

CME 295: Transformers & Large Language Models



Afshine Amidi & Shervine Amidi





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

A cute teddy bear is reading.

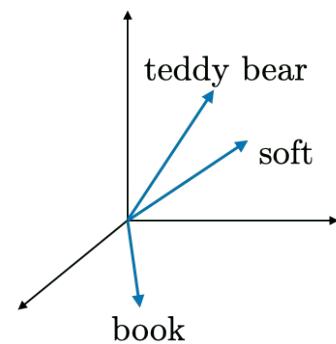
a cute teddy bear is reading .

Rewinding the quarter...

Lecture 1

A cute teddy bear is reading.

a cute teddy bear is reading .



Rewinding the quarter...

Lecture 1

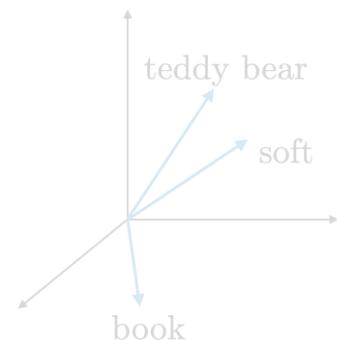
A cute teddy bear is reading.

a cute teddy bear is reading .

v_a v_{cute} $v_{\text{teddy bear}}$ v_{is} v_{reading} $v.$
a cute teddy bear is reading .
 k_a^T k_{cute}^T $k_{\text{teddy bear}}^T$ k_{is}^T k_{reading}^T $k^T.$

$q_{\text{teddy bear}}$

a cute teddy bear is reading .



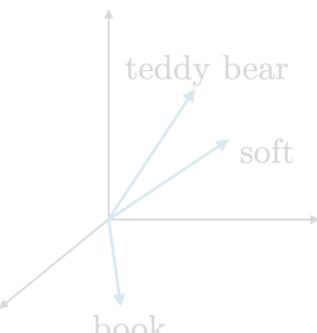
Rewinding the quarter...

Lecture 1

A cute teddy bear is reading.

a cute teddy bear is reading .

v_a v_{cute} $v_{\text{teddy bear}}$ v_{is} v_{reading} v_{\cdot}
[a] [cute] [teddy bear] [is] [reading] [.]
 k_a^T k_{cute}^T $k_{\text{teddy bear}}^T$ k_{is}^T k_{reading}^T k_{\cdot}^T



$q_{\text{teddy bear}}$

[a] [cute] [teddy bear] [is] [reading] [.]

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

Rewinding the quarter...

Lecture 1

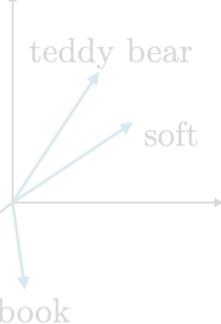
A cute teddy bear is reading.

a cute teddy bear is reading .

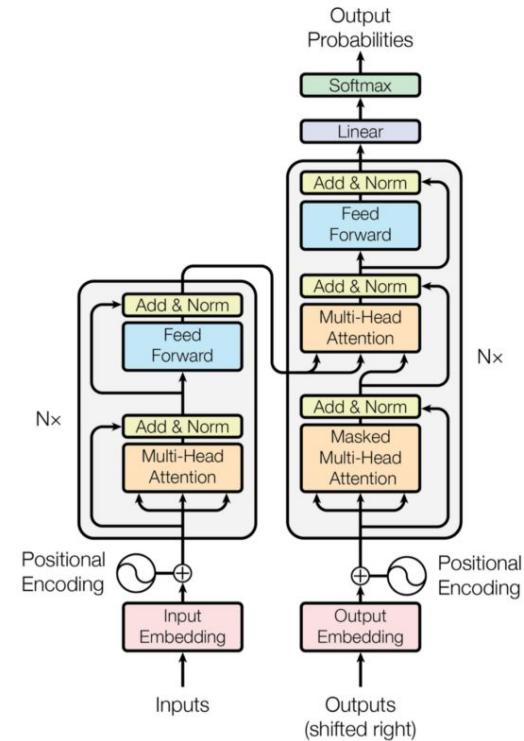
v_a v_{cute} $v_{\text{teddy bear}}$ v_{is} v_{reading} v_{\cdot}
a cute teddy bear is reading .
 k_a^T k_{cute}^T $k_{\text{teddy bear}}^T$ k_{is}^T k_{reading}^T k_{\cdot}^T

$q_{\text{teddy bear}}$

a cute teddy bear is reading .



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



Rewinding the quarter...

Lecture 1

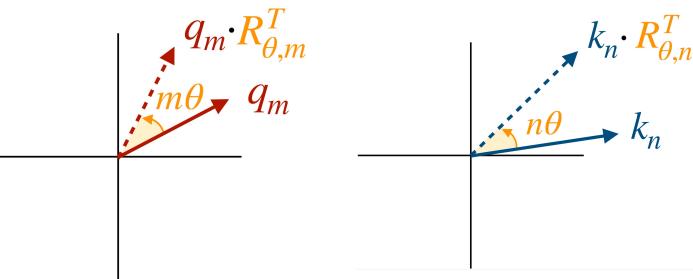
Lecture 2

Transformer-based models & tricks

Rewinding the quarter...

Lecture 1

Lecture 2

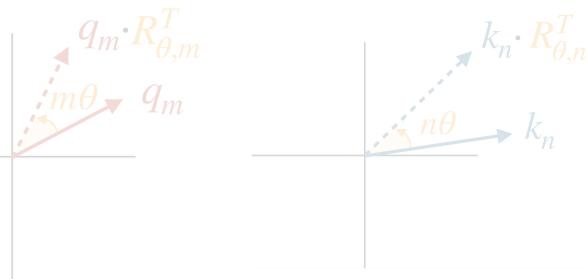


$$R_{\theta,m} = \begin{pmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{pmatrix}$$

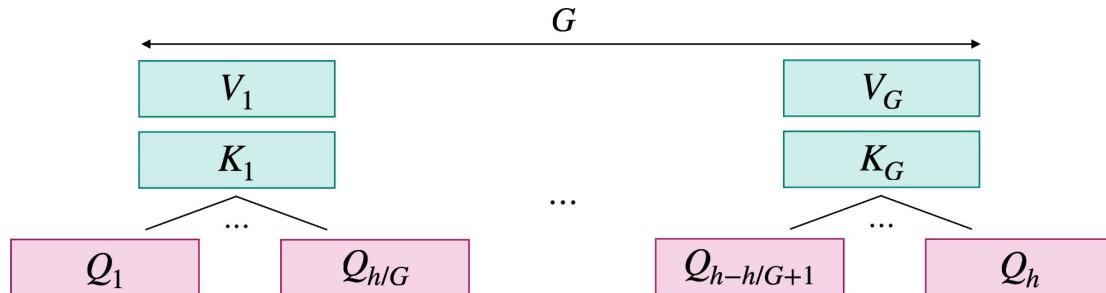
Rewinding the quarter...

Lecture 1

Lecture 2



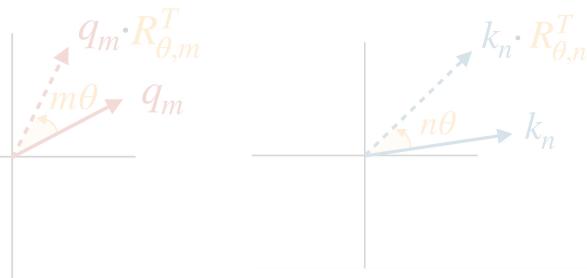
$$R_{\theta,m} = \begin{pmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{pmatrix}$$



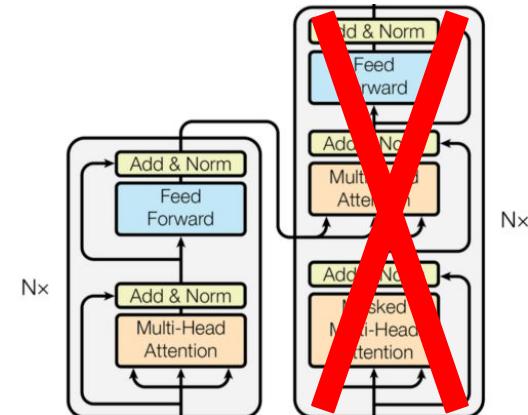
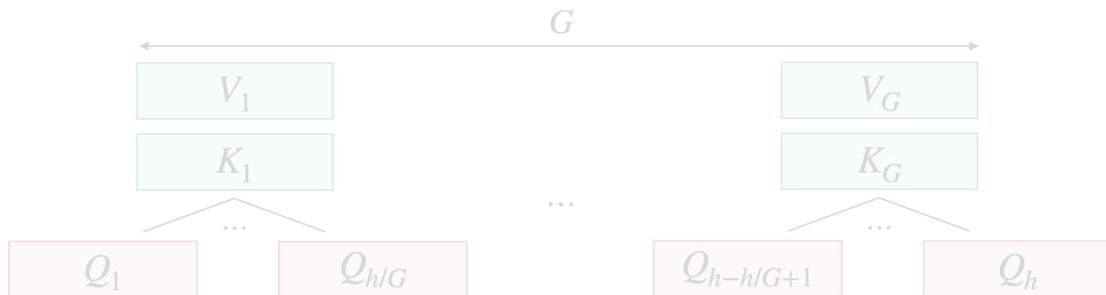
Rewinding the quarter...

Lecture 1

Lecture 2



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



Rewinding the quarter...

