule of thumb "

"Training Compute-Optimal Large Language Models", Hoffmann et al., 2022.

Rewinding the quarter...

Lecture 1 | Lecture 2 | Lecture 3 | Lecture 4

Parameters	FLOPs	FLOPs (in *Gopher* unit)	Tokens
400 Million	1.92e+19	1/29,968	8.0 Billion
1 Billion	1.21e+20	1/4,761	20.2 Billion
10 Billion	1.23e+22	1/46	205.1 Billion
67 Billion	5.76e+23	1	1.5 Trillion
175 Billion	3.85e+24	6.7	3.7 Trillion
280 Billion	9.90e+24	17.2	5.9 Trillion
520 Billion	3.43e+25	59.5	11.0 Trillion
1 Trillion	1.27e+26	221.3	21.2 Trillion
10 Trillion	1.30e+28	22515.9	216.2 Trillion



Flash attention for making computation efficient. (leveraging the structure of hardware

flash attention tries to minimize # of read & write operations to memory

"*FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness*", Dao et al., 2022.

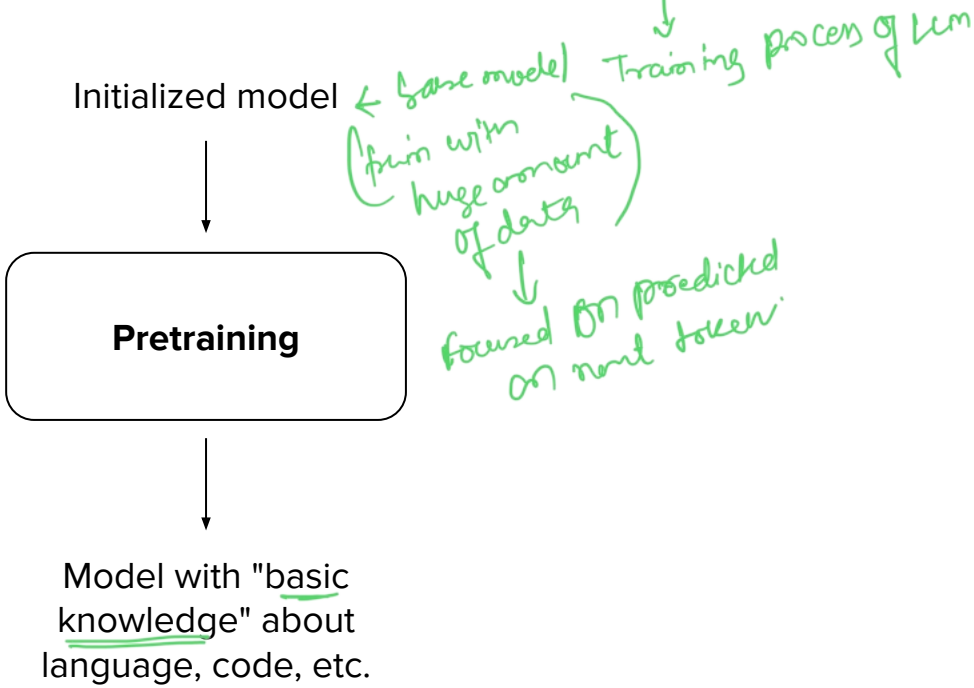
Rewinding the quarter...

Lecture 1 | Lecture 2 | Lecture 3 | Lecture 4

Initialized model

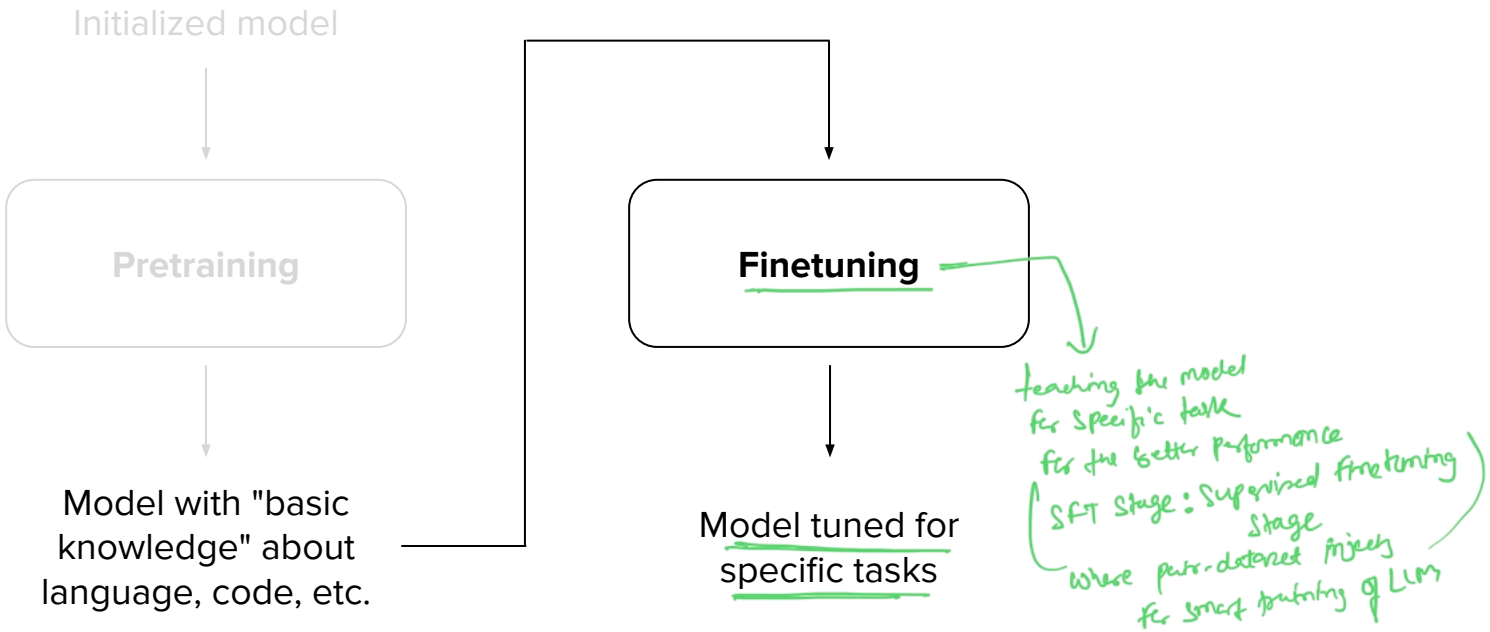
Rewinding the quarter...

Lecture 1 | Lecture 2 | Lecture 3 | Lecture 4



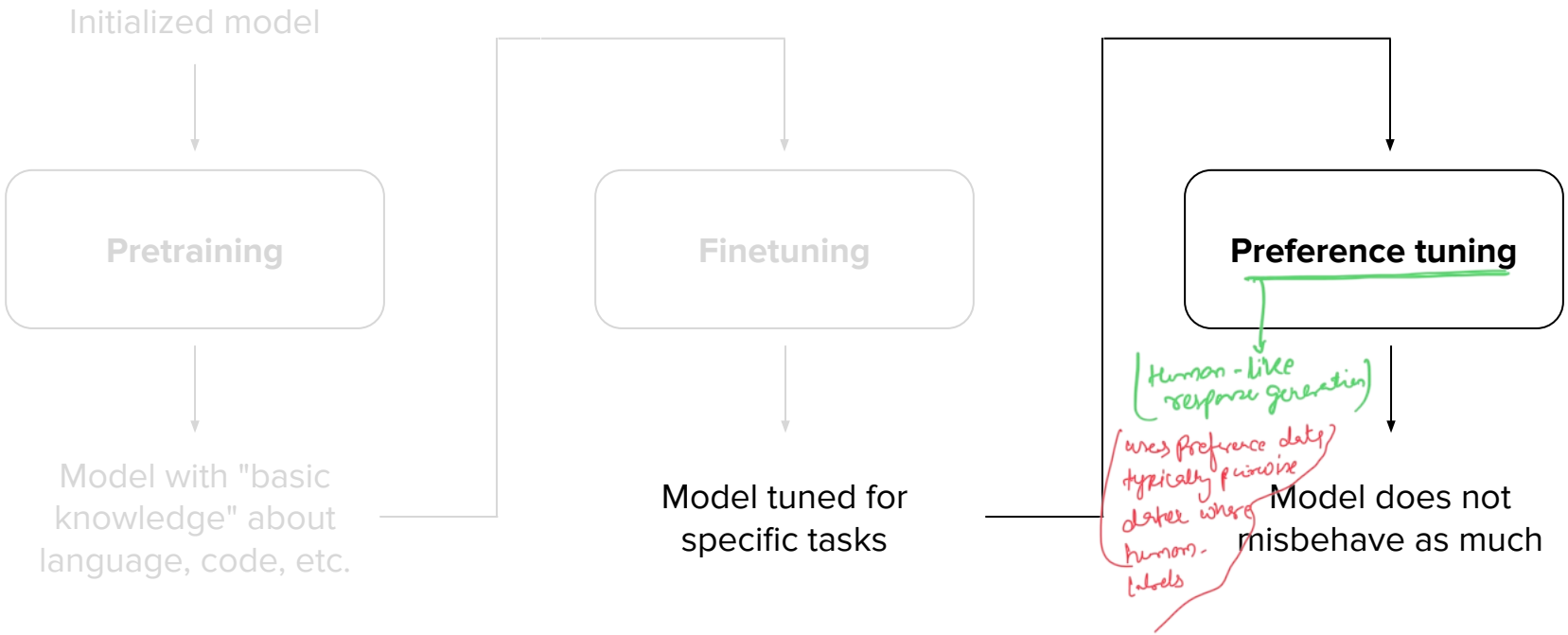
Rewinding the quarter...

Lecture 1 | Lecture 2 | Lecture 3 | Lecture 4



Rewinding the quarter...

Lecture 1 | Lecture 2 | Lecture 3 | Lecture 4



Rewinding the quarter...

Lecture 1 | Lecture 2 | Lecture 3 | Lecture 4 | Lecture 5

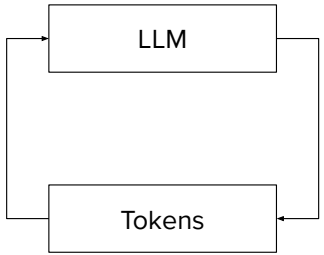
RL-based techniques introduced for "preference-tuning"

LLM tuning

Rewinding the quarter...

Lecture 1 | Lecture 2 | Lecture 3 | Lecture 4 | **Lecture 5**

Human preference r_t



Input so far

Next token

Probability of next token $\pi_\theta(a_t|s_t)$

Rewinding the quarter...

Lecture 1 | Lecture 2 | Lecture 3 | Lecture 4 | **Lecture 5**

Human preference r_t

Input so far

Next token

Probability of next token $\pi_\theta(a_t|s_t)$

Bradley-Terry formulation.

$$p(y_i > y_j) = \frac{e^{r_i}}{e^{r_i} + e^{r_j}} = \sigma(r_i - r_j)$$

(RM) modelling reward (given two outputs, it say which output is good/bad)

"*The rank analysis of incomplete block designs: I. the method of paired comparisons*", Bradley et al., 1952.

Rewinding the quarter...

Lecture 1 | Lecture 2 | Lecture 3 | Lecture 4 | **Lecture 5**

Human preference r_t

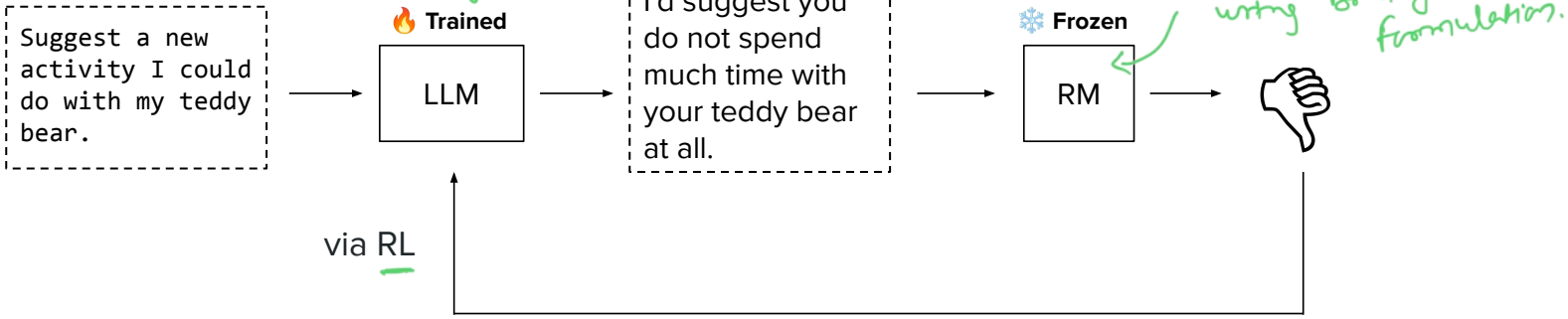
Input so far

Next token

Probability of next token $\pi_\theta(a_t|s_t)$

$$p(y_i > y_j) = \frac{e^{r_i}}{e^{r_i} + e^{r_j}} = \sigma(r_i - r_j)$$

importing LLM for Human-preference output.



"*Training language models to follow instructions with human feedback*", Ouyang et al., 2022.

Rewinding the quarter...

Lecture 1 | Lecture 2 | Lecture 3 | Lecture 4 | **Lecture 5**

Input so far

Next token

$$p(y_i > y_j) = \frac{e^{r_i}}{e^{r_i} + e^{r_j}} = \sigma(r_i - r_j)$$

$$\mathcal{L}(\theta) = \text{Maximize rewards} + \text{Don't deviate too much from old/base model}$$

Loss function

Validate w/ ground truth data.

"*Proximal Policy Optimization Algorithms*", Schulman et al., 2017.

Rewinding the quarter...

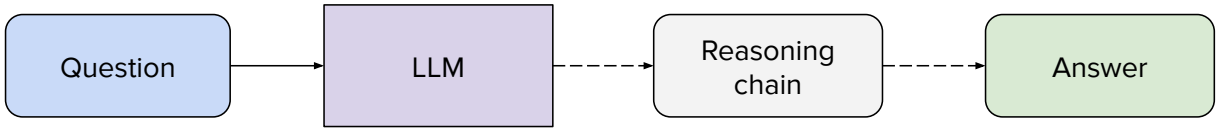
Lecture 1 | Lecture 2 | Lecture 3 | Lecture 4 | Lecture 5 | **Lecture 6**

LLM reasoning

using "chain of thought" - RL being used for LLM training.

Rewinding the quarter...

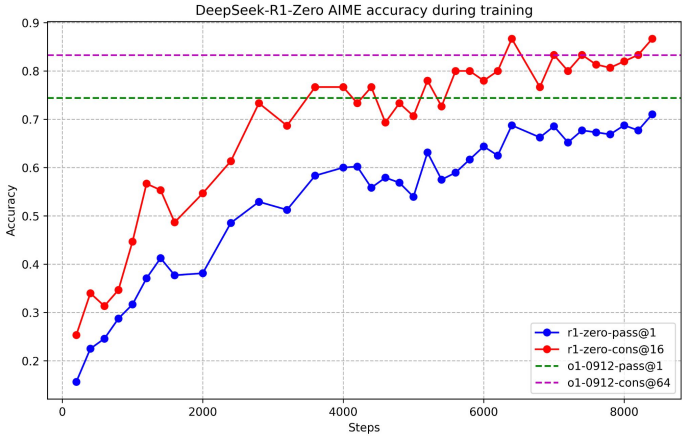
Lecture 1 | Lecture 2 | Lecture 3 | Lecture 4 | Lecture 5 | **Lecture 6**



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

Rewinding the quarter...

Lecture 1 | Lecture 2 | Lecture 3 | Lecture 4 | Lecture 5 | **Lecture 6**



"*DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*", *DeepSeek-AI, 2025.*

Rewinding the quarter...