Lecture 1

Lecture 2

Lecture 3

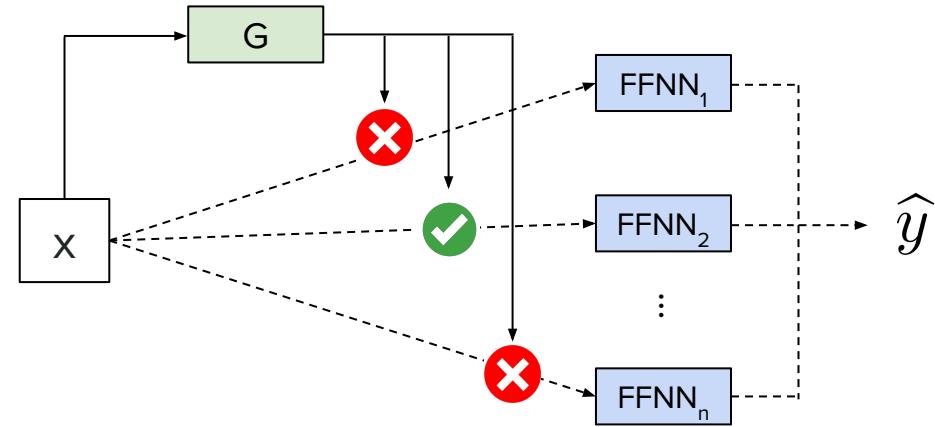
Large Language Models

Rewinding the quarter...

Lecture 1

Lecture 2

Lecture 3



Rewinding the quarter...

Lecture 1

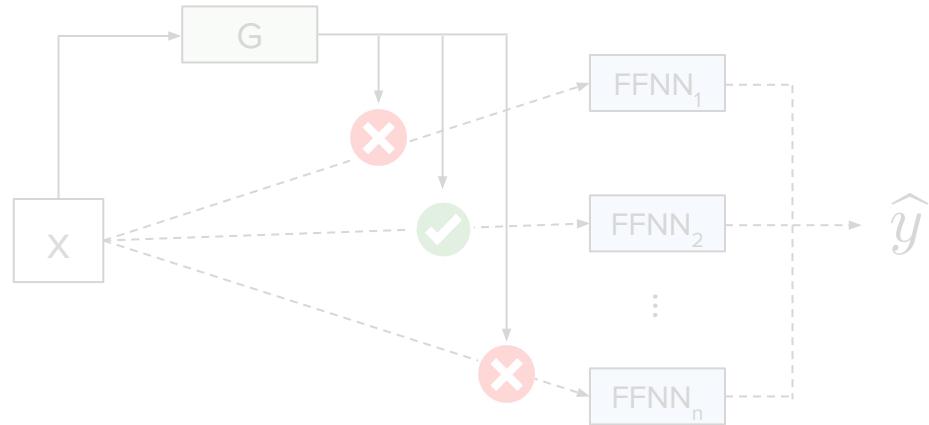
Lecture 2

Lecture 3

Layer 0

```
class MoeLayer(nn.Module):
    def __init__(self, experts: List[nn.Module], gate: nn.Module, moe_args: Dict):
        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.shape[-1])
        gate_logits = self.gate(inputs_squashed)
        weights, selected_experts = torch.topk(
            gate_logits, self.args.numExpertsPerBatch)
        weights = nn.functional.softmax(
            weights,
            dim=1,
            dtype=torch.float,
        ).type_as(inputs)
        results = torch.zeros_like(inputs_squashed)
        for i, expert in enumerate(self.experts):
            batch_idx, nthExpert = torch.where(selected_experts == i)
            results[batch_idx] += weights[batch_idx] * expert(inputs_squashed[batch_idx])
        return results.view_as(inputs)
```



Rewinding the quarter...

Lecture 1

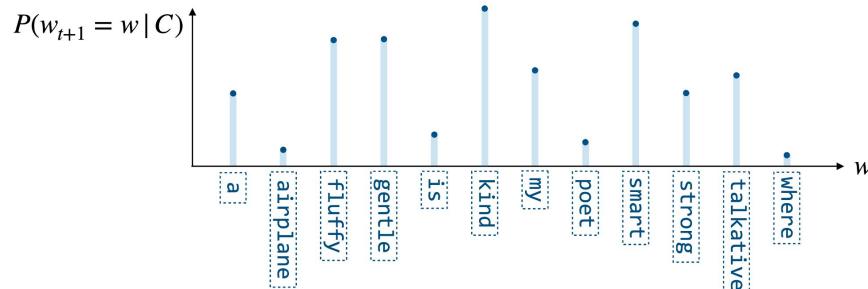
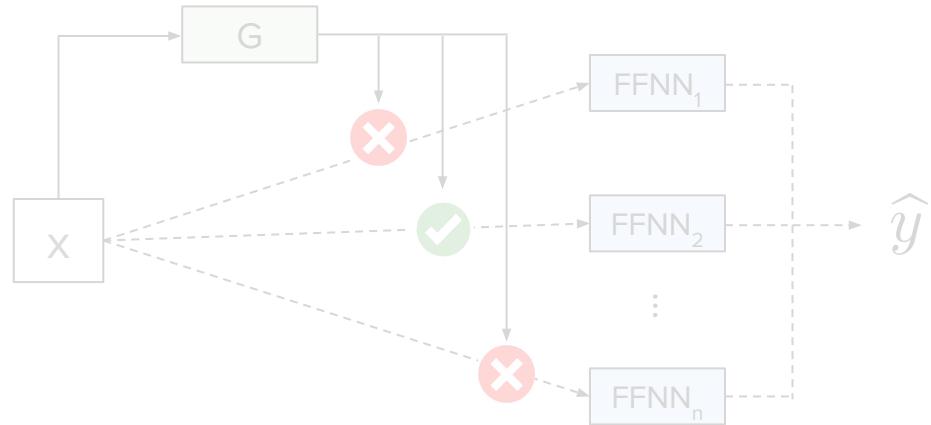
Lecture 2

Lecture 3

Layer 0

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

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



Rewinding the quarter...

Lecture 1

Lecture 2

Lecture 3

Lecture 4

LLM training

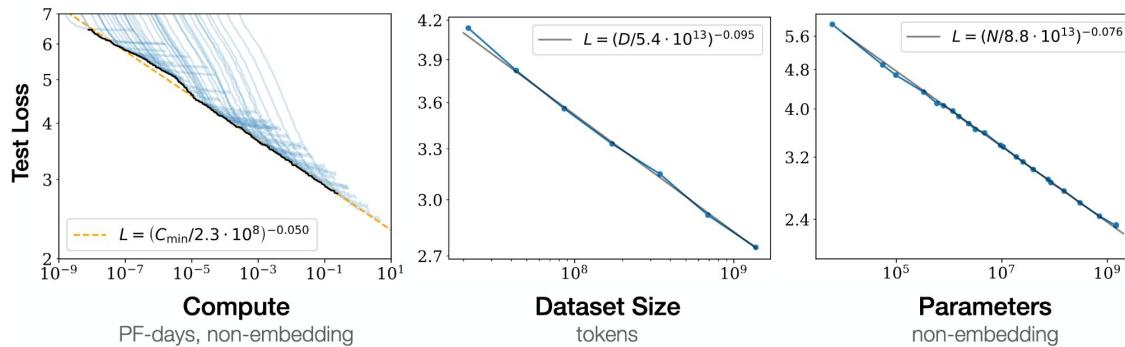
Rewinding the quarter...

Lecture 1

Lecture 2

Lecture 3

Lecture 4



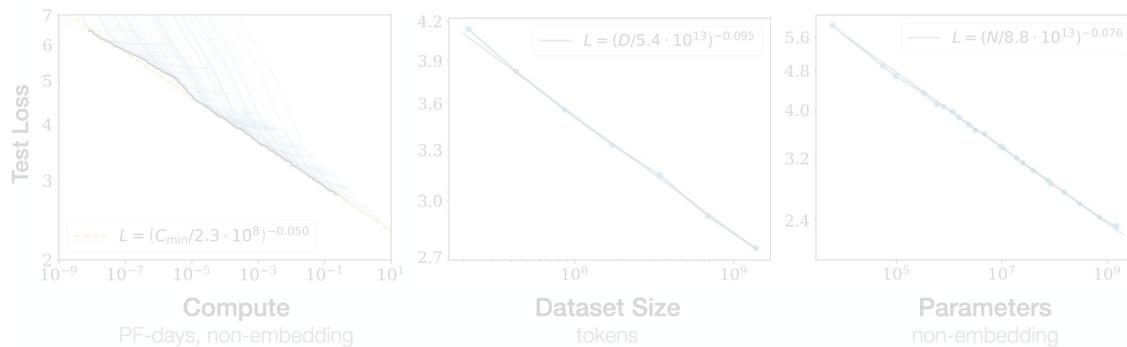
Rewinding the quarter...

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

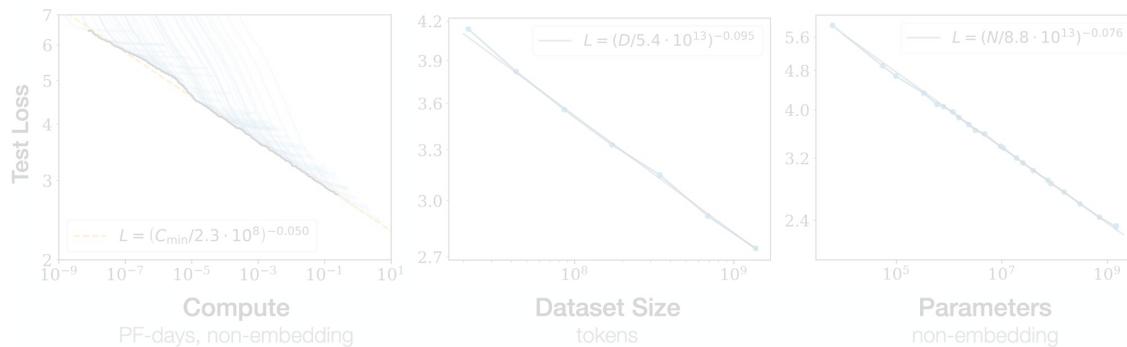
Rewinding the quarter...

Lecture 1

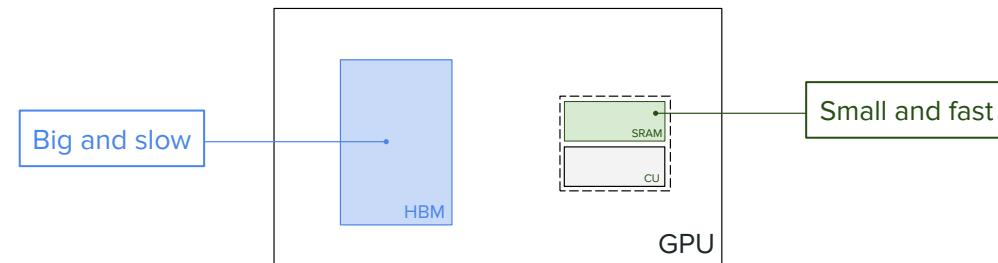
Lecture 2

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



Rewinding the quarter...

Lecture 1

Lecture 2

Lecture 3

Lecture 4

Initialized model

Rewinding the quarter...

Lecture 1

Lecture 2

Lecture 3

Lecture 4

Initialized model



Pretraining



Model with "basic
knowledge" about
language, code, etc.

Rewinding the quarter...

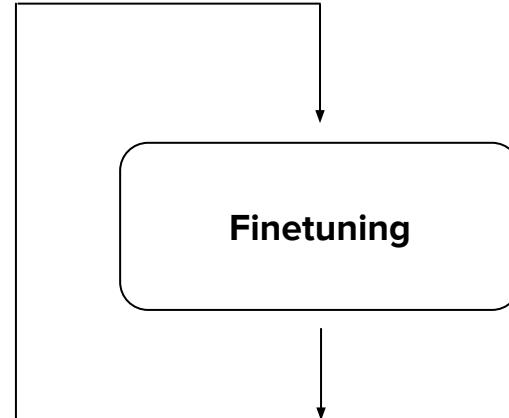
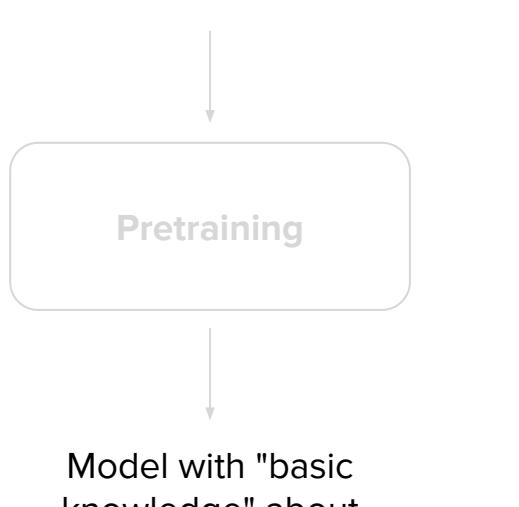
Lecture 1

Lecture 2

Lecture 3

Lecture 4

Initialized model



Rewinding the quarter...

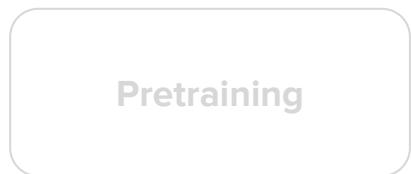
Lecture 1

Lecture 2

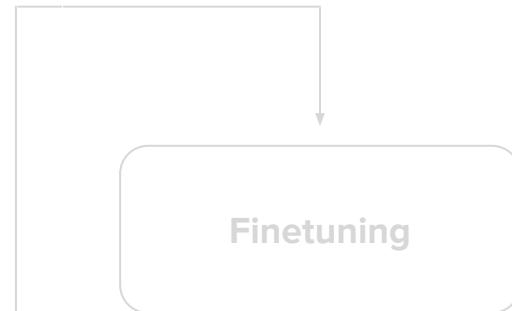
Lecture 3

Lecture 4

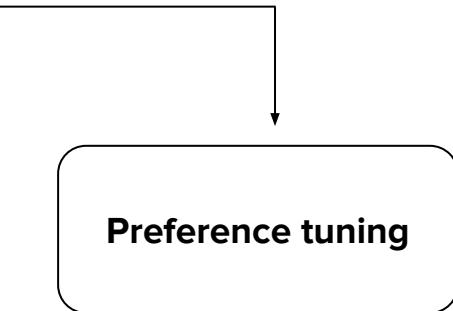
Initialized model



Model with "basic knowledge" about language, code, etc.



Model tuned for specific tasks



Model does not misbehave as much

Rewinding the quarter...

Lecture 1

Lecture 2

Lecture 3

Lecture 4

Lecture 5

LLM tuning

Rewinding the quarter...

Lecture 1

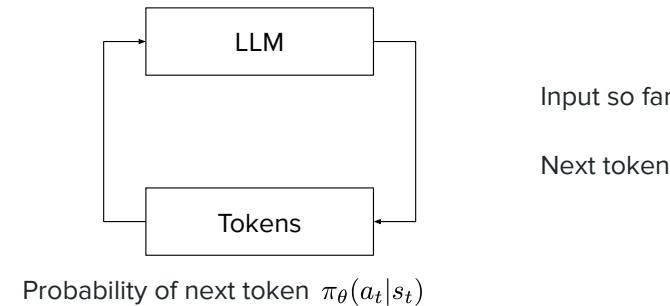
Lecture 2

Lecture 3

Lecture 4

Lecture 5

Human
preference r_t



Rewinding the quarter...

Lecture 1

Lecture 2

Lecture 3

Lecture 4

Lecture 5

Human
preference r_t



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

Rewinding the quarter...

Lecture 1

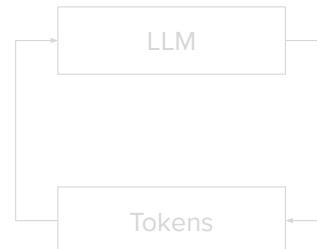
Lecture 2

Lecture 3

Lecture 4

Lecture 5

Human
preference r_t

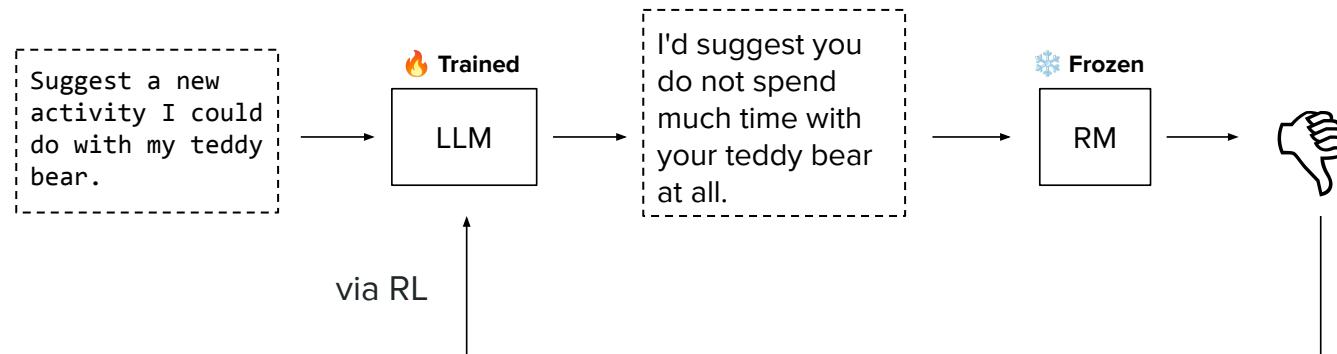


Input so far

Next token

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

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



Rewinding the quarter...

Lecture 1

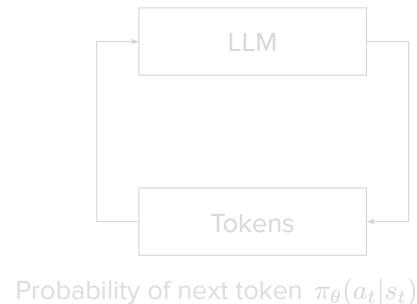
Lecture 2

Lecture 3

Lecture 4

Lecture 5

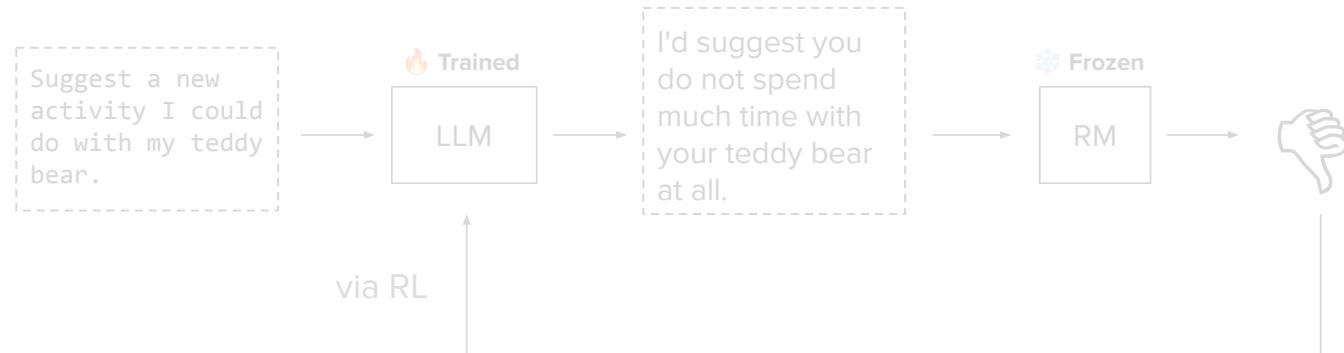
Human
preference r_t



Input so far
Next token

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

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



Rewinding the quarter...