Lecture 1 | Lecture 2 | Lecture 3 | Lecture 4 | Lecture 5 | **Lecture 6**

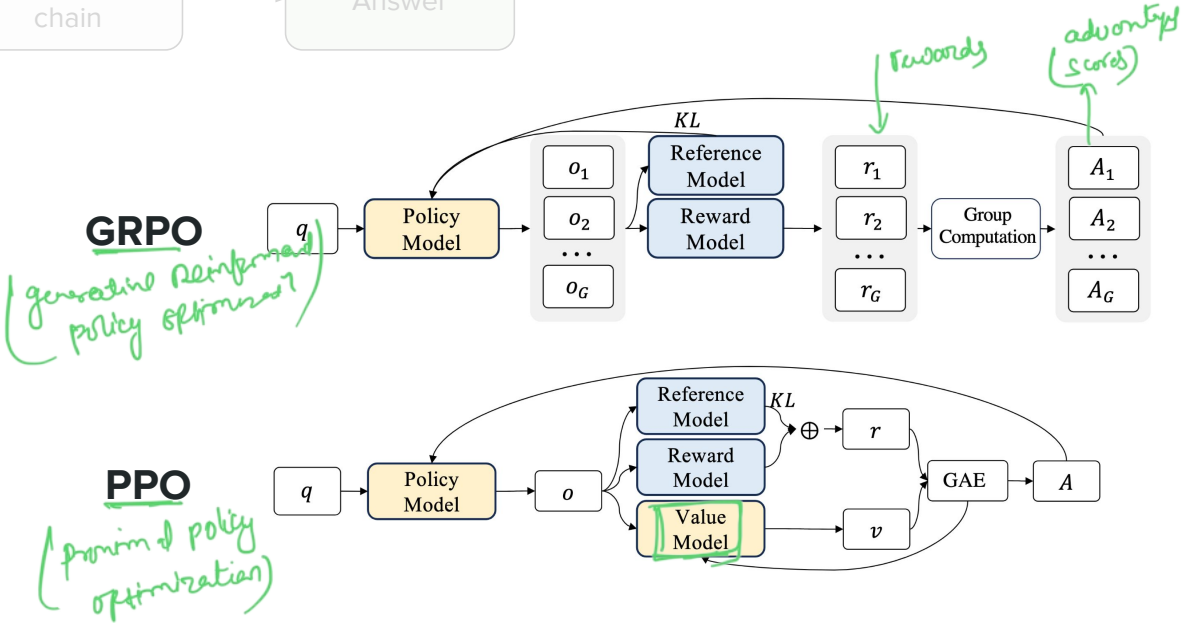


Figure from "DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models", Shao et al., 2024.

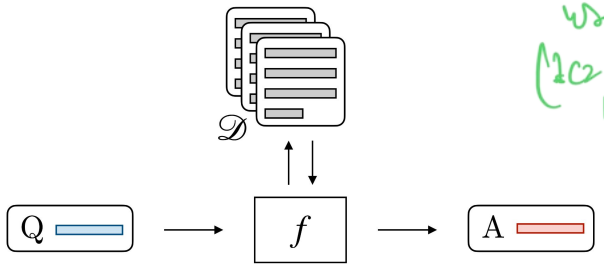
Rewinding the quarter...

Lecture 1 | Lecture 2 | Lecture 3 | Lecture 4 | Lecture 5 | Lecture 6 | Lecture 7

Agentic LLMs

Rewinding the quarter...

Lecture 1 | Lecture 2 | Lecture 3 | Lecture 4 | Lecture 5 | Lecture 6 | Lecture 7

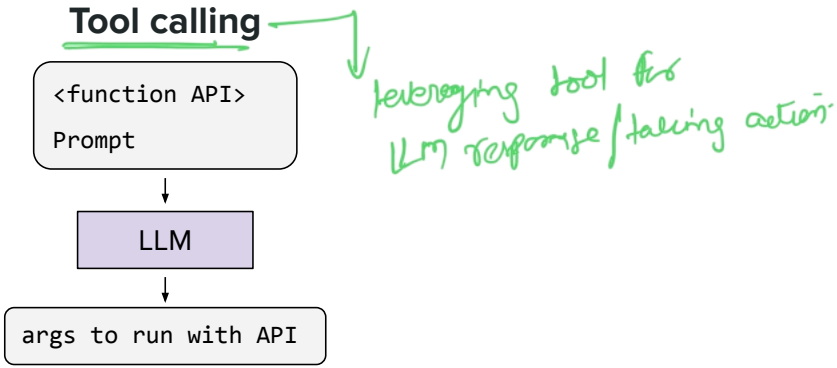


Retrieval Augmented Generation
↓
used for fetching relevant document
(bc model is trained upto cutoff date
but to make LLM smarter, need to
be updated with latest information
which RAG does
↓
using cosine similarity for retrieving
relevant information
(ranking)

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

Rewinding the quarter...

Lecture 1 | Lecture 2 | Lecture 3 | Lecture 4 | Lecture 5 | Lecture 6 | **Lecture 7**



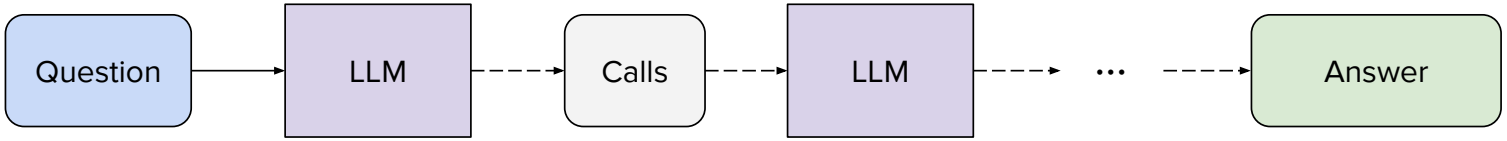
Rewinding the quarter...

Lecture 1 | Lecture 2 | Lecture 3 | Lecture 4 | Lecture 5 | Lecture 6 | **Lecture 7**

RAG

Tool calling

Agent



"*ReAct: Synergizing Reasoning and Acting in Language Models*", *Yao et al., 2022.*

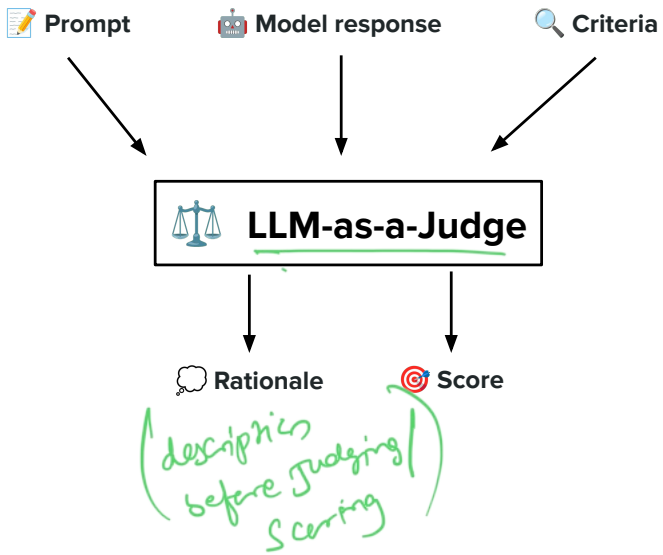
Rewinding the quarter...

Lecture 1 | Lecture 2 | Lecture 3 | Lecture 4 | Lecture 5 | Lecture 6 | Lecture 7 | Lecture 8

LLM evaluation

Rewinding the quarter...

Lecture 1	Lecture 2	Lecture 3	Lecture 4	Lecture 5	Lecture 6	Lecture 7	Lecture 8



"*Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena*", *Zheng et al., 2023.*

Rewinding the quarter...

Lecture 1 | Lecture 2 | Lecture 3 | Lecture 4 | Lecture 5 | Lecture 6 | Lecture 7 | **Lecture 8**

Knowledge	Reasoning	Coding	Safety
• Accurately state facts about the world • More "breadth" than "depth" • Reflects pretraining quality	• Solve multi-step problems • Includes both "maths" and "common sense" subfields • Internal reasoning	• Generate syntactically correct code • Tests for programming proficiency • Proxy for tool use abilities	• Prevent harmful, toxic, inappropriate behavior • Surfaces vulnerabilities before deployment • Alignment with custom preferences
MMLU	AIME PIQA	SWE-bench	HarmBench

Transformers & Large Language Models

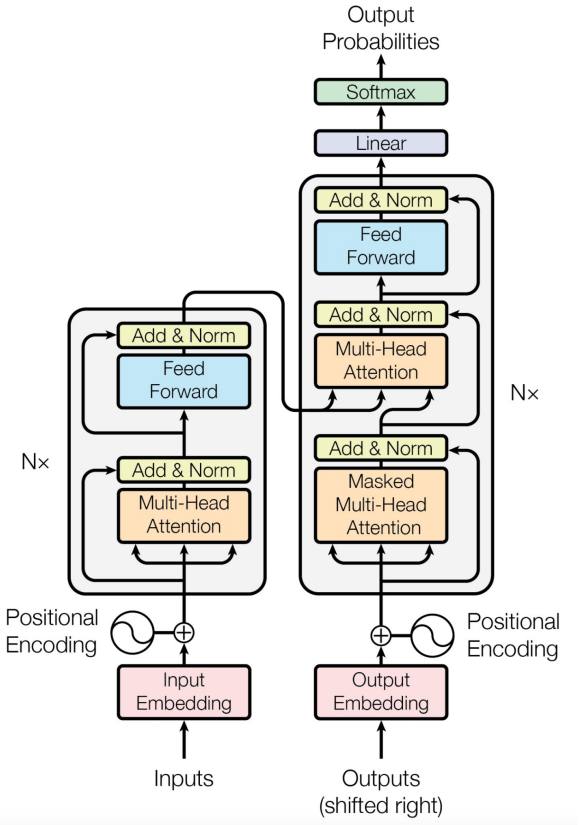
Recap

Beyond Transformer-based LLMs

Diffusion LLMs

Closing thoughts

Can we use Transformers for other things?



Context

- Introduced in 2017 for machine translation
- Relies on **self-attention**
- Concepts of query, key, value

Benefits

- Weaker inductive biases
- More generalizability

"*Attention Is All You Need*", *Vaswani et al., 2017.*

Can we use Transformers for other things?

Context

- Introduced in 2017 for machine translation
- Relies on **self-attention**
- Concepts of query, key, value

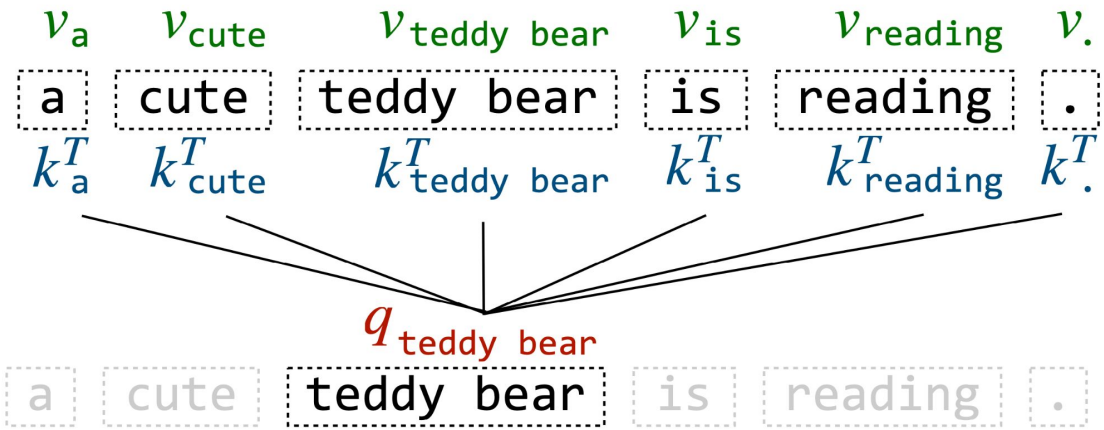
Benefits

- Weaker inductive biases
- More generalizability

"*Attention Is All You Need*", Vaswani et al., 2017.

Refresher on attention mechanism

Concept of **Q**uery, **K**ey, **V**alue



Attention mechanism...

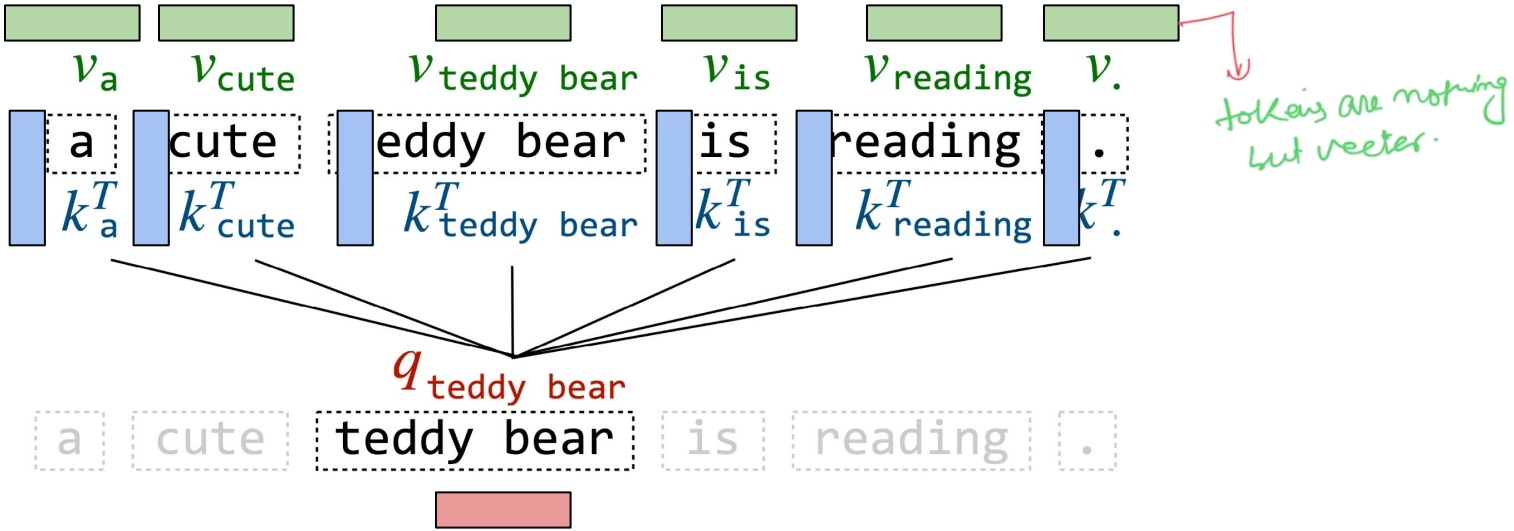


Figure adapted from "Super Study Guide: Transformers & Large Language Models", by Amidi, 2024.

Attention mechanism...

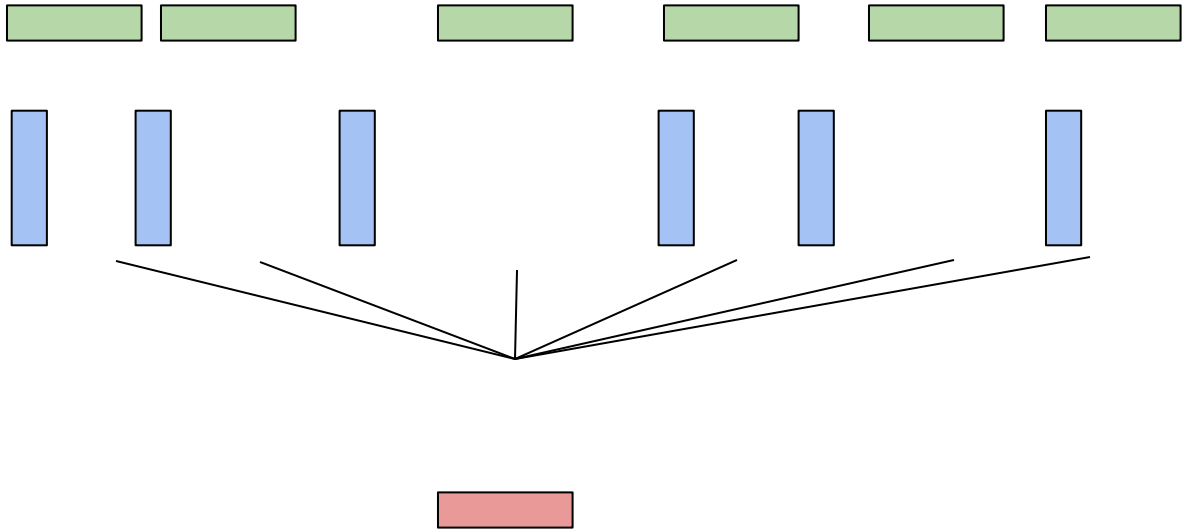


Figure adapted from "Super Study Guide: Transformers & Large Language Models", by Amidi, 2024.

Attention mechanism...for images!

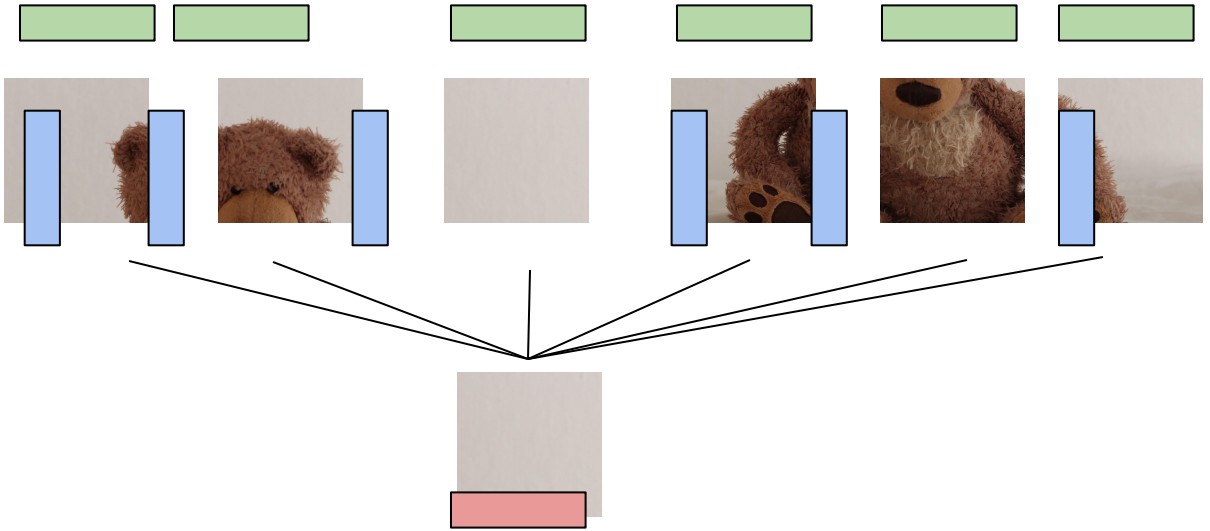
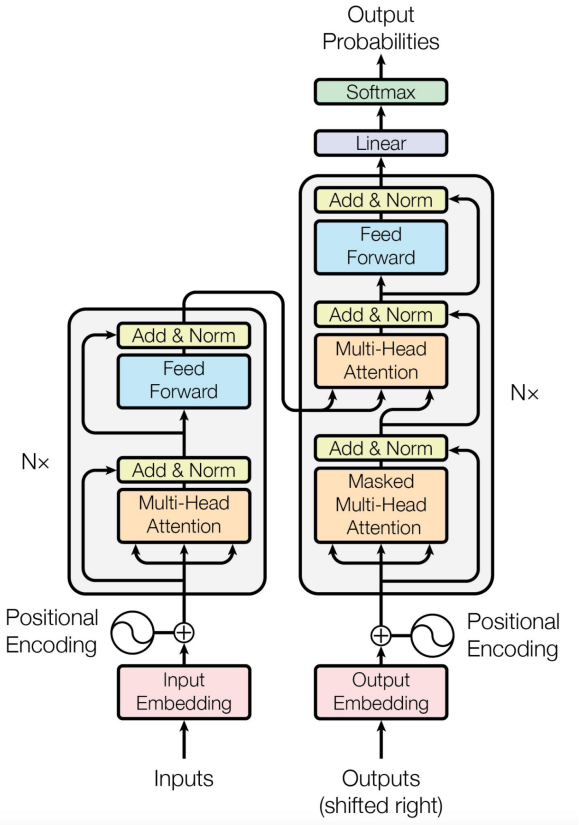


Figure adapted from "Super Study Guide: Transformers & Large Language Models", by Amidi, 2024.

Adapting Transformer architecture for images



How to adapt Transformer for **image understanding** tasks?

"Attention Is All You Need", Vaswani et al., 2017.

Adapting Transformer architecture for images

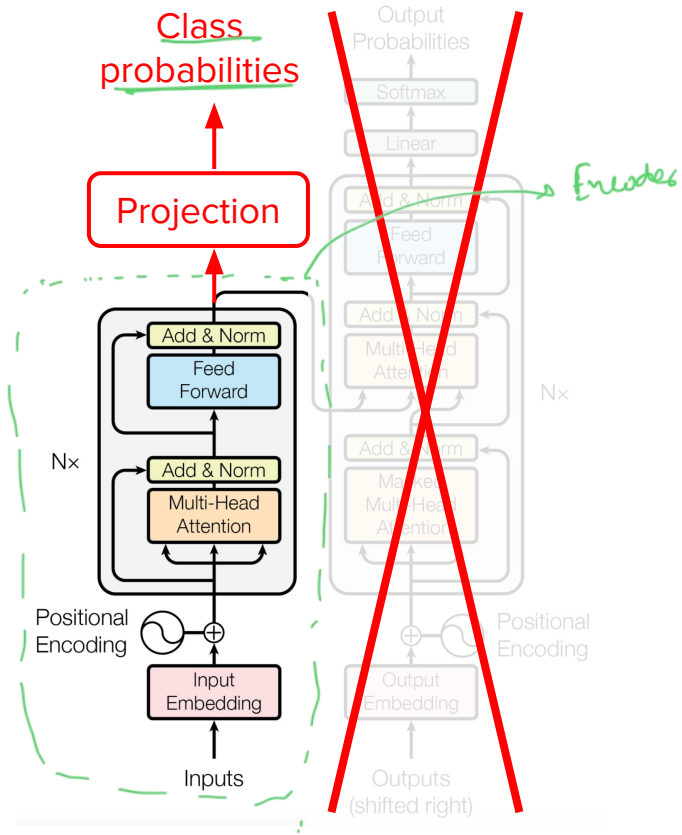
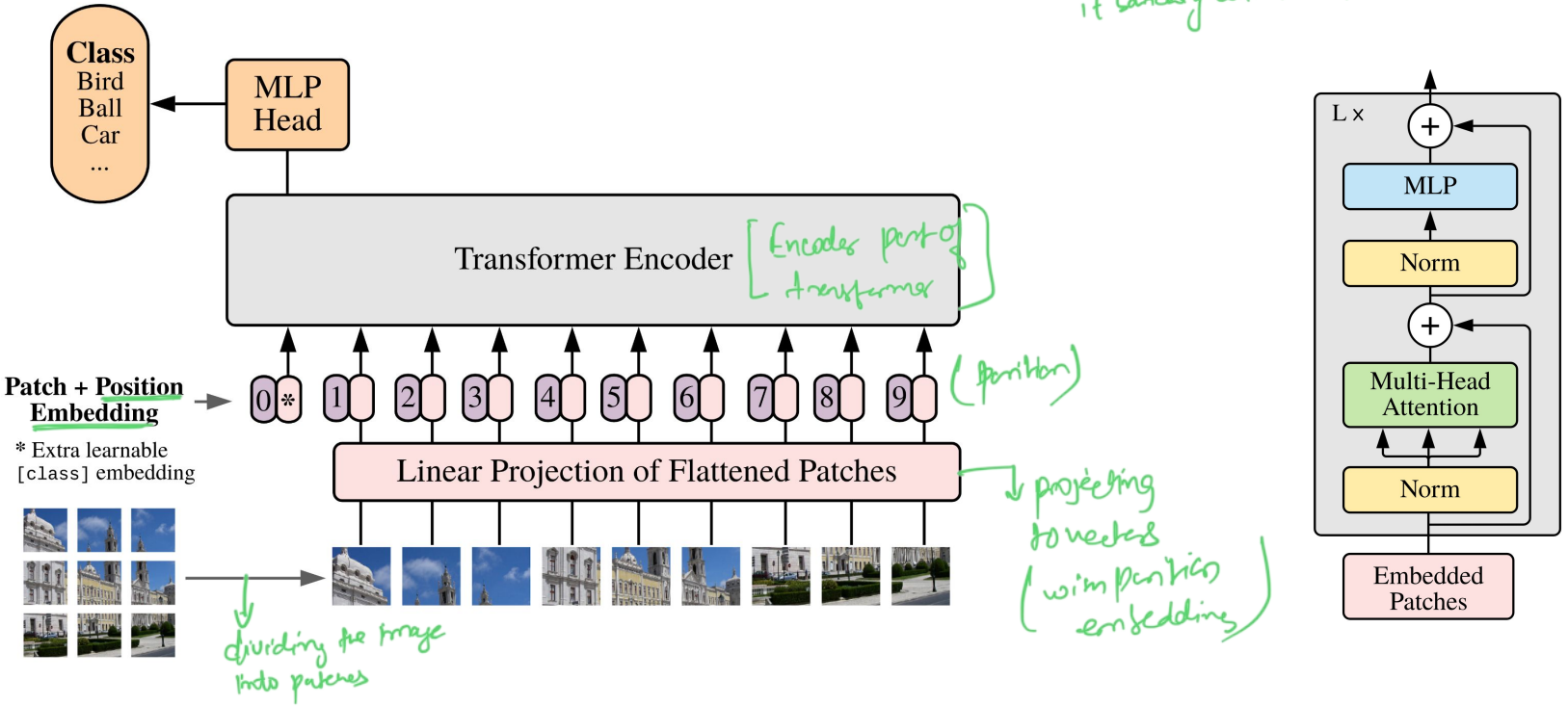


Figure adapted from "Attention Is All You Need", Vaswani et al., 2017.

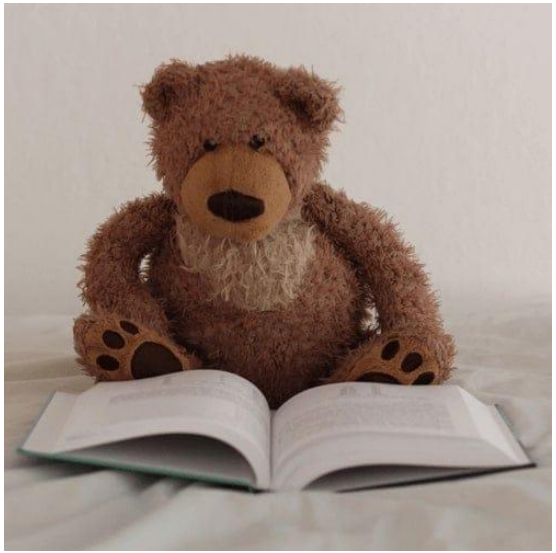
Vision Transformer

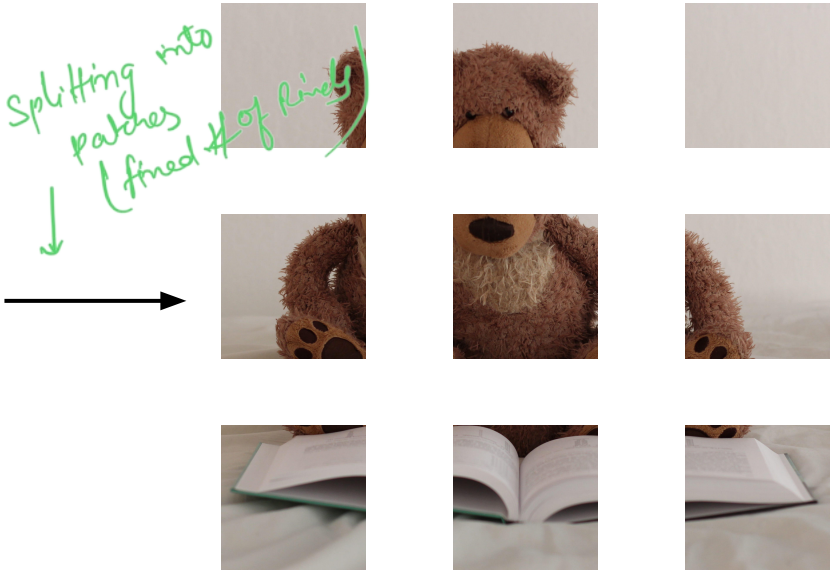
ViT = Vision Transformer



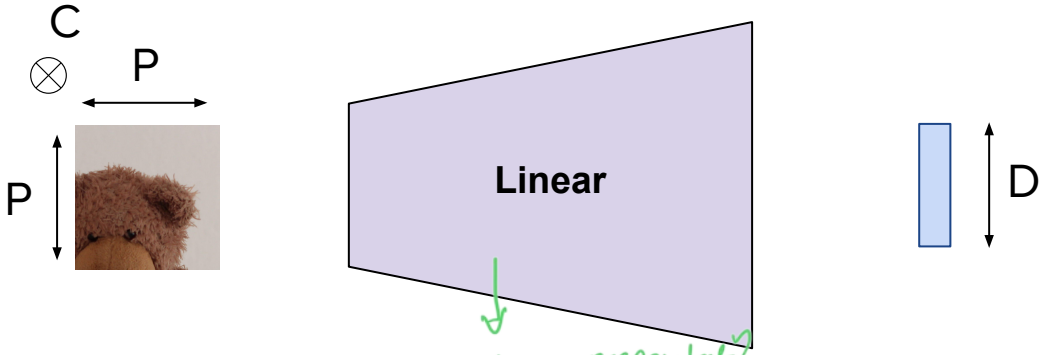
"An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", Dosovitskiy et al., 2020.

End-to-end example using ViT





End-to-end example using ViT

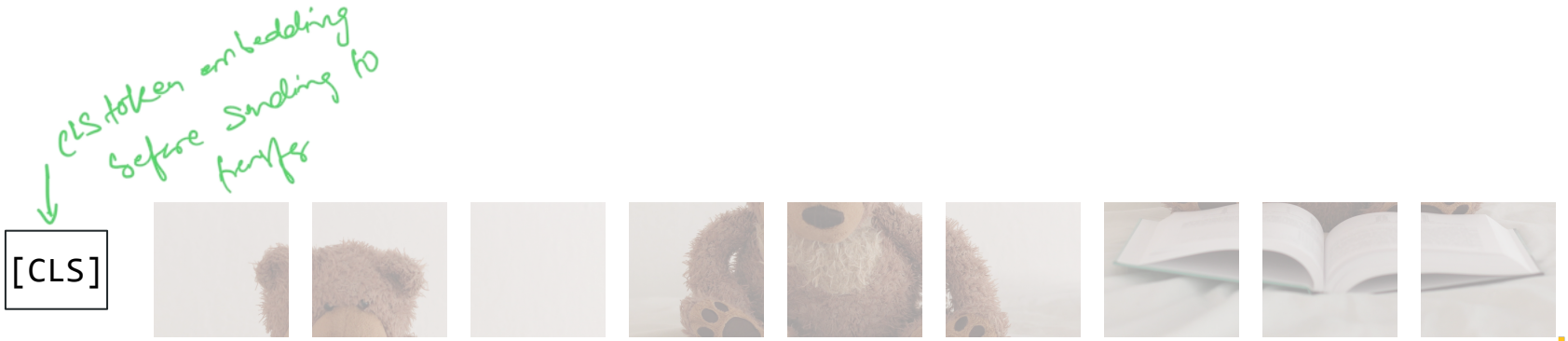


Vector representation
(projected to some lower dimension flattened space)

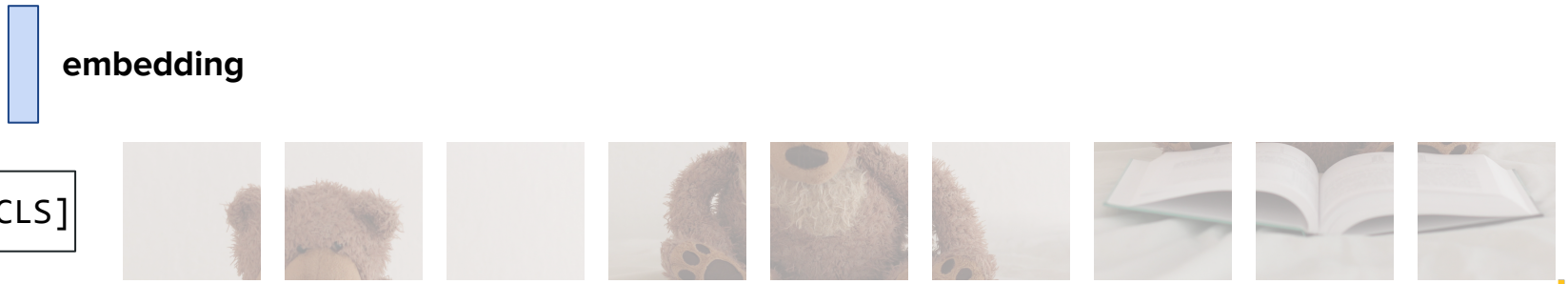
i.e. finding a way to associate a vector to each of the patches that represent every single one of the input