Lecture 1

Lecture 2

Lecture 3

Lecture 4

Lecture 5

Lecture 6

LLM reasoning

Rewinding the quarter...

Lecture 1

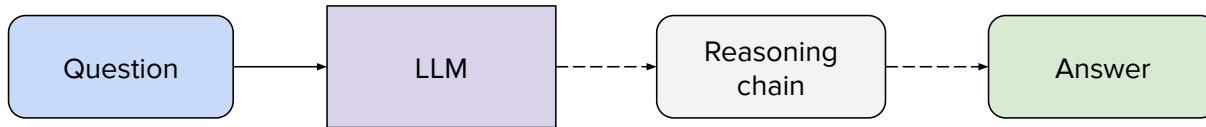
Lecture 2

Lecture 3

Lecture 4

Lecture 5

Lecture 6



Rewinding the quarter...

Lecture 1

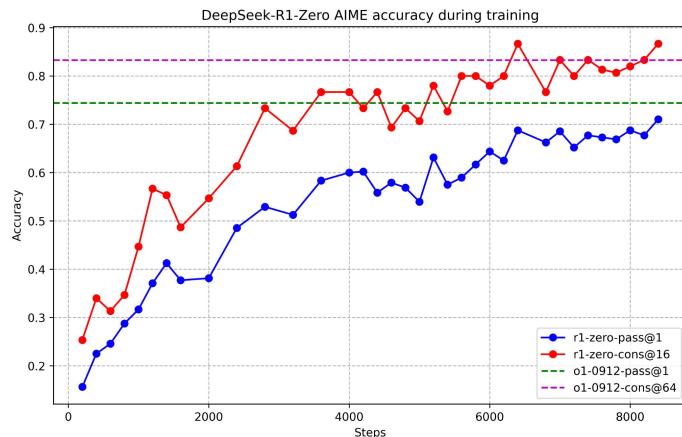
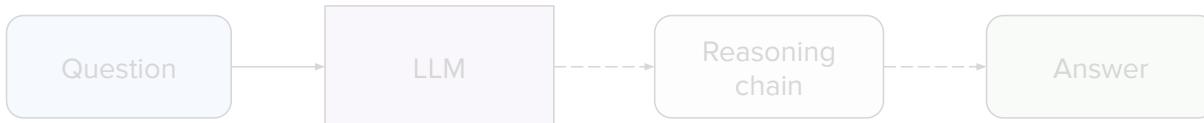
Lecture 2

Lecture 3

Lecture 4

Lecture 5

Lecture 6



Rewinding the quarter...

Lecture 1

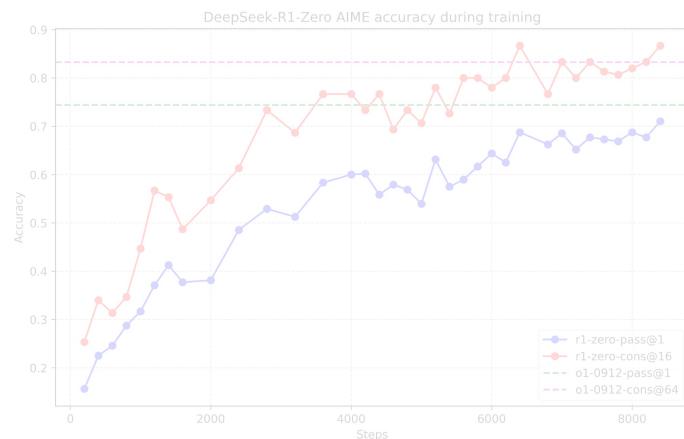
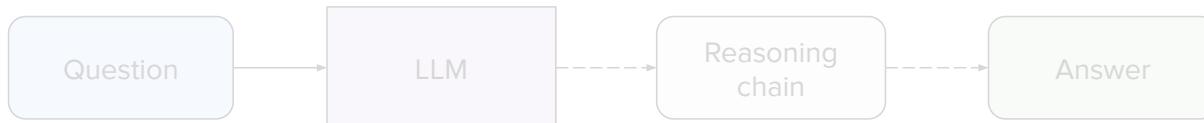
Lecture 2

Lecture 3

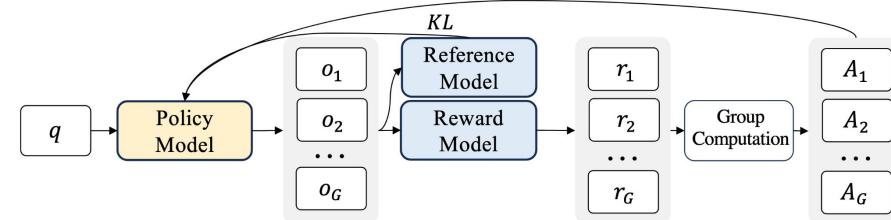
Lecture 4

Lecture 5

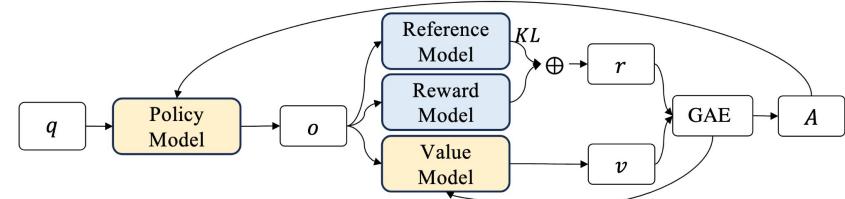
Lecture 6



GRPO



ppo



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

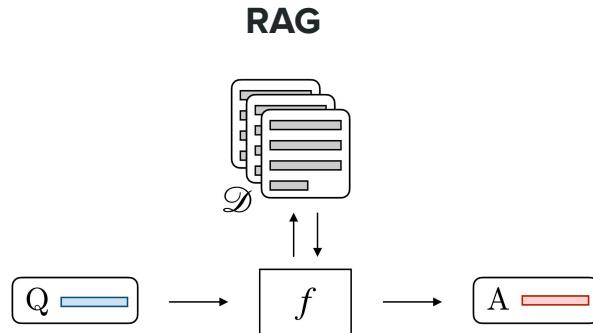
Lecture 3

Lecture 4

Lecture 5

Lecture 6

Lecture 7



Rewinding the quarter...

Lecture 1

Lecture 2

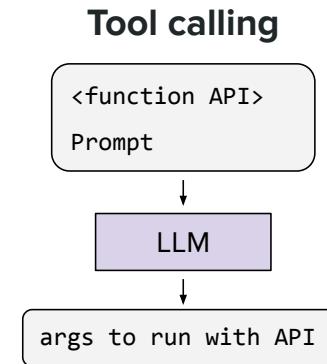
Lecture 3

Lecture 4

Lecture 5

Lecture 6

Lecture 7



Rewinding the quarter...

Lecture 1

Lecture 2

Lecture 3

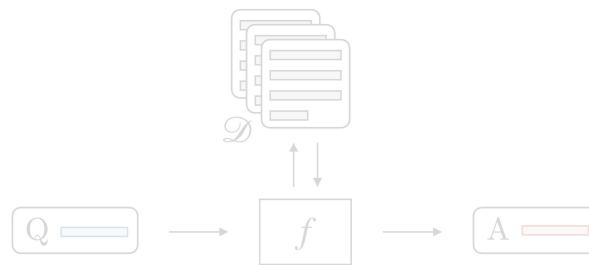
Lecture 4

Lecture 5

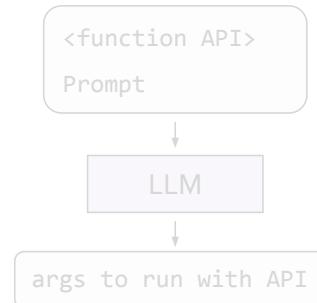
Lecture 6

Lecture 7

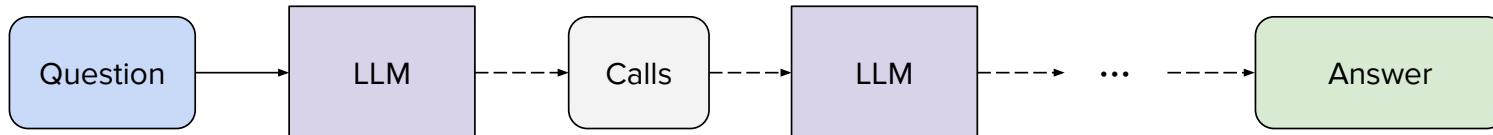
RAG



Tool calling



Agent



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

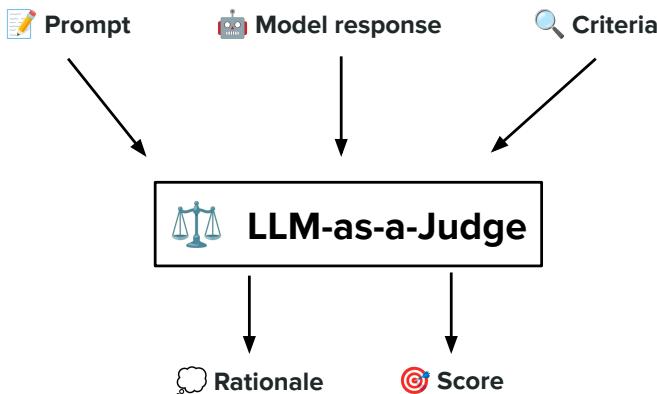
Lecture 4

Lecture 5

Lecture 6

Lecture 7

Lecture 8



Rewinding the quarter...