End-to-end example using ViT

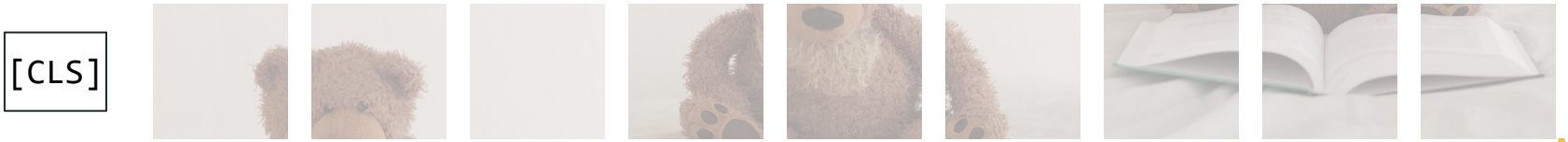
End-to-end example using ViT



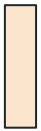
End-to-end example using ViT



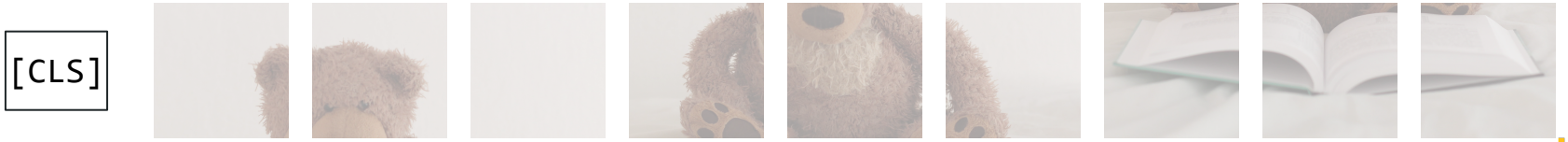
End-to-end example using ViT



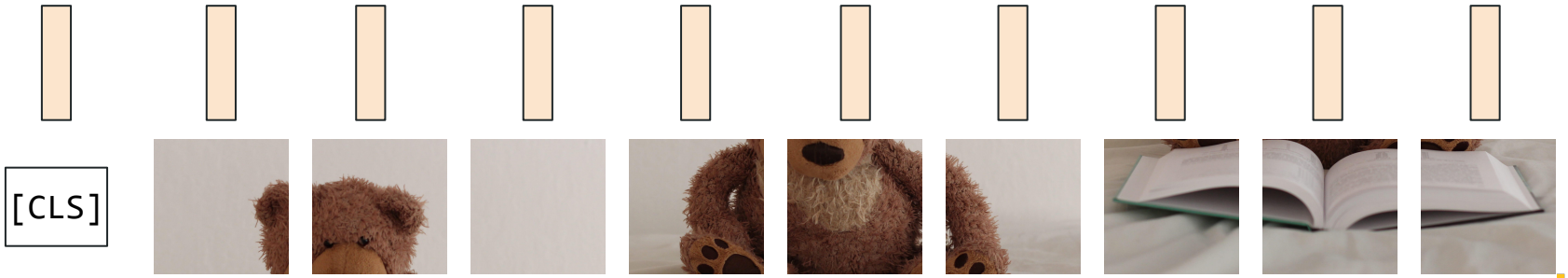
End-to-end example using ViT



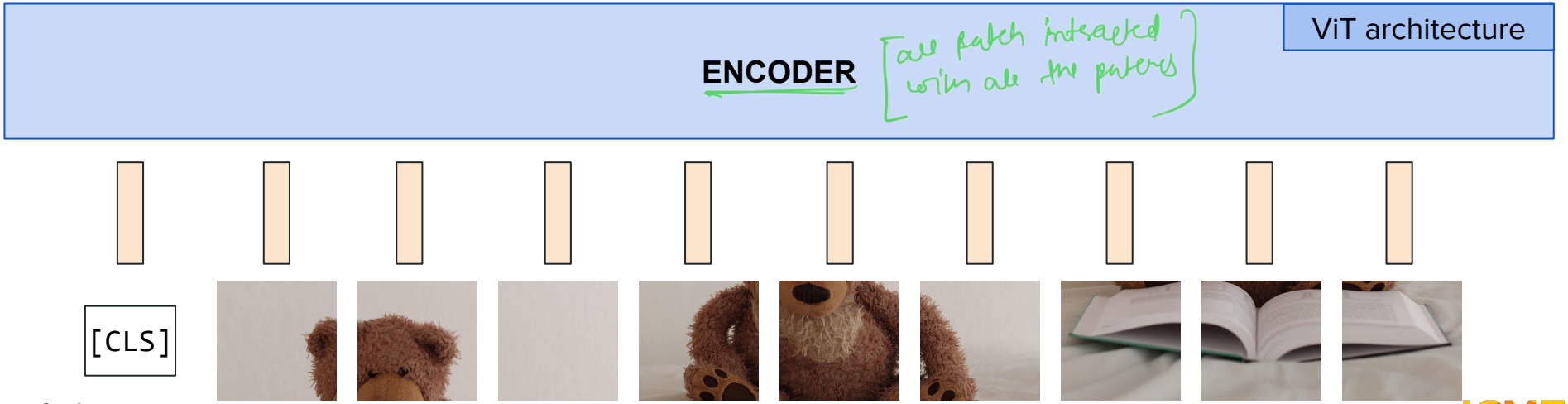
position-aware embedding



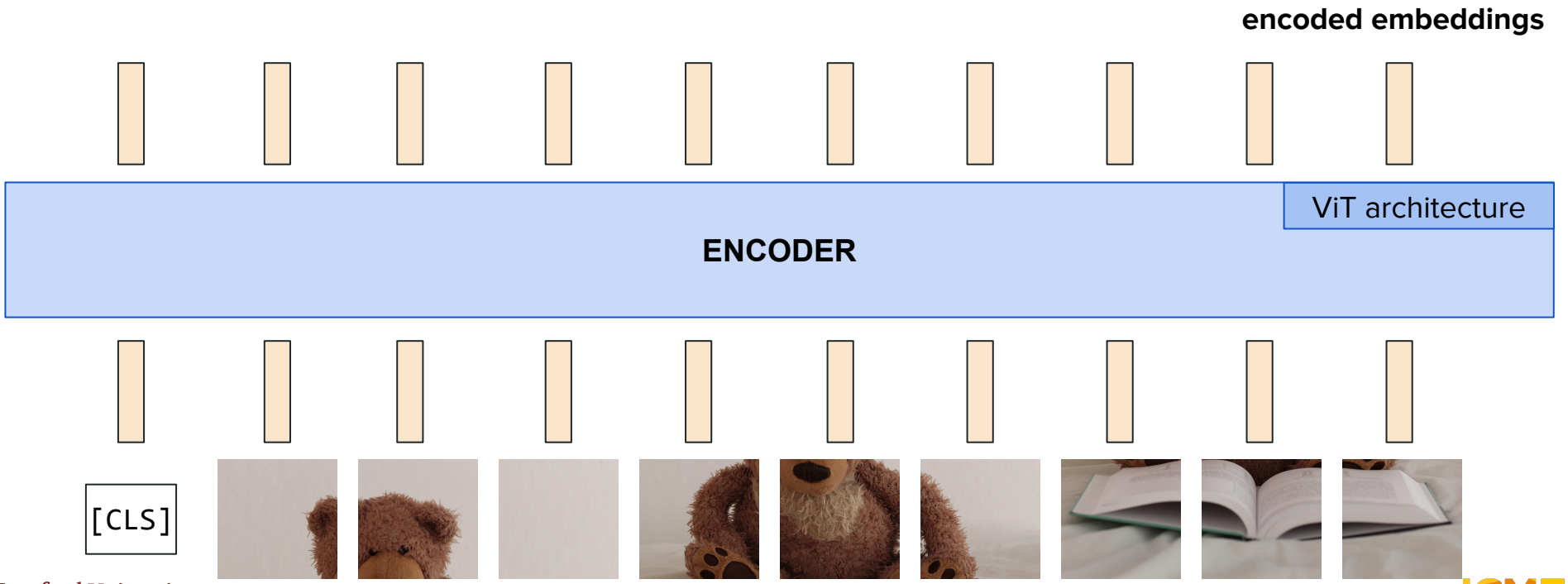
End-to-end example using ViT



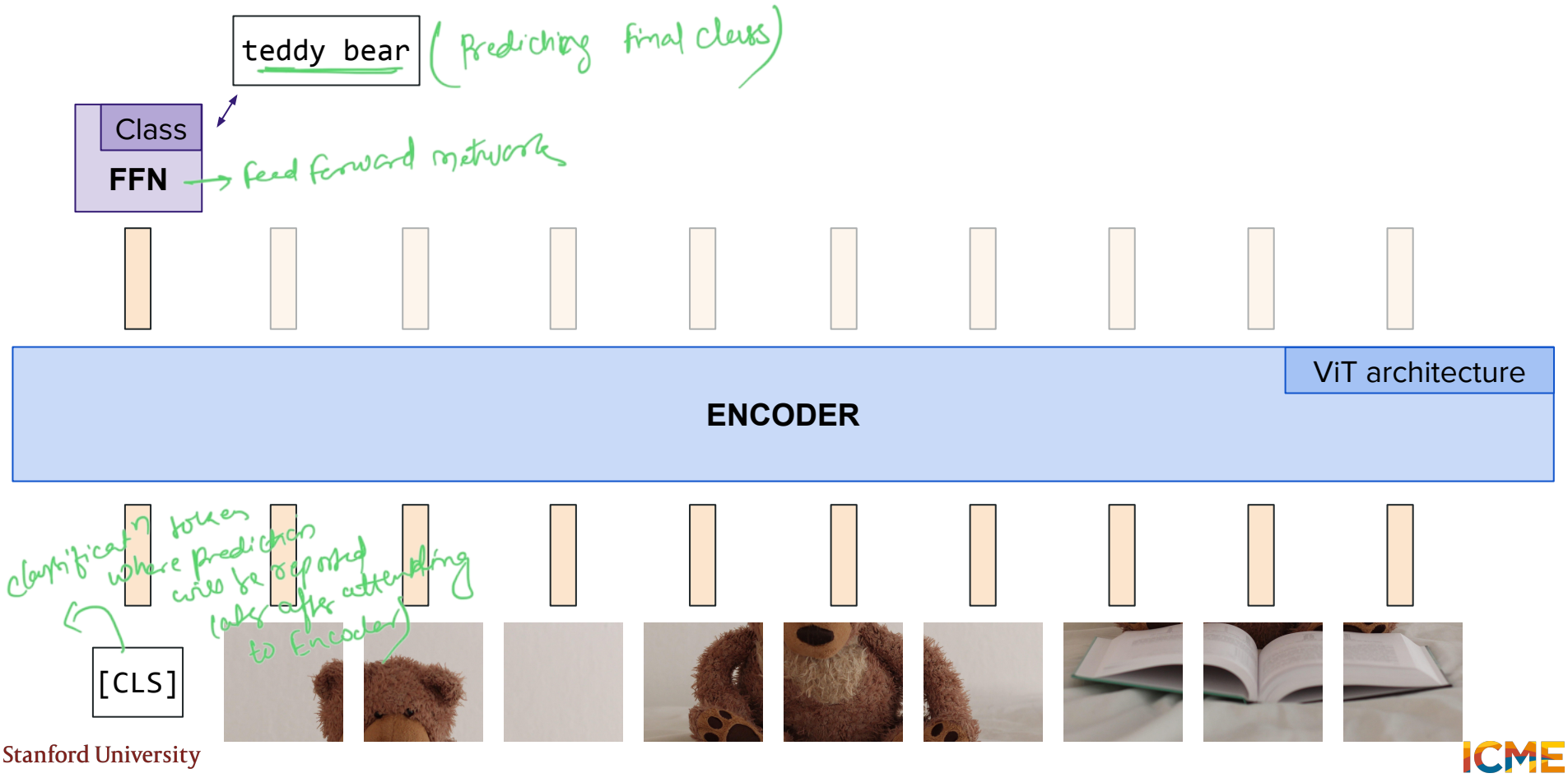
End-to-end example using ViT



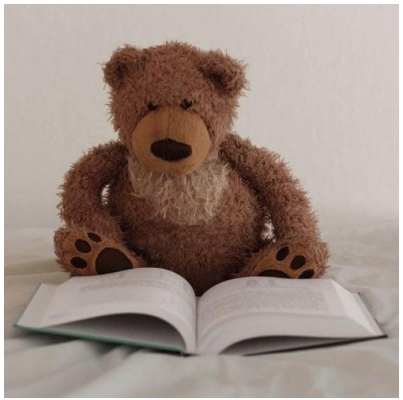
End-to-end example using ViT



End-to-end example using ViT



VLM = **V**ision **L**anguage **M**odel



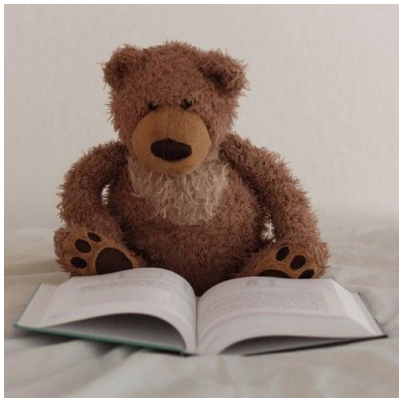
How cute is this teddy bear?

?

Very cute!

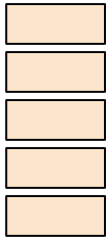
Method 1: recycle decoder-only architecture

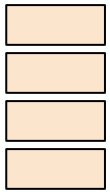
input 1:



input 2:

How cute is this teddy bear?

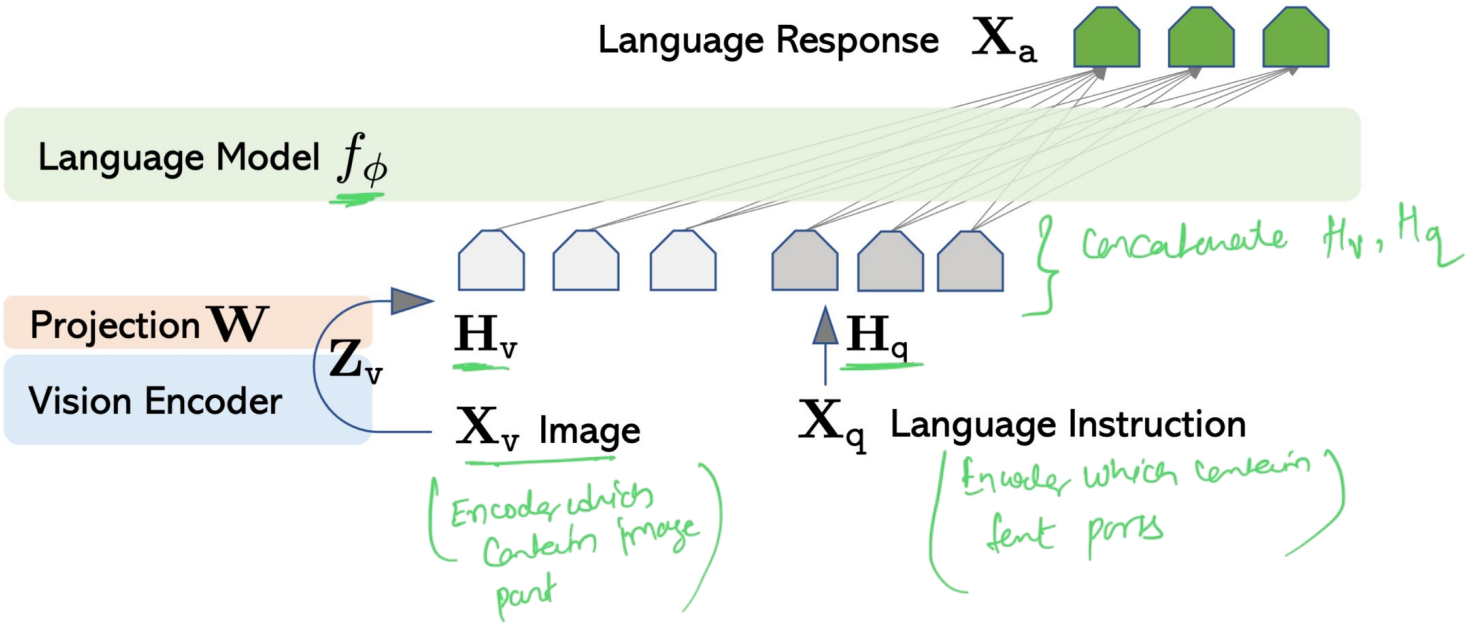




Very cute!

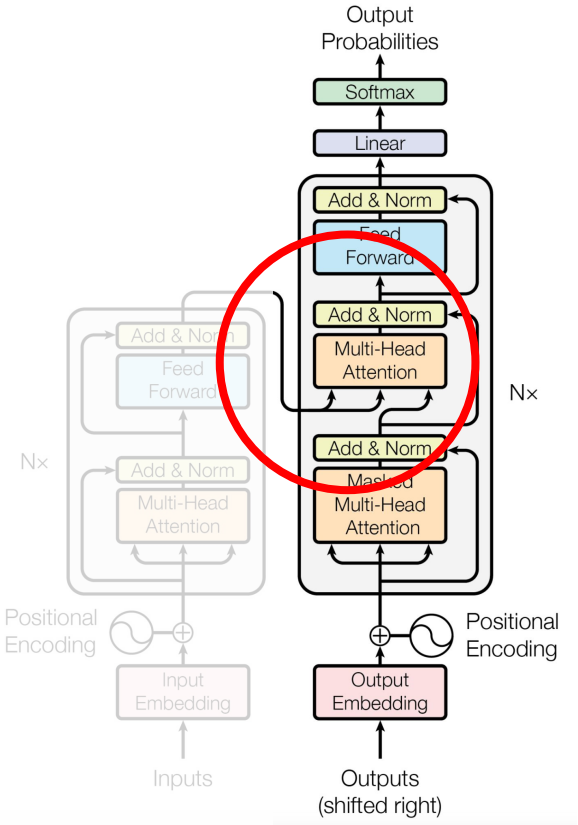
(output)

Method 1: recycle decoder-only architecture

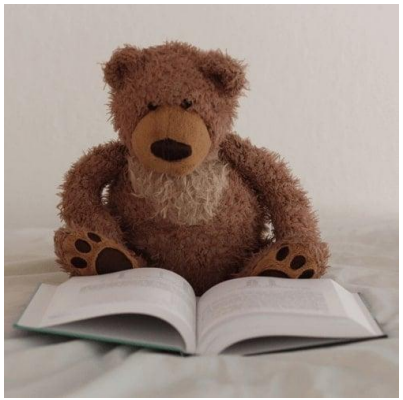


"Visual Instruction Tuning", Liu et al., 2023.

Method 2: leverage cross-attention layer



Method 2: leverage cross-attention layer

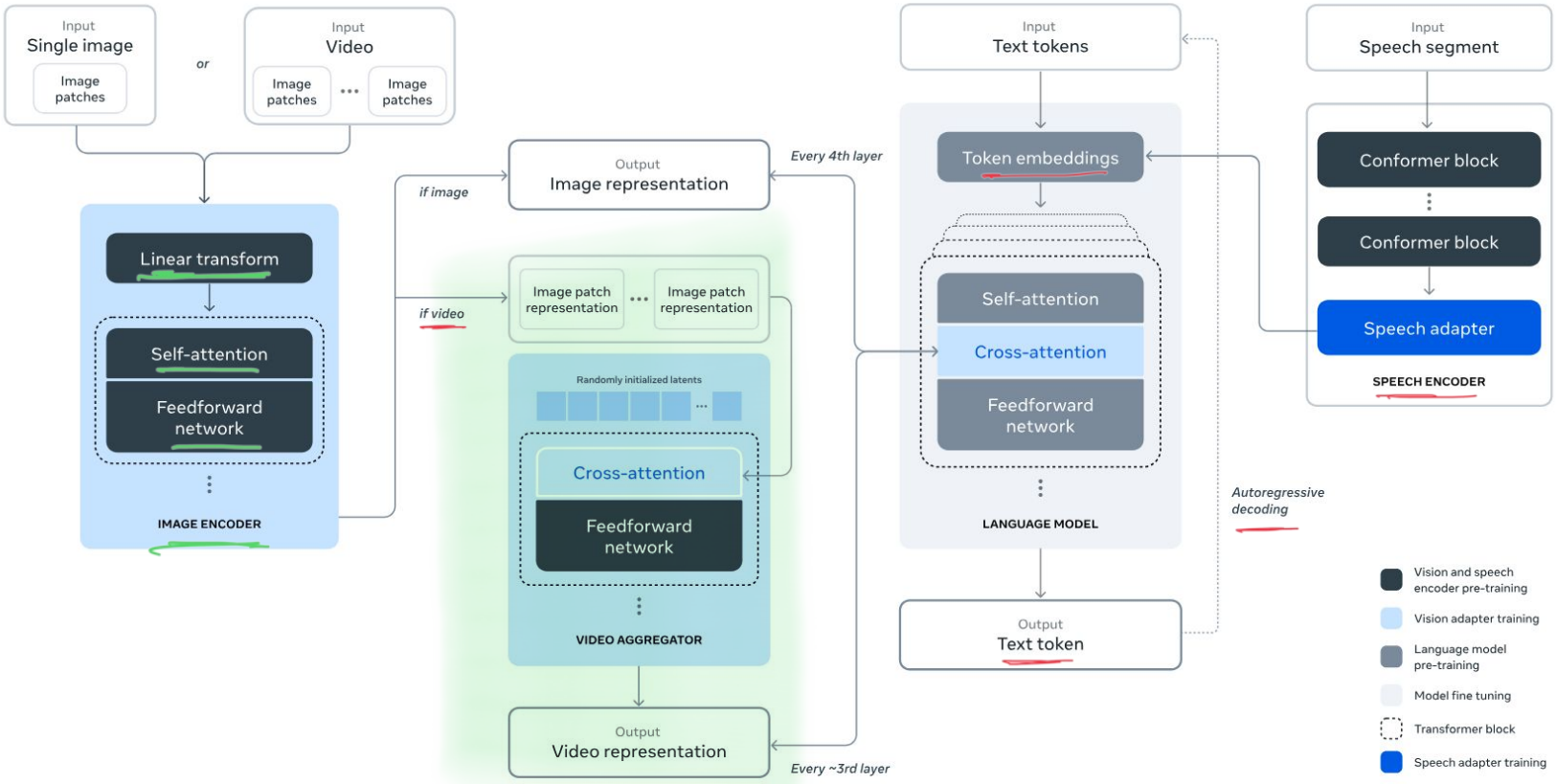


How cute is this teddy bear?

Decoder with cross-attention

Very cute!

Method 2: leverage cross-attention layer – Llama



"The Llama 3 Herd of Models", Llama Team, 2024.

Transformer-based architectures

Transformers are actually used in a lot of different places!

- **Text generation**. CME 295!
- **Vision understanding**. Vision Transformer (ViT)
- **Image generation**. Diffusion Transformer (DiT), MultiModal-DiT (MM-DiT), etc.

and many others (recommendation, speech, etc.)

Transformers & Large Language Models

Recap

Beyond Transformer-based LLMs

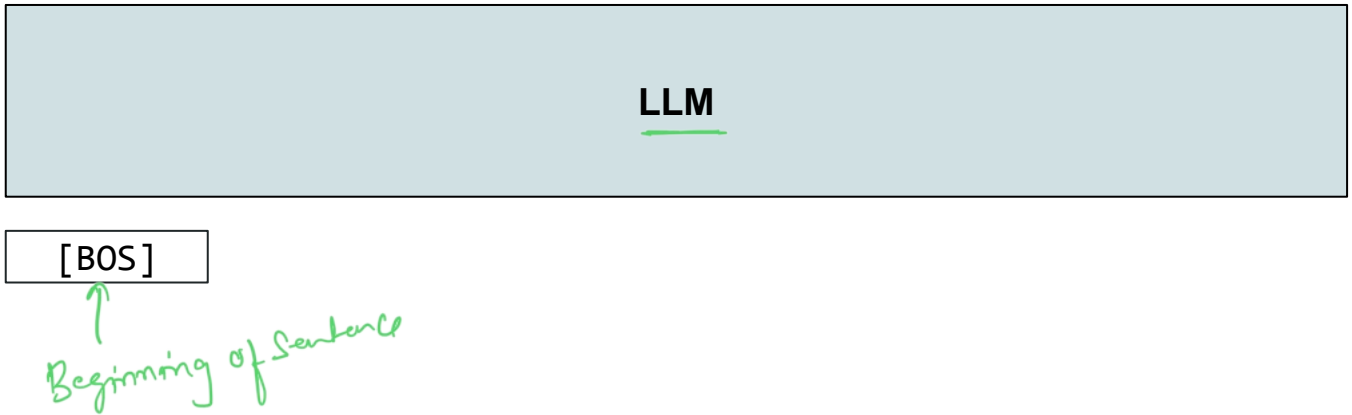
Diffusion LLMs

Closing thoughts

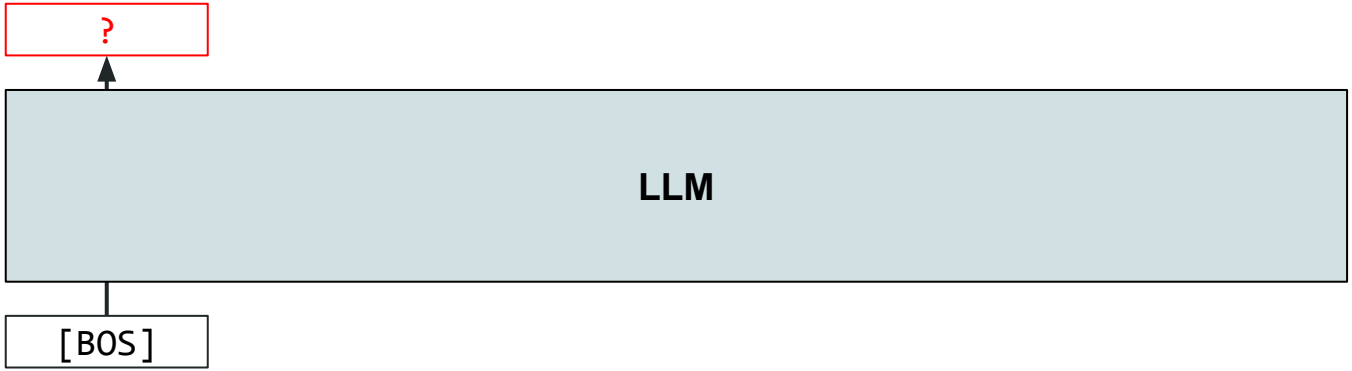
Diffusion basically being introduced in vision based tasks but diffusion concept can also be adapted in LLM world.

Current approach and limitations

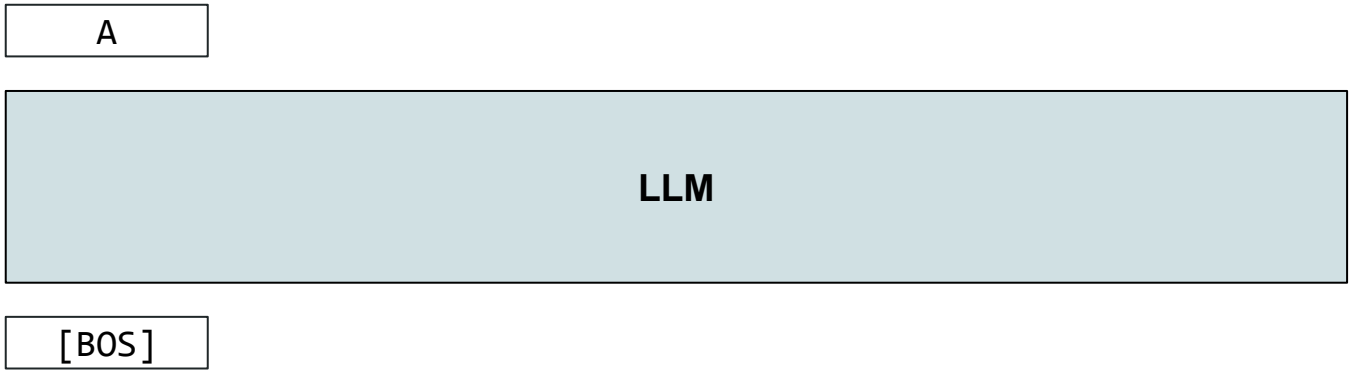
— LLM is autoregressive & often!