Lecture 1

Lecture 2

Lecture 3

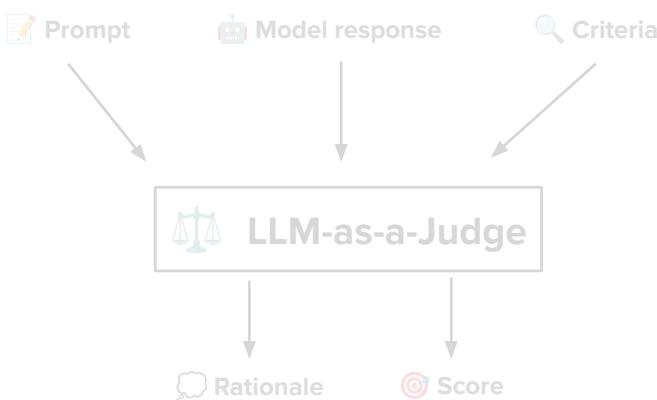
Lecture 4

Lecture 5

Lecture 6

Lecture 7

Lecture 8



Knowledge
<ul style="list-style-type: none">• Accurately state facts about the world• More "breadth" than "depth"• Reflects pretraining quality

MMLU

Reasoning
<ul style="list-style-type: none">• Solve multi-step problems• Includes both "maths" and "common sense" subfields• Internal reasoning

AIME
PIQA

Coding
<ul style="list-style-type: none">• Generate syntactically correct code• Tests for programming proficiency• Proxy for tool use abilities

SWE-bench

Safety
<ul style="list-style-type: none">• Prevent harmful, toxic, inappropriate behavior• Surfaces vulnerabilities before deployment• Alignment with custom preferences

HarmBench



Transformers & Large Language Models

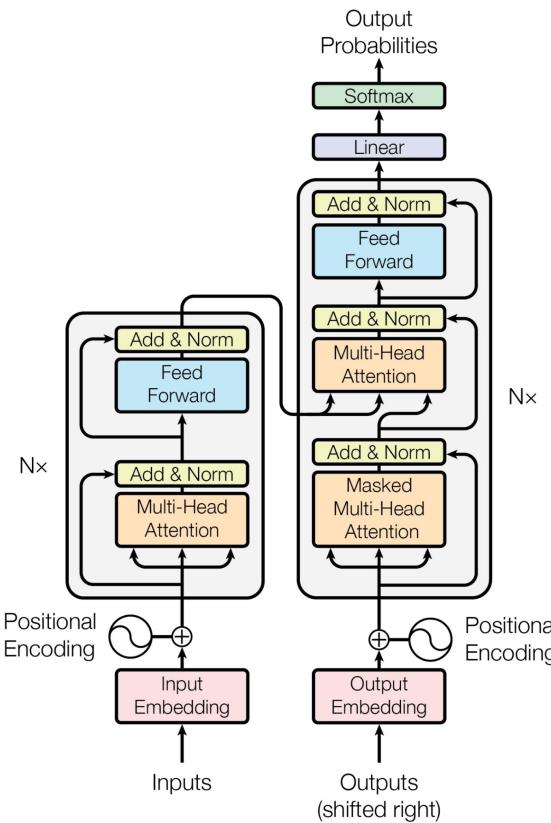
Recap

**Beyond Transformer-based
LLMs**

Diffusion LLMs

Closing thoughts

Can we use Transformers for other things?



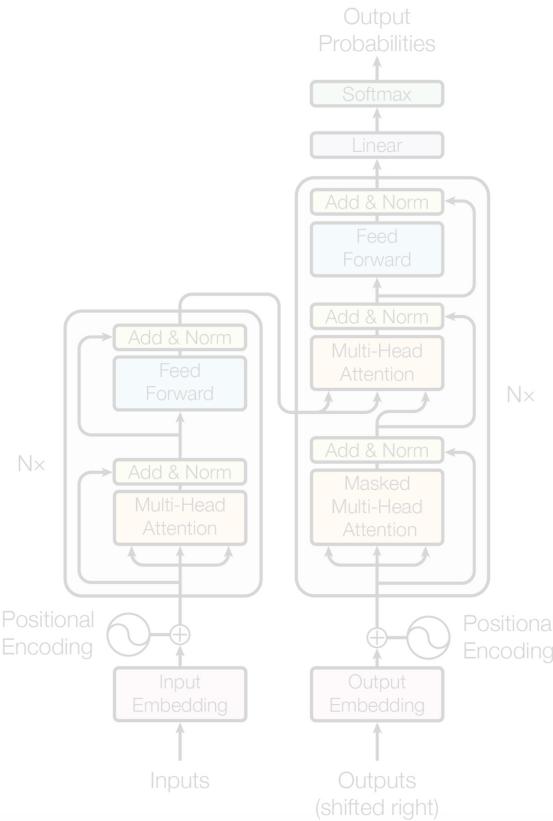
Context

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

Benefits

- Weaker inductive biases
- More generalizability

Can we use Transformers for other things?



Context

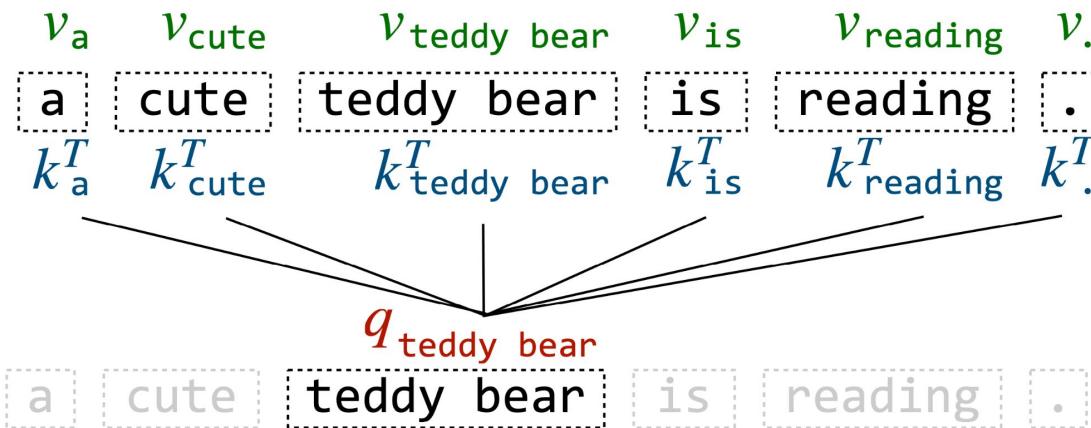
- Introduced in 2017 for machine translation
- Relies on **self-attention**
- Concepts of query, key, value

Benefits

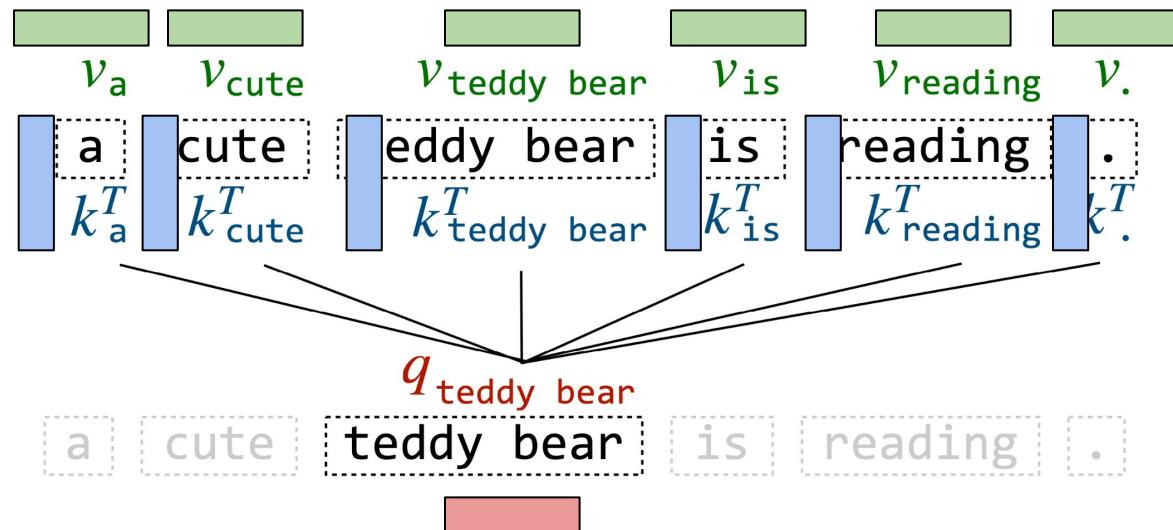
- Weaker inductive biases
- More generalizability

Refresher on attention mechanism

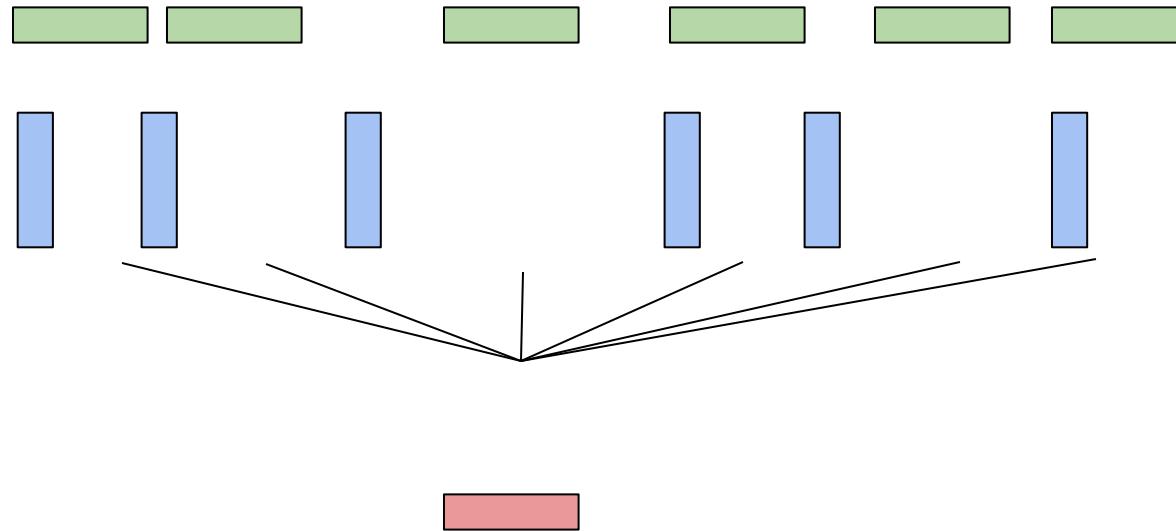
Concept of **Query**, **Key**, **Value**



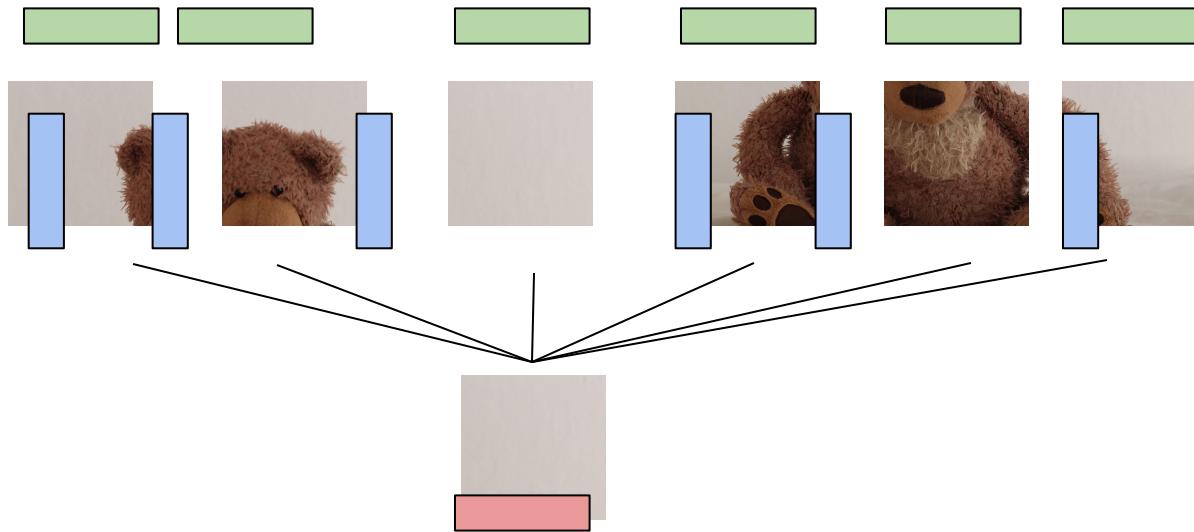
Attention mechanism...



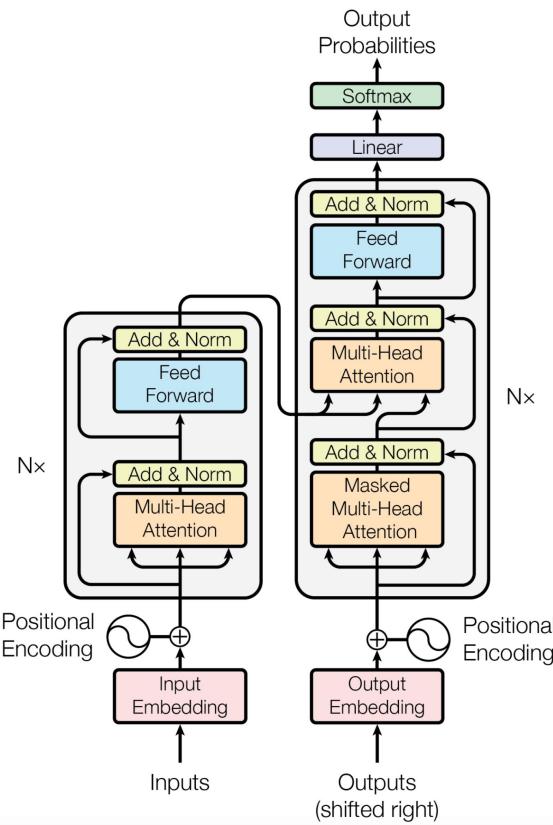
Attention mechanism...



Attention mechanism...for images!



Adapting Transformer architecture for images



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

Adapting Transformer architecture for images

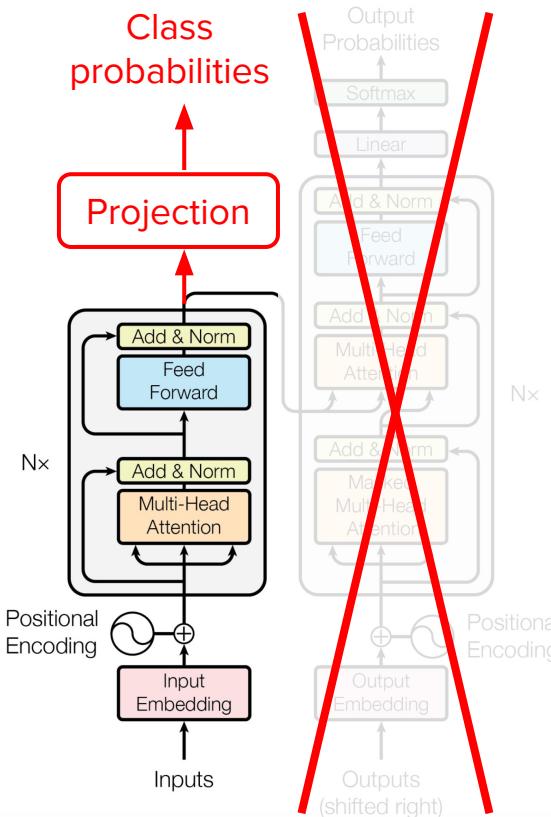
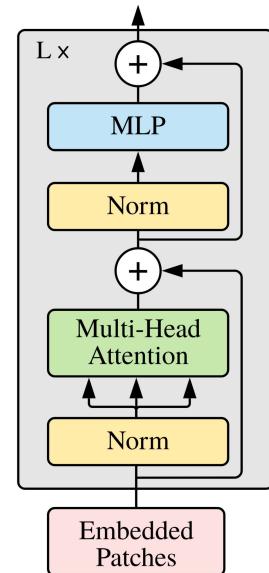
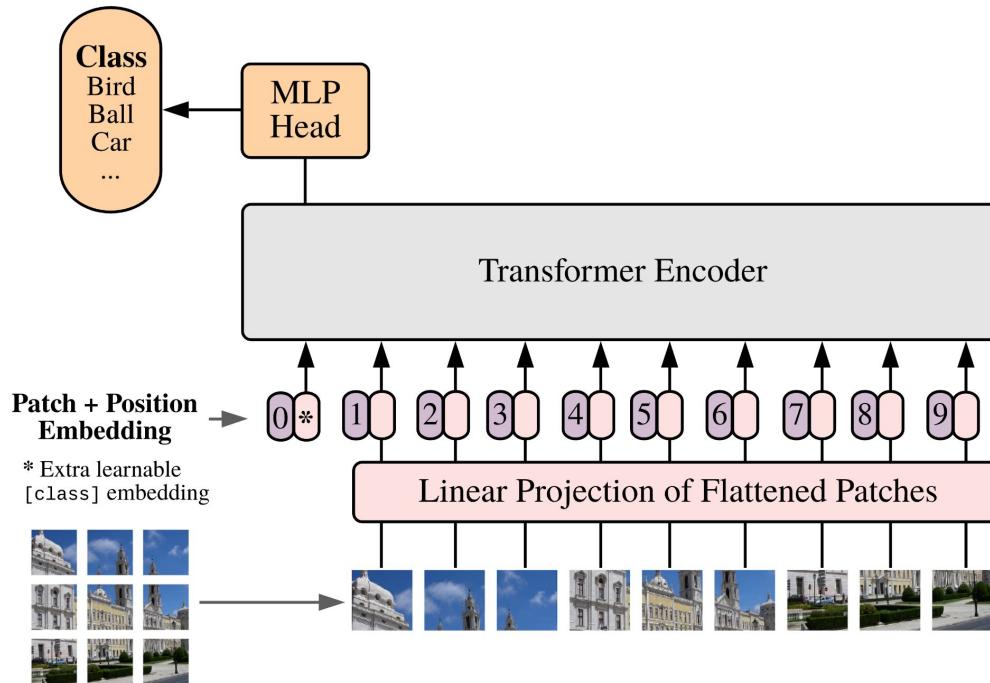