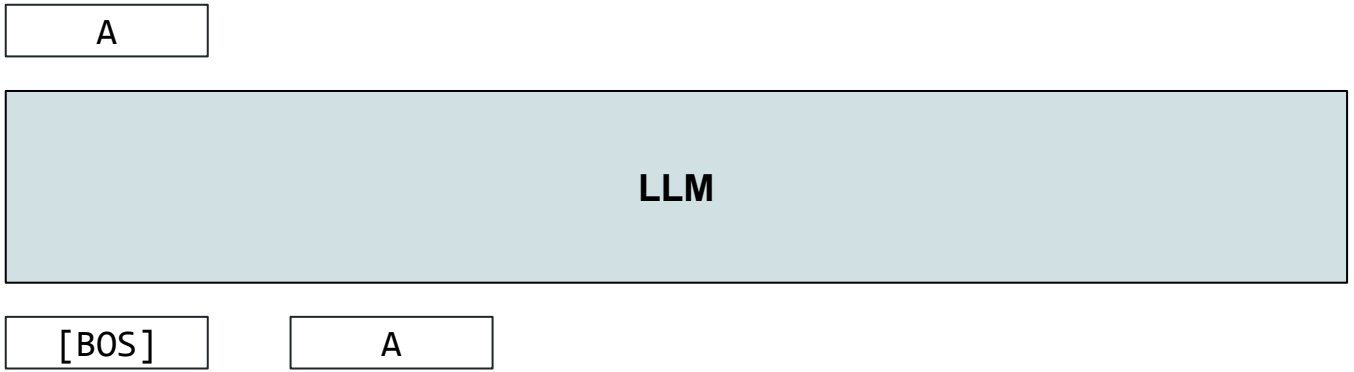
Current approach and limitations



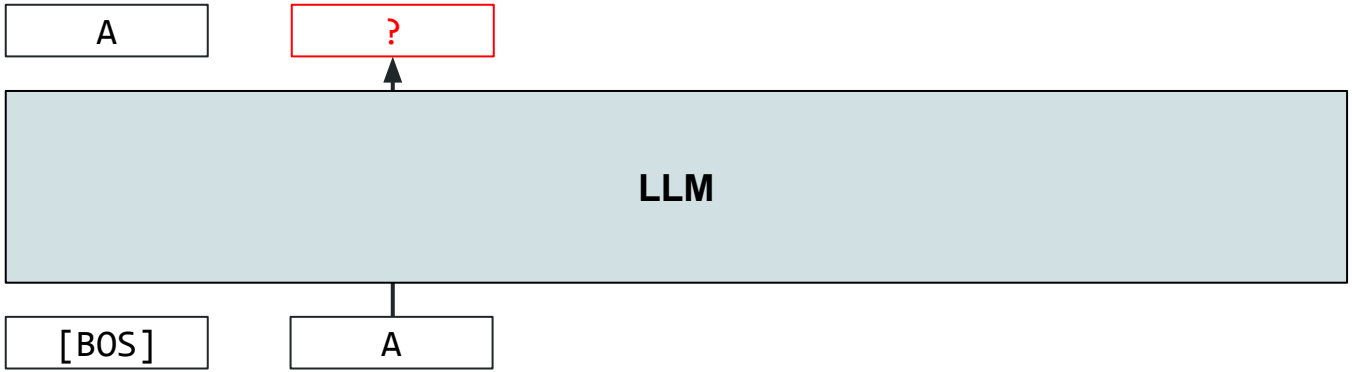
Current approach and limitations



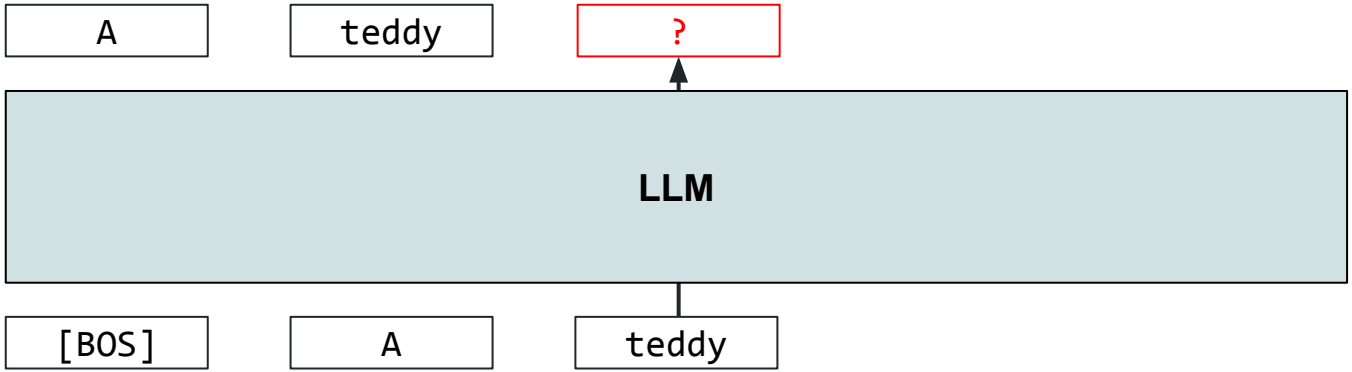
Current approach and limitations



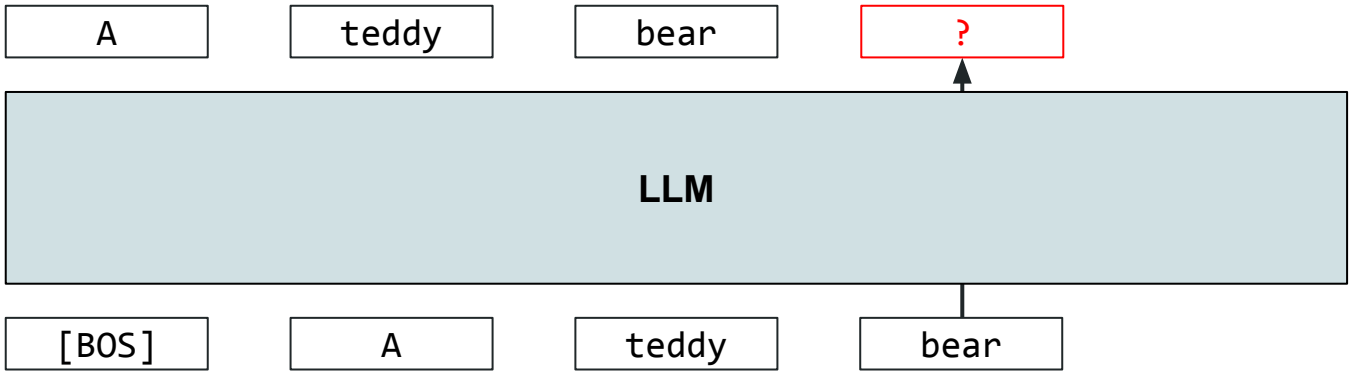
Current approach and limitations



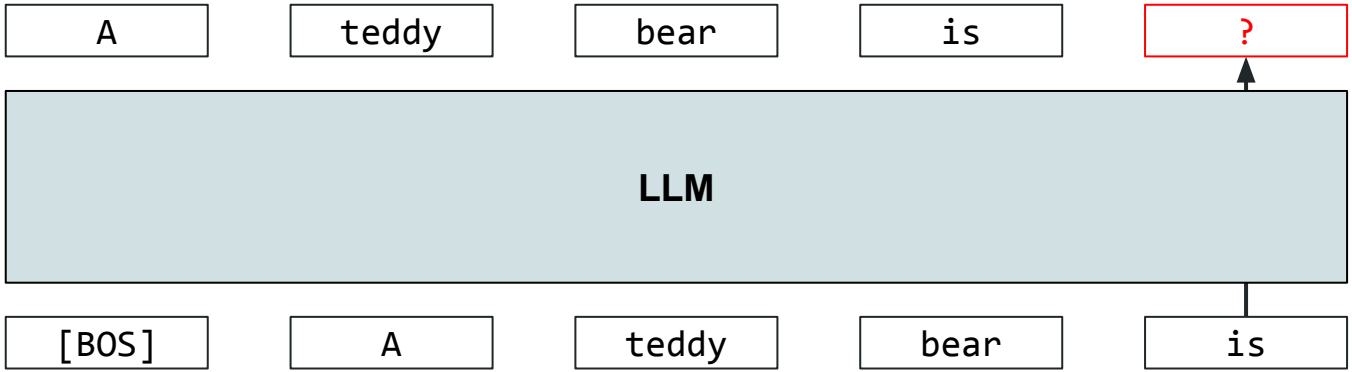
Current approach and limitations



Current approach and limitations

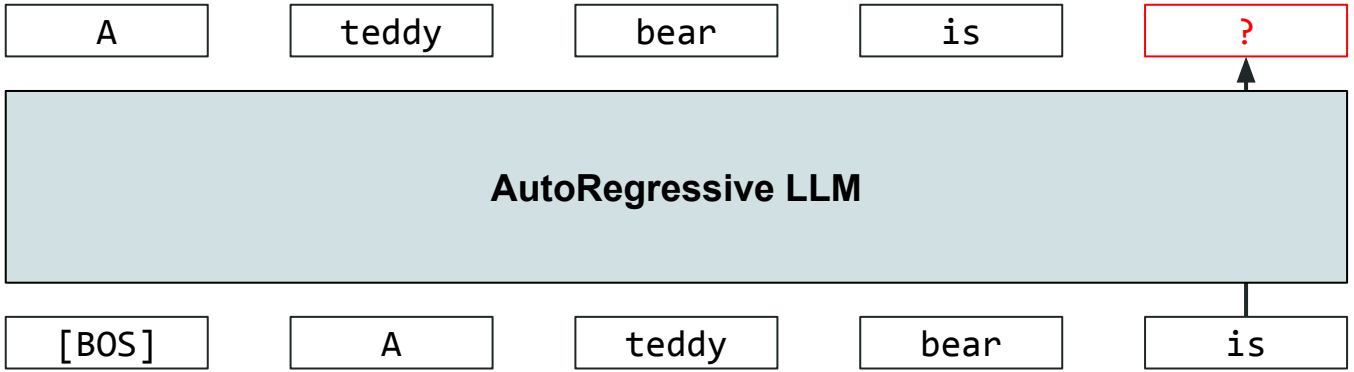


Current approach and limitations



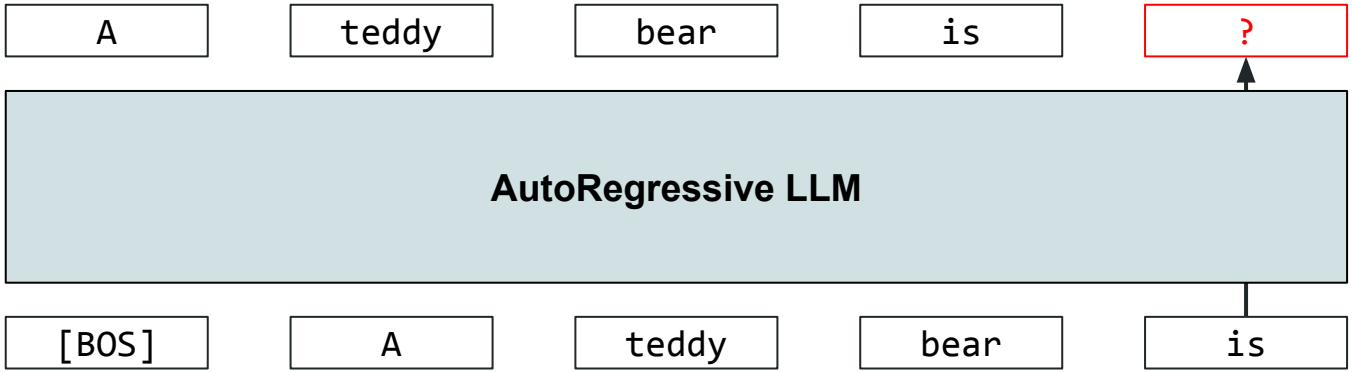
Current approach and limitations

Paradigm. **A**uto**R**egressive **M**odeling (ARM)



Current approach and limitations

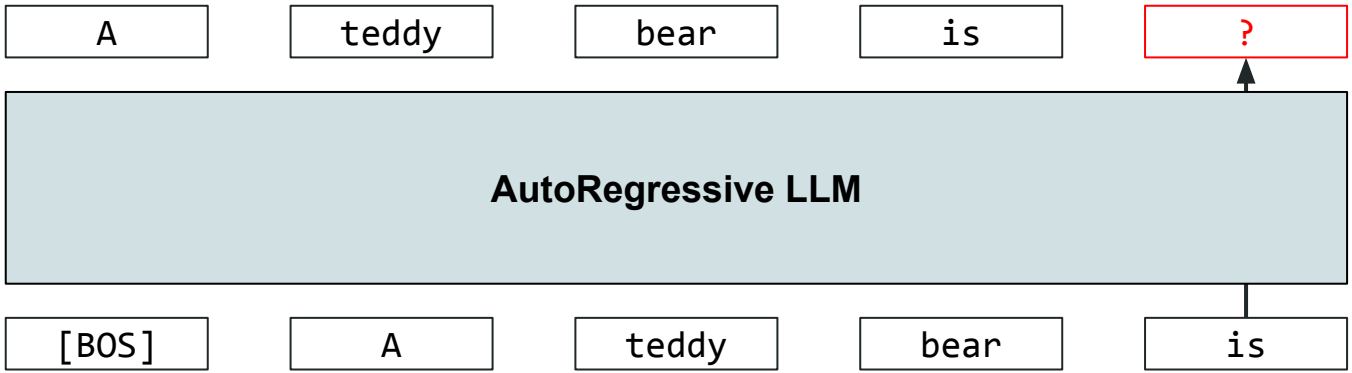
Paradigm. **A**uto**R**egressive **M**odeling (ARM)



Problem. Inference-time generation is not parallelizable

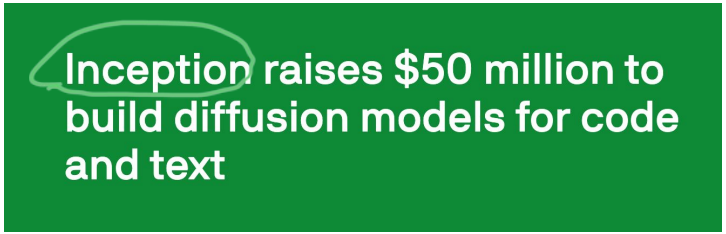
Current approach and limitations

Paradigm. **A**uto**R**egressive **M**odeling (ARM)



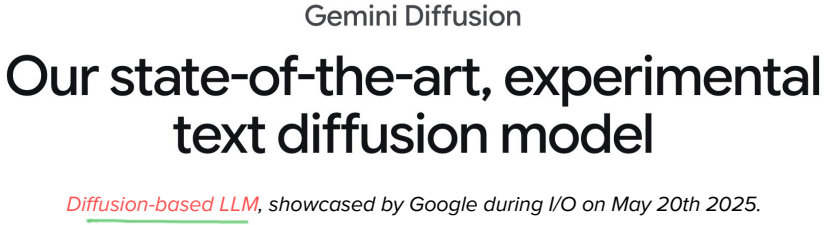
Problem. Inference-time generation is not parallelizable (although training is)

Alternative approach in the news



Announcement on TechCrunch, November 6th 2025.

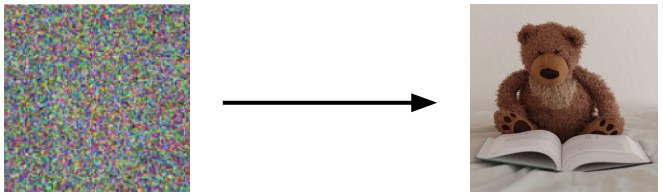
Inception website, screenshot taken on November 26th 2025.



Diffusion-based LLM, showcased by Google during I/O on May 20th 2025.

ByteDance announcement, July 31st 2025.

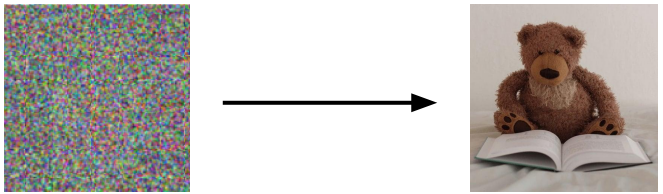
Context on image generation models



Context on image generation models

Intuition (why noise?)

- Noise is easy to sample
- Learned transformation
- Mathematically well-defined

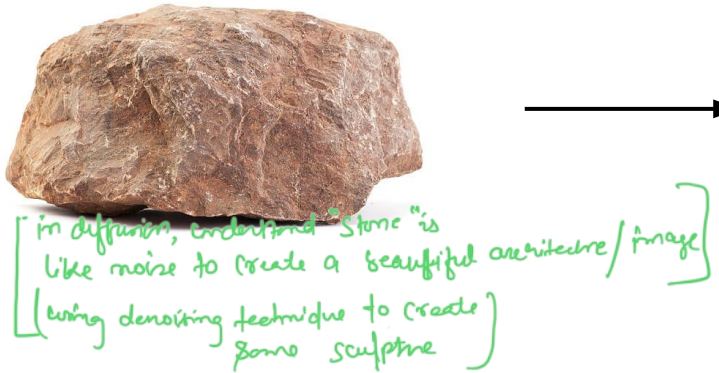


Context on image generation models

Intuition

- Noise is easy to sample
- Learned transformation
- Mathematically well-defined

Analogy. Building a sculpture –



Context on image generation models

Intuition

- Noise is easy to sample
- Learned transformation
- Mathematically well-defined

Analogy. Building a sculpture

> *The sculpture is already complete within the marble block, before I start my work. It is already there, I just have to chisel away the superfluous material.*
>
> –Michelangelo

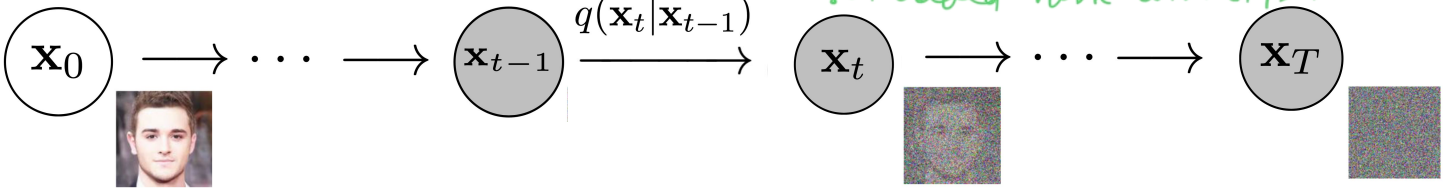
High-level goal of image generation models

Goal

Learn *some transformation* that goes from **noise** to **desired data distribution**

Intuition behind diffusion for images

1. Add **noise** to image (**forward process**)



noising step in diffusion where input is taken as clean image but added noise with steps to create "Extreme" noisy image from a clean image

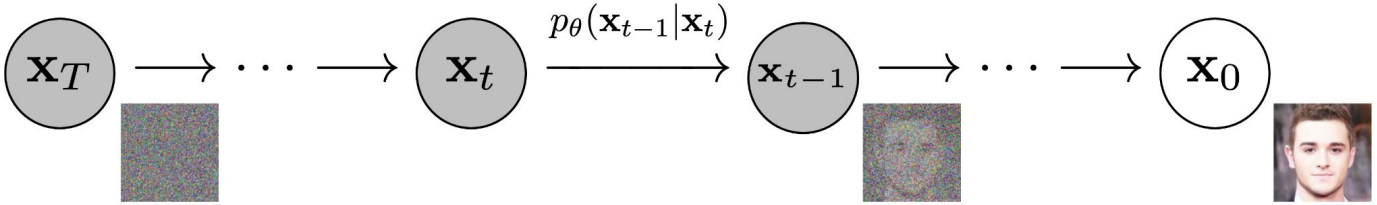
"Denoising Diffusion Probabilistic Models", Ho et al., 2020.

Intuition behind diffusion for images

1. Add **noise** to image (**forward process**)

2. Learn to **denoise** it (**reverse process**)

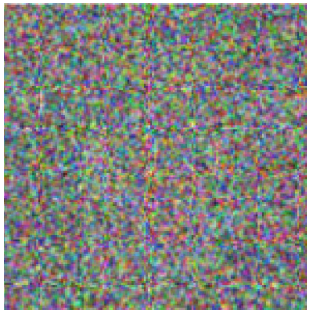
denoising process where Entropy noisy image is being used to create a clean image.



"Denoising Diffusion Probabilistic Models", Ho et al., 2020.

From image to text

but how to adapt diffusion to LLM?



Image

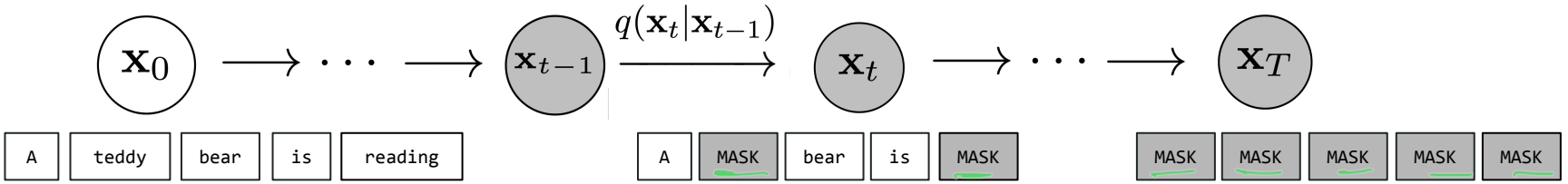
MASK

Text

concept of MASK is being used in LLM is like "hide" some token for model to input and later ask to predict the "mask"/"hidden" text/token.

Intuition behind diffusion for text

1. Mask tokens with some probability (**forward process**)



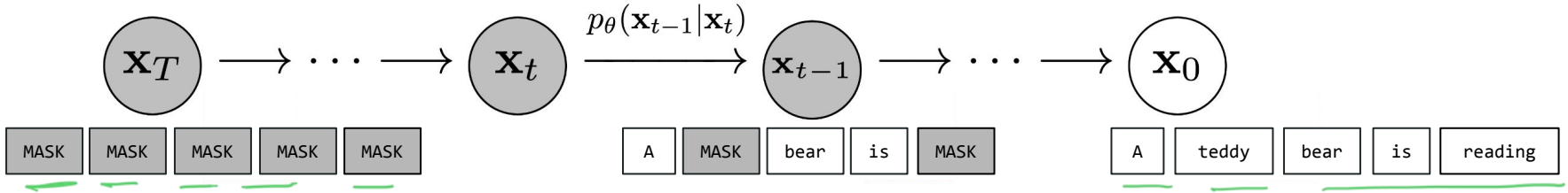
↓

In forward process of "diffusion for text" like noise is adding and denoising happens for the case of vision to diffusion – adding as much as token as mask in text input w.r.t diffusion for text and removing mask as much as token in reverse process of diffusion in text like denoising in vision.

Intuition behind diffusion for text

1. Mask tokens with some probability (**forward process**)

2. Learn to **unmask** tokens (**reverse process**)



"*Denoising Diffusion Probabilistic Models*", Ho et al., 2020.

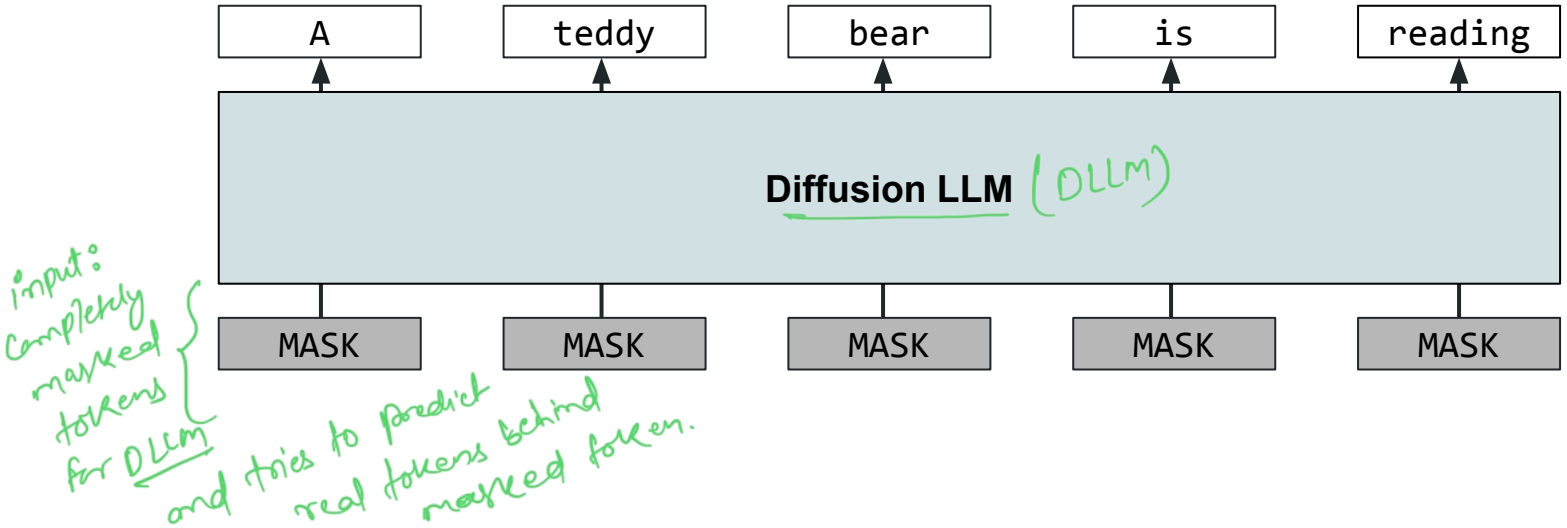
Diffusion in LLMs

MDM = **M**asked **D**iffusion **M**odel

DLLM - Diffusion based LLM

Diffusion in LLMs

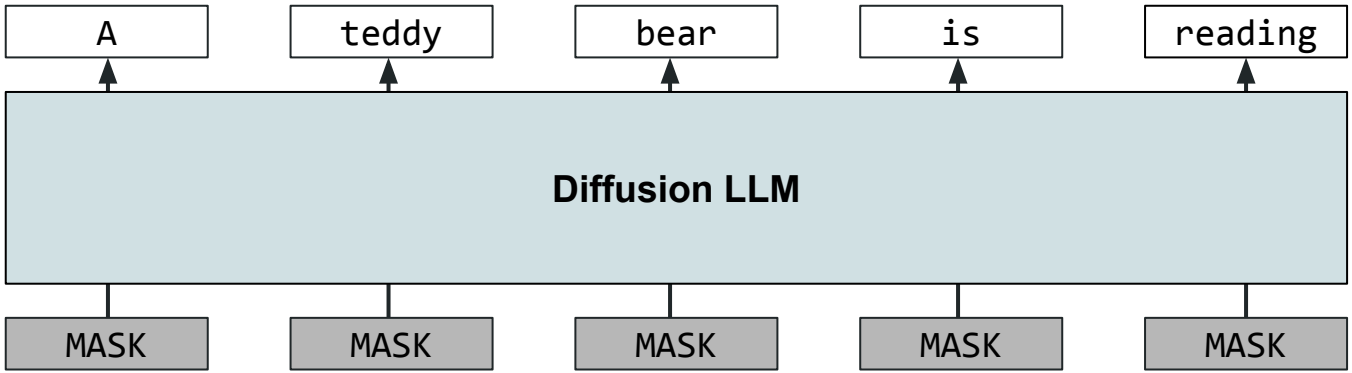
MDM = **M**asked **D**iffusion **M**odel



Diffusion in LLMs

MDM = **M**asked **D**iffusion **M**odel

Decoding done in fewer forward passes! — Like writing a script (after multiple input & refinement)

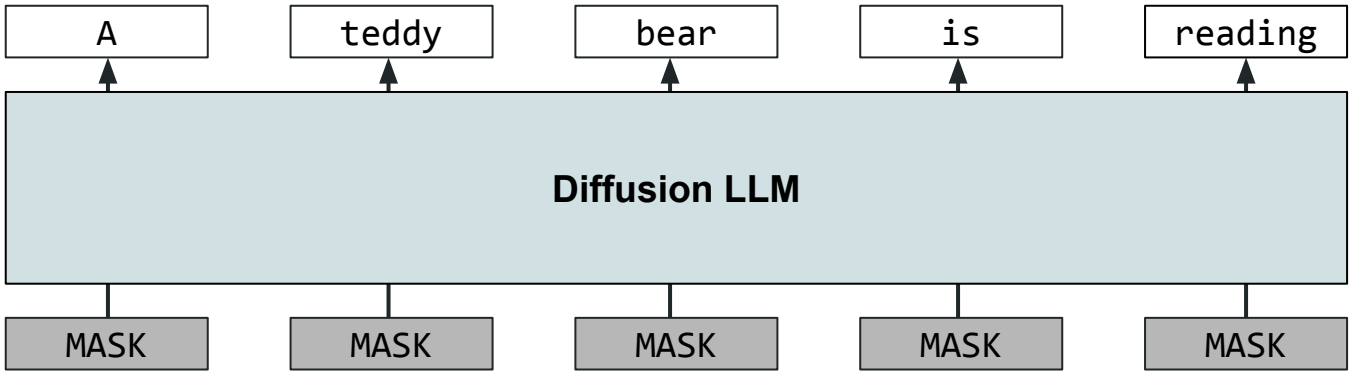


Diffusion in LLMs

MDM = **M**asked **D**iffusion **M**odel

Decoding done in fewer forward passes!

Previously in LLM, we need to have as many as forward passes as much as we have had tokens but here, we need lesser forward process to diffusion process



Suggested readings:

- *"Discrete Diffusion Modeling by Estimating the Ratios of the Data Distribution", Lou et al., 2023.*
- *"Simple and Effective Masked Diffusion Language Models", Sahoo et al., 2024.*
- *"Large Language Diffusion Models", Nie et al., 2025.*

Discussion

Advantages.

- ⚡ 10x output tokens per second compared to ARM
- 💻 Better suited for some tasks

Discussion

Advantages.

- ⚡ 10x output tokens per second compared to ARM
- 💻 Better suited for some tasks

Challenges and current work.

- Performance
- Adapt ARM-based techniques

Transformers & Large Language Models

Recap

Beyond Transformer-based LLMs

Diffusion LLMs

Closing thoughts

Cross-pollination between modalities

Trend. Modalities trade ideas on concepts that work

Cross-pollination between modalities

Trend. Modalities trade ideas on concepts that work

Architecture. lower latency, higher accuracy for Text format.

- "Diffusion" training for text output

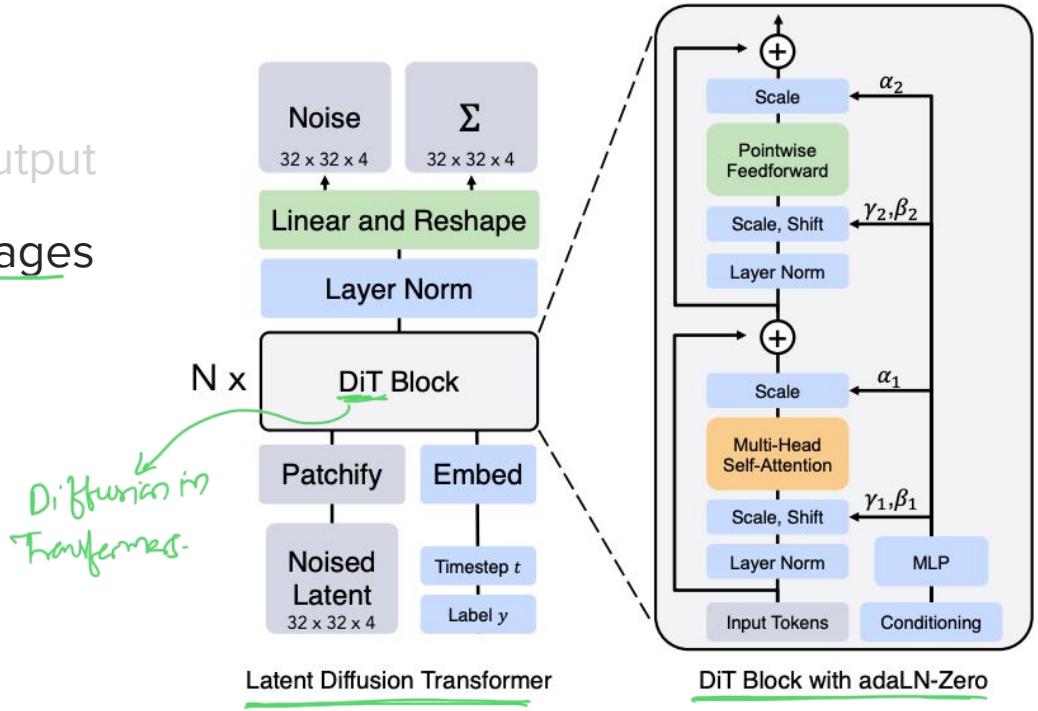
"*Large Language Diffusion Models*", Nie et al., 2025.

Cross-pollination between modalities

Trend. Modalities trade ideas on concepts that work

Architecture.

- "Diffusion" training for text output
- Transformers to deal with images



"Scalable Diffusion Models with Transformers", Peebles et al., 2022.

Cross-pollination between modalities

Trend. Modalities trade ideas on concepts that work

Architecture.

- "Diffusion" training for text output
- Transformers to deal with images

Input representation. DeepSeek-OCR

"DeepSeek-OCR: Contexts Optical Compression", Wei et al., 2025.

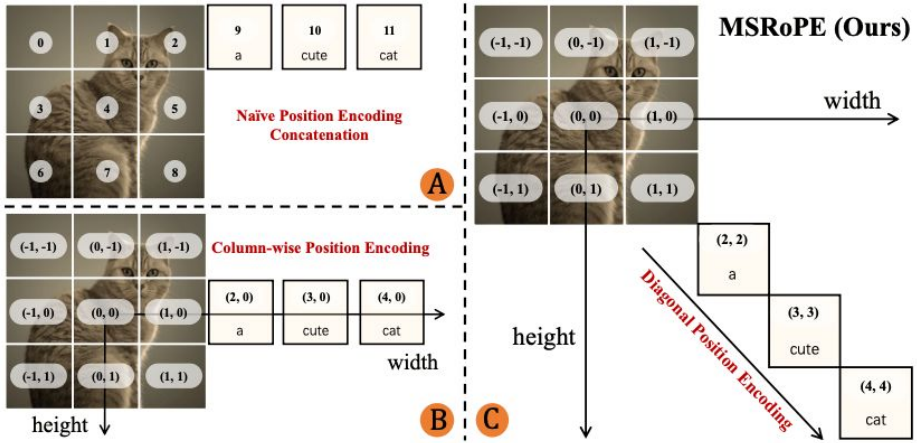
Cross-pollination between modalities

Trend. Modalities trade ideas on concepts that work

Architecture.

- "Diffusion" training for text output
- Transformers to deal with images

Input representation. DeepSeek-OCR



Tricks. RoPE variants (e.g. image: Multimodal Scalable RoPE)

relative position in text.

"Qwen-Image Technical Report", Wu et al., 2025.

Back to the basics

Foundational research still very much ongoing.

Back to the basics

Foundational research still very much ongoing.

Microscopic level. Design choices still widely very across papers.

- Optimizers: AdamW vs. MuonClip vs. others
- Normalization?
- MHA/MQA/GQA design choices
- Activation functions
- MoE vs. not
- Number of layers

"Kimi K2: Open Agentic Intelligence", Kimi Team, 2025.

Back to the basics

Foundational research still very much ongoing.

Microscopic level. Design choices still widely very across papers.

- Optimizers: AdamW vs. MuonClip vs. others
- Normalization?
- MHA/MQA/GQA design choices
- Activation functions
- MoE vs. not
- Number of layers

"Fuel". Future streams of high-quality data? — Research on data generated side.

"The Curse of Recursion: Training on Generated Data Makes Models Forget", Shumailov et al., 2023.

Back to the basics

Foundational research still very much ongoing.

Microscopic level. Design choices still widely very across papers.

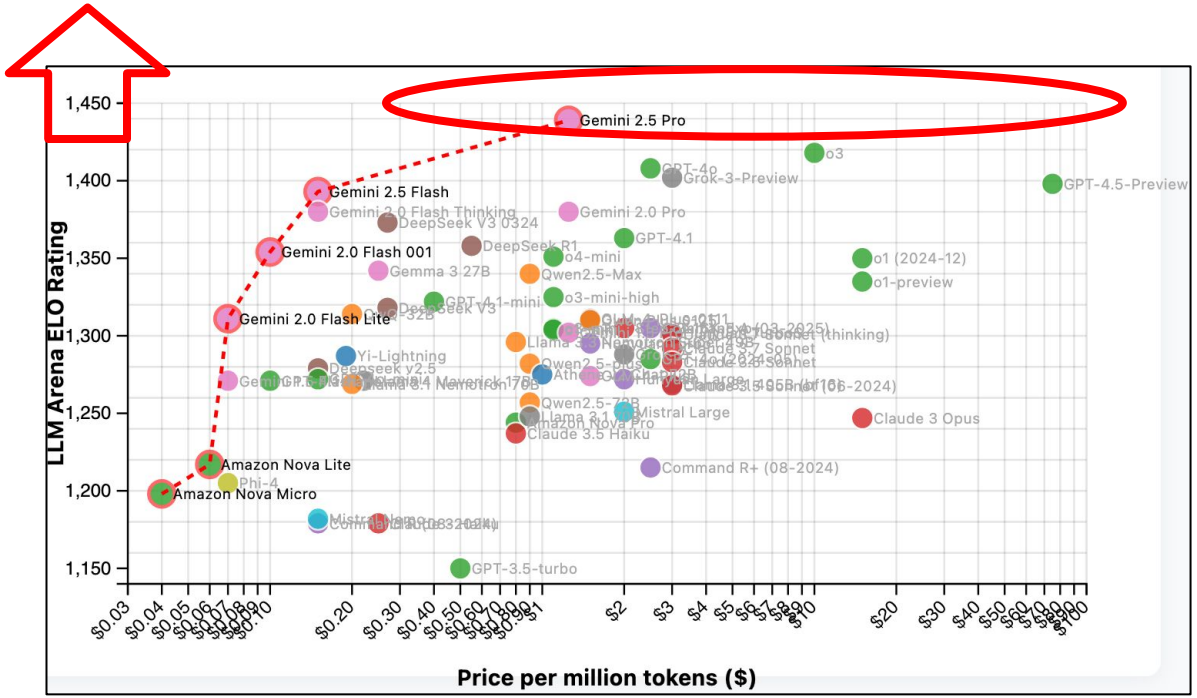
- Optimizers: AdamW vs. MuonClip vs. others
- Normalization?
- MHA/MQA/GQA design choices
- Activation functions
- MoE vs. not
- Number of layers

"Fuel". Future streams of high-quality data?

Taking a step back. Transformer = best architecture?

Interest in pushing through new frontiers

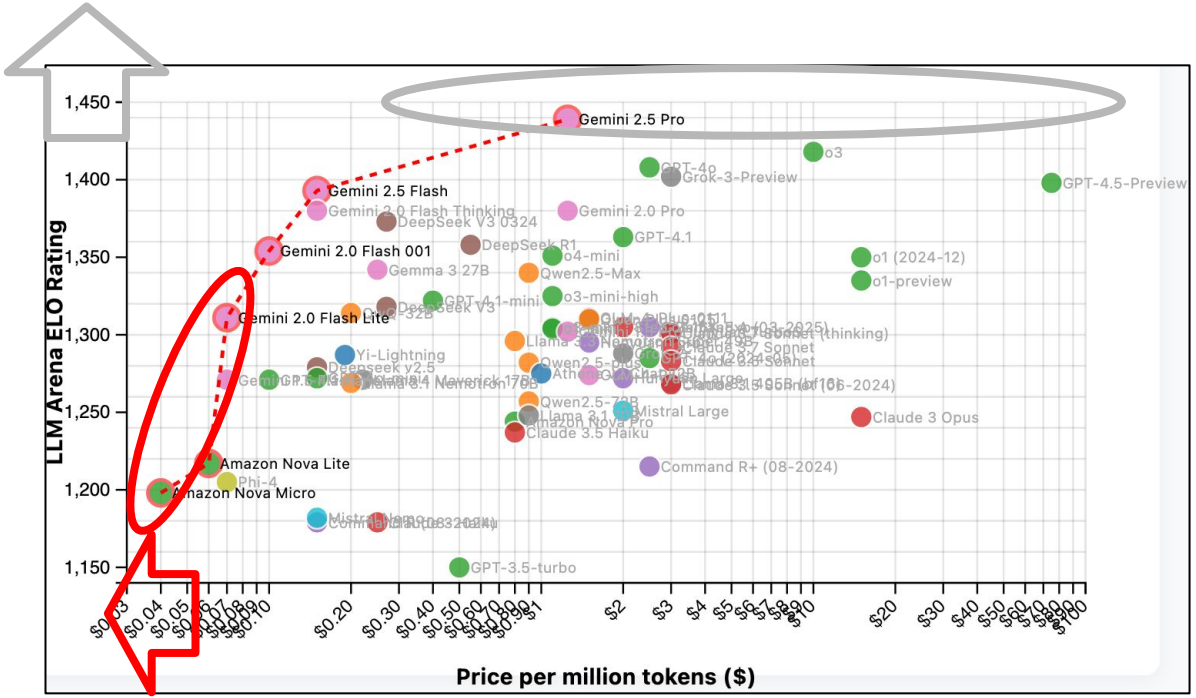
Until now. Focus on best performance.