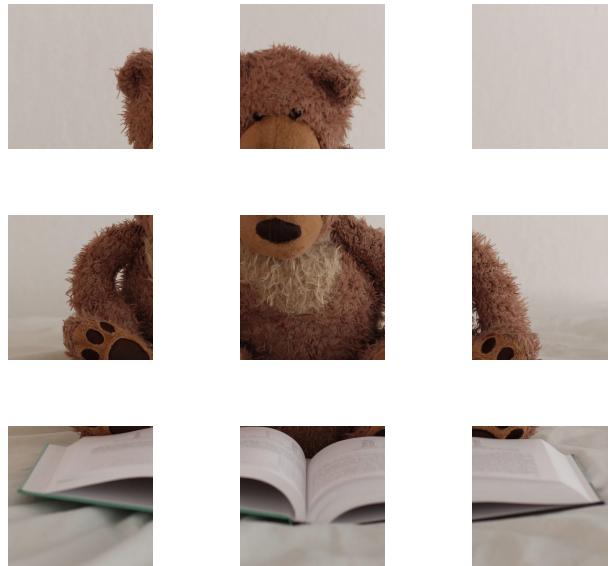
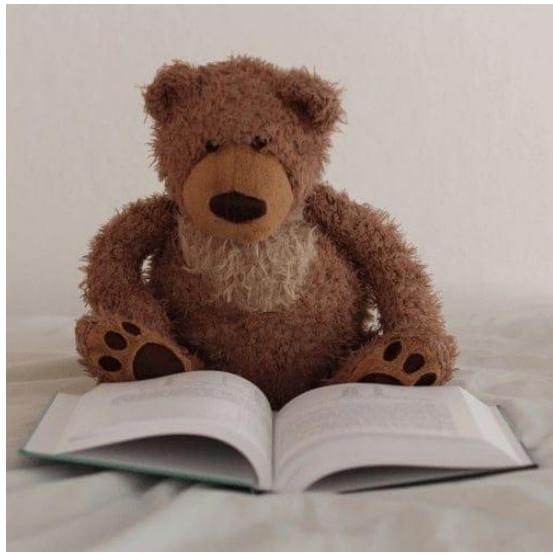
Figure adapted from "[Attention Is All You Need](#)", Vaswani et al., 2017.

Vision Transformer

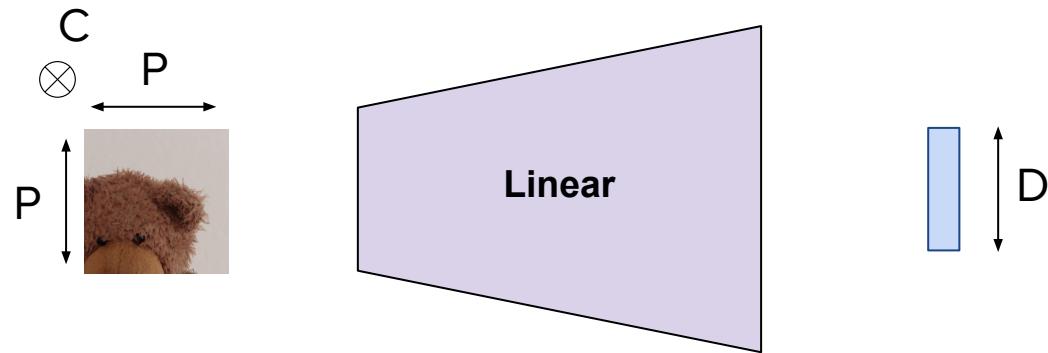
ViT = Vision Transformer



End-to-end example using ViT



End-to-end example using ViT

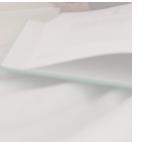


End-to-end example using ViT



End-to-end example using ViT

[CLS]



End-to-end example using ViT



embedding



End-to-end example using ViT



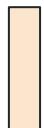
position embedding



embedding



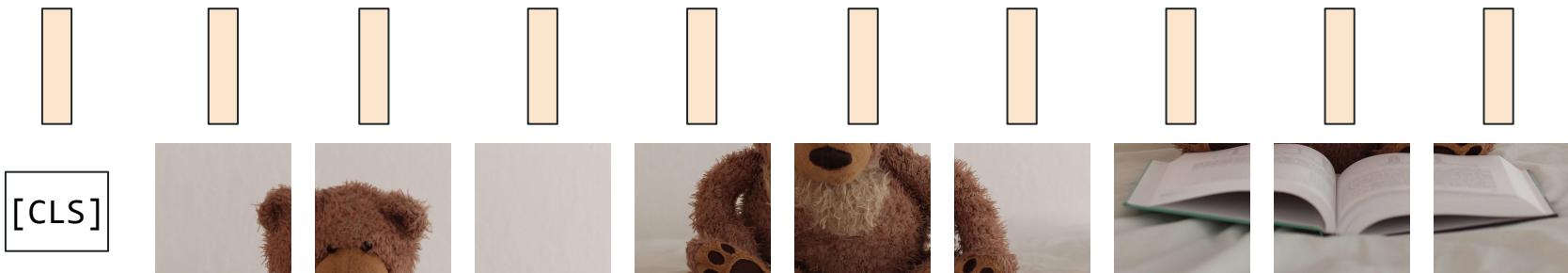
End-to-end example using ViT



position-aware embedding



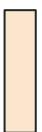
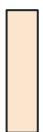
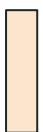
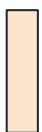
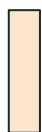
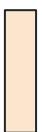
End-to-end example using ViT