"*LLM Arena Pareto Frontier: Performance vs Cost (April 2025)*", *screenshot taken on November 18th, 2025.*

Interest in pushing through new frontiers

More and more? Focus on the best quality/cost trade-off.



"LLM Arena Pareto Frontier: Performance vs Cost (April 2025)", screenshot taken on November 18th, 2025.

Hardware optimization

Observation. Current GPUs optimize for matrix-vector/matrix-matrix operations.

"Analog in-memory computing attention mechanism for fast and energy-efficient large language models", Leroux et al., 2025.

Hardware optimization

Observation. Current GPUs optimize for matrix-vector/matrix-matrix operations.

Challenges. Transformer blocks have special needs:

- Attention requires frequent KV reads/writes
- Memory movement dominates cost on GPUs

"Analog in-memory computing attention mechanism for fast and energy-efficient large language models", Leroux et al., 2025.

Hardware optimization

Observation. Current GPUs optimize for matrix-vector/matrix-matrix operations.

Challenges. Transformer blocks have special needs:

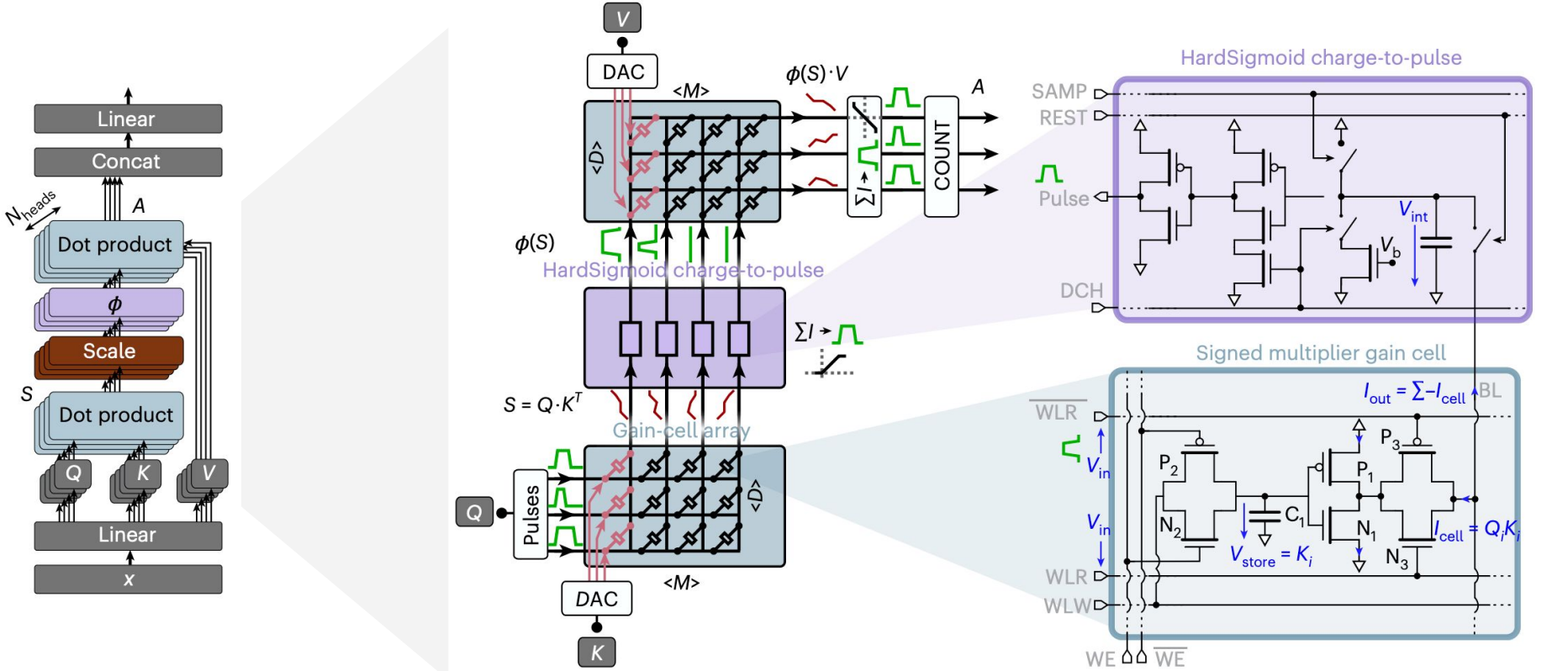
- Attention requires frequent KV reads/writes
- Memory movement dominates cost on GPUs

Idea. "Natively" support attention operations in hardware:

- Store KV cache in dedicated cells
- Use analog signals to model computations

"Analog in-memory computing attention mechanism for fast and energy-efficient large language models", Leroux et al., 2025.

Hardware optimization



Figures adapted from "Analog in-memory computing attention mechanism for fast and energy-efficient large language models", Leroux et al., 2025.

Hardware optimization

Observation. Current GPUs optimize for matrix-vector/matrix-matrix operations.

Challenges. Transformer blocks have special needs:

- Attention requires frequent KV reads/writes
- Memory movement dominates cost on GPUs

Idea. "Natively" support attention operations in hardware:

- Store KV cache in dedicated cells
- Use analog signals to model computations

Results. Up to ~100x latency/~70,000x energy savings with respect to Nvidia's H100.

"Analog in-memory computing attention mechanism for fast and energy-efficient large language models", Leroux et al., 2025.

Daily life has already dramatically changed by LLMs

Today. Many uses already.

Some key applications:

- ✓ Coding ("true" coding, text-to-query)
- ✓ General conversational assistant (common facts, web search)
- ✓ Creativity
- ✓ Learning
- ...

More use cases to be expected

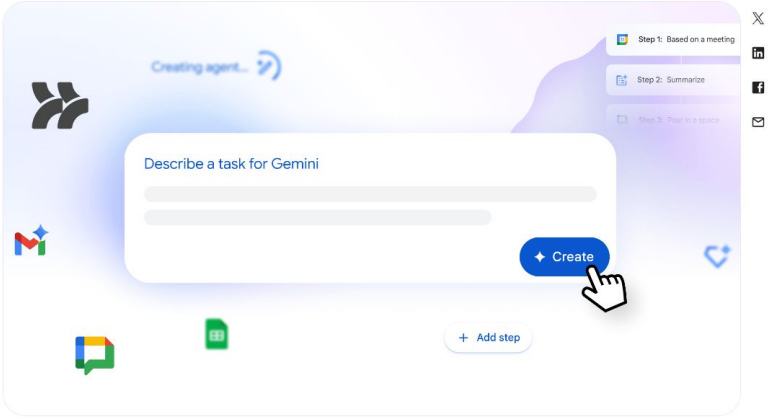
Tomorrow.

Democratization of existing agents (e.g. Google Workspace launch)

Product Announcements

Introducing Google Workspace Studio: Automate everyday work with AI agents

December 3, 2025



"*Introducing Google Workspace Studio: Automate everyday work with AI agents*", Google Workspace, December 3, 2025.

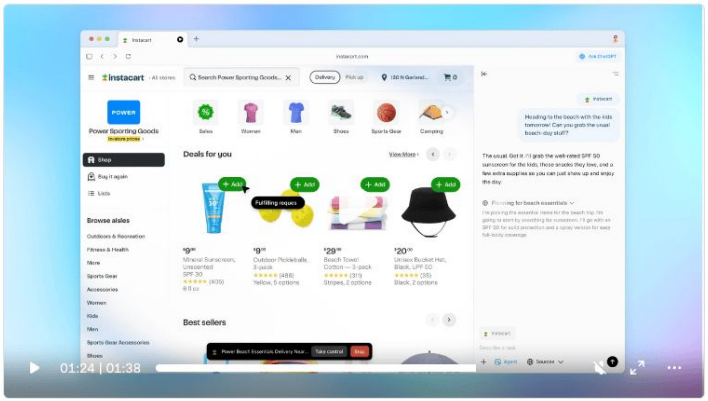
More use cases to be expected

Tomorrow.

Democratization of existing agents (e.g. Google Workspace launch)

Near term.

Browser? (e.g. OpenAI's Atlas launch). Looking further: OS-level LLM?



"*Introducing ChatGPT Atlas*", OpenAI, October 21, 2025.

More use cases to be expected

Tomorrow.

Democratization of existing agents (e.g. Google Workspace launch)

Near term.

Browser? (e.g. OpenAI's Atlas launch). Looking further: OS-level LLM?

Long term.

"Truly" autonomous agents with vast responsibilities?

More use cases to be expected

Tomorrow.

Democratization of existing agents (e.g. Google Workspace launch)

Near term.

Browser? (e.g. OpenAI's Atlas launch). Looking further: OS-level LLM?

Long term.

"Truly" autonomous agents with vast responsibilities?

Impossible?

Actually useful customer service (finally?)

Ongoing challenges

Areas of improvement. Top of mind:

- Fixed weights, not updated (≠ continuous learning)
- "Hallucinations"
- Personalization
- Interpretability
- Safety
- ...

How to stay up-to-date with NLP advances

Papers

- arXiv > Computer Science > Computation and Language
- General ML (NeurIPS, ICML, ICLR) and NLP (ACL, EMNLP) venues

How to stay up-to-date with NLP advances

Papers

- arXiv > Computer Science > Computation and Language
- General ML (NeurIPS, ICML, ICLR) and NLP (ACL, EMNLP) venues

Code

- Authors' GitHub repositories linked in their papers
- Hugging Face's "trending papers": browse state-of-the-art datasets/methods

How to stay up-to-date with NLP advances

Papers

- arXiv > Computer Science > Computation and Language
- General ML (NeurIPS, ICML, ICLR) and NLP (ACL, EMNLP) venues

Code

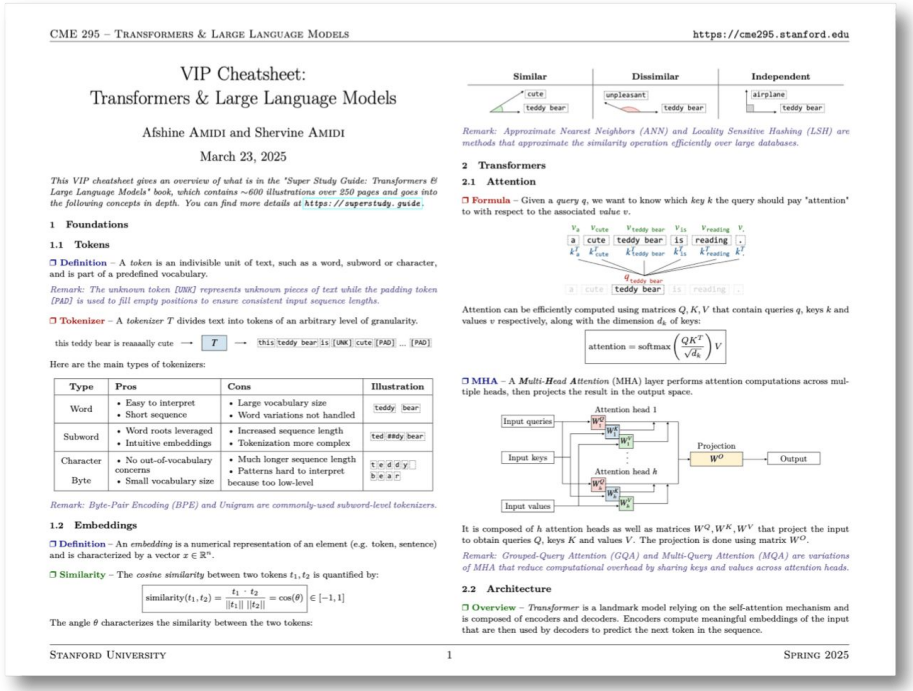
- Authors' GitHub repositories linked in their papers
- Hugging Face's "trending papers": browse state-of-the-art datasets/methods

Miscellaneous

- Twitter (academics + industry leaders)
- YouTube theoretical (Two Minute Papers, Yannic Kilcher) and practical (Google Developers, HuggingFace, Andrej Karpathy)
- Company/academia technical papers + blogs (Amazon Science, Anthropic, Apple ML, Google DeepMind, Meta AI, Microsoft AI, OpenAI, Stanford NLP, ...)

Maybe useful for after the class

VIP Cheatsheet



CME 295 – Transformers & Large Language Models — https://cme295.stanford.edu

VIP Cheatsheet:
Transformers & Large Language Models

Afshine Amidi and Shervine Amidi

March 23, 2025

This VIP cheatsheet gives an overview of what is in the "Super Study Guide: Transformers & Large Language Models" book, which contains ~600 illustrations over 250 pages and goes into the following concepts in depth. You can find more details at `https://superstudy.guide`.

1 Foundations

1.1 Tokens

❒ **Definition** – A token is an indivisible unit of text, such as a word, subword or character, and is part of a predefined vocabulary.

Remark: The unknown token [UNK] represents unknown pieces of text while the padding token [PAD] is used to fill empty positions to ensure consistent input sequence lengths.

❒ **Tokenizer** – A *tokenizer* T divides text into tokens of an arbitrary level of granularity.

this teddy bear is reaaaally cute ⟶ T ⟶ `this` `teddy bear` `is` `[UNK]` `cute` `[PAD]` .. `[PAD]`

Here are the main types of tokenizers:

Type	Pros	Cons	Illustration
Word	• Easy to interpret • Short sequence	• Large vocabulary size • Word variations not handled	`teddy` `bear`
Subword	• Word roots leveraged • Intuitive embeddings	• Increased sequence length • Tokenization more complex	`ted` `##dy` `bear`
Character Byte	• No out-of-vocabulary concerns • Small vocabulary size	• Much longer sequence length • Patterns hard to interpret because too low-level	`t` `e` `d` `d` `y` `b` `e` `a` `r`

Remark: Byte-Pair Encoding (BPE) and Unigram are commonly-used subword-level tokenizers.

1.2 Embeddings

❒ **Definition** – An *embedding* is a numerical representation of an element (e.g. token, sentence) and is characterized by a vector $x \in \mathbb{R}^n$.

❒ **Similarity** – The *cosine similarity* between two tokens t_1, t_2 is quantified by:

$$\text{similarity}(t_1, t_2) = \frac{t_1 \cdot t_2}{||t_1|| \, ||t_2||} = \cos(\theta) \in [-1, 1]$$

The angle θ characterizes the similarity between the two tokens:

Similar	Dissimilar	Independent
`cute` `teddy bear`	`unpleasant` `teddy bear`	`airplane` `teddy bear`

Remark: Approximate Nearest Neighbors (ANN) and Locality Sensitive Hashing (LSH) are methods that approximate the similarity operation efficiently over large databases.

2 Transformers

2.1 Attention

❒ **Formula** – Given a *query* q, we want to know which *key* k the query should pay "attention" to with respect to the associated *value* v.

`a` `cute` `teddy bear` `is` `reading` `.`

Attention can be efficiently computed using matrices Q, K, V that contain queries q, keys k and values v respectively, along with the dimension d_k of keys:

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

❒ **MHA** – A *Multi-Head Attention* (MHA) layer performs attention computations across multiple heads, then projects the result in the output space.

It is composed of h attention heads as well as matrices W^Q, W^K, W^V that project the input to obtain queries Q, keys K and values V. The projection is done using matrix W^O.

Remark: Grouped-Query Attention (GQA) and Multi-Query Attention (MQA) are variations of MHA that reduce computational overhead by sharing keys and values across attention heads.

2.2 Architecture

❒ **Overview** – *Transformer* is a landmark model relying on the self-attention mechanism and is composed of encoders and decoders. Encoders compute meaningful embeddings of the input that are then used by decoders to predict the next token in the sequence.

Stanford University — 1 — Spring 2025

https://github.com/afshinea/stanford-cme-295-transformers-large-language-models

Maybe useful for after the class

VIP Cheatsheet

العربية - Čeština - English - Español -

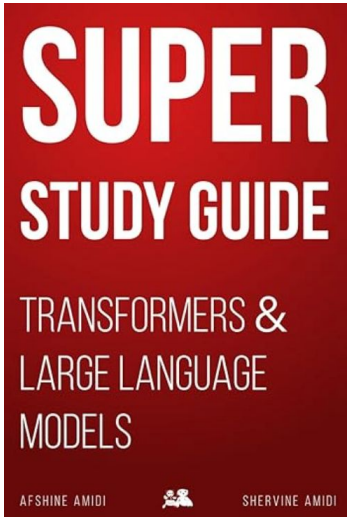
فارسی - Français - Italiano - 日本語 -

한국어 - ไทย - Türkçe - 中文

https://github.com/afshinea/stanford-cme-295-transformers-large-language-models

Thank you for this quarter!

Afshine **Shervine**



Super Study Guide:
Transformers &
Large Language Models

Book: https://superstudy.guide

Wishing you all the best!

Feel free to reach out if you want to chat!