End-to-end example using ViT

ViT architecture

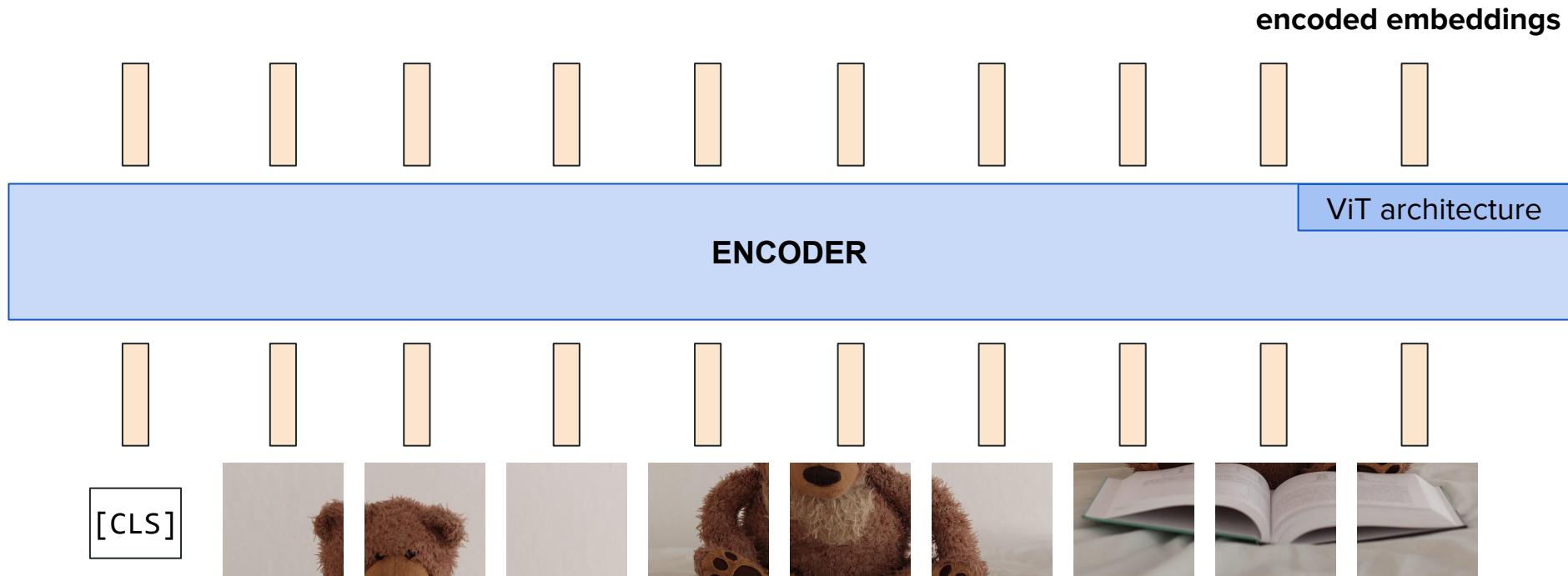
ENCODER



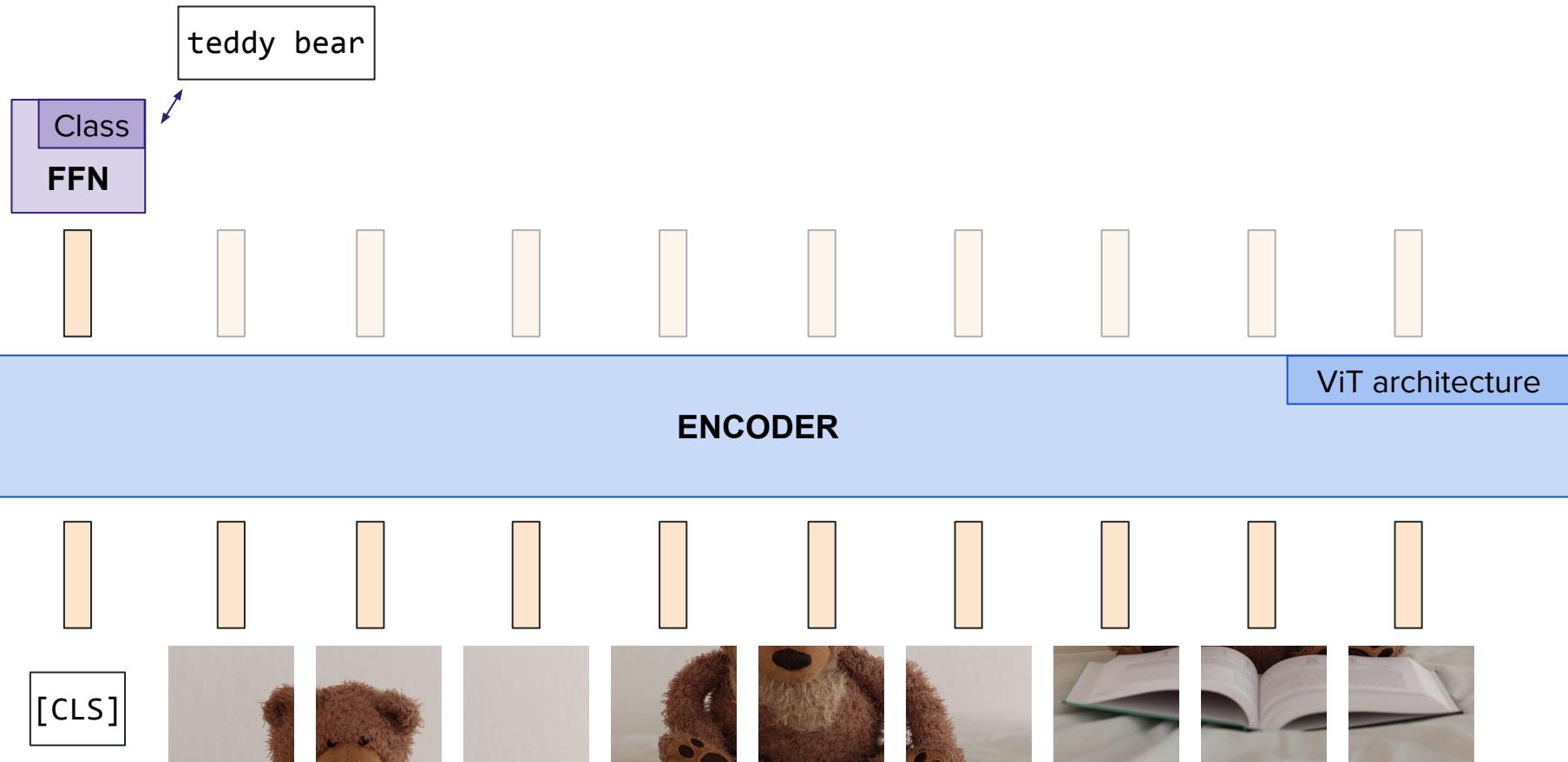
[CLS]



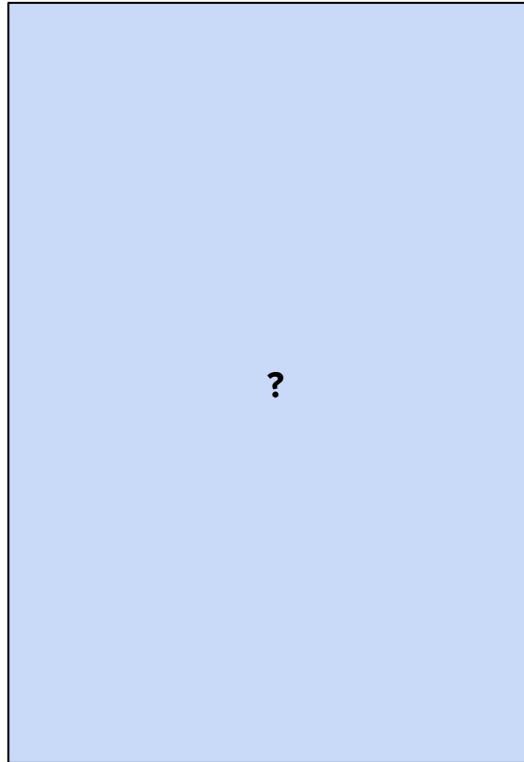
End-to-end example using ViT



End-to-end example using ViT

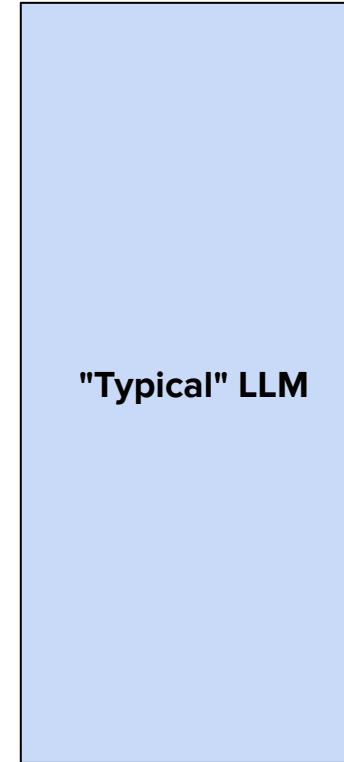


VLM = Vision Language Model



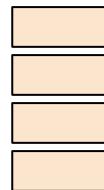
How cute is this teddy bear? →

Method 1: recycle decoder-only architecture

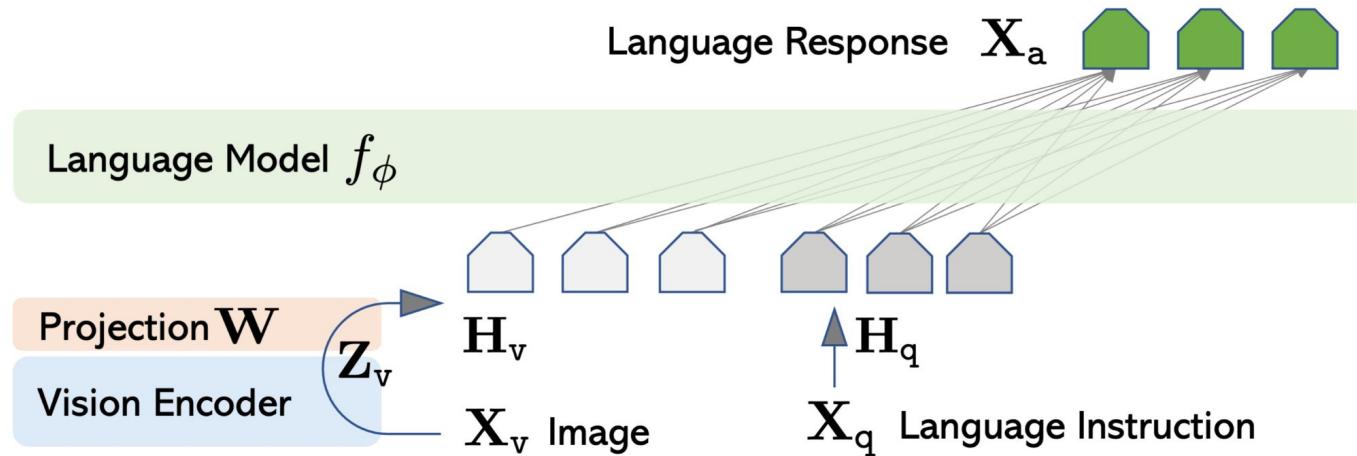


Very cute!

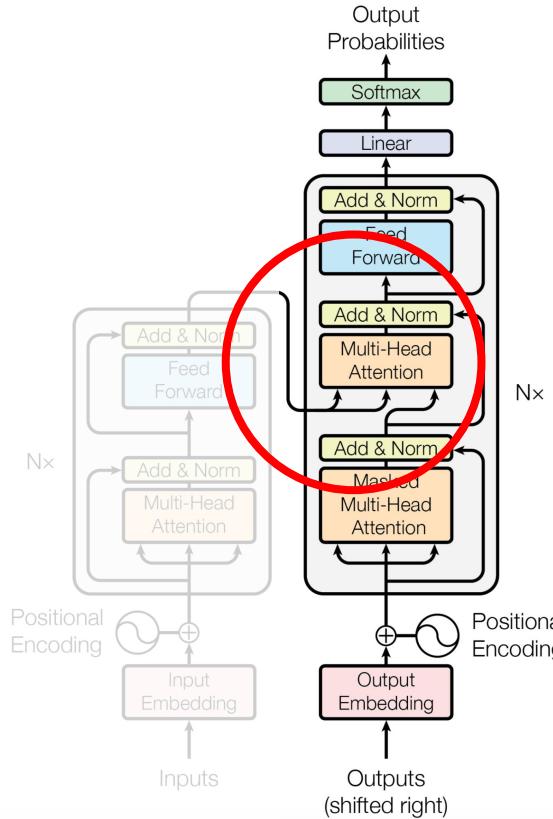
How cute is this teddy bear?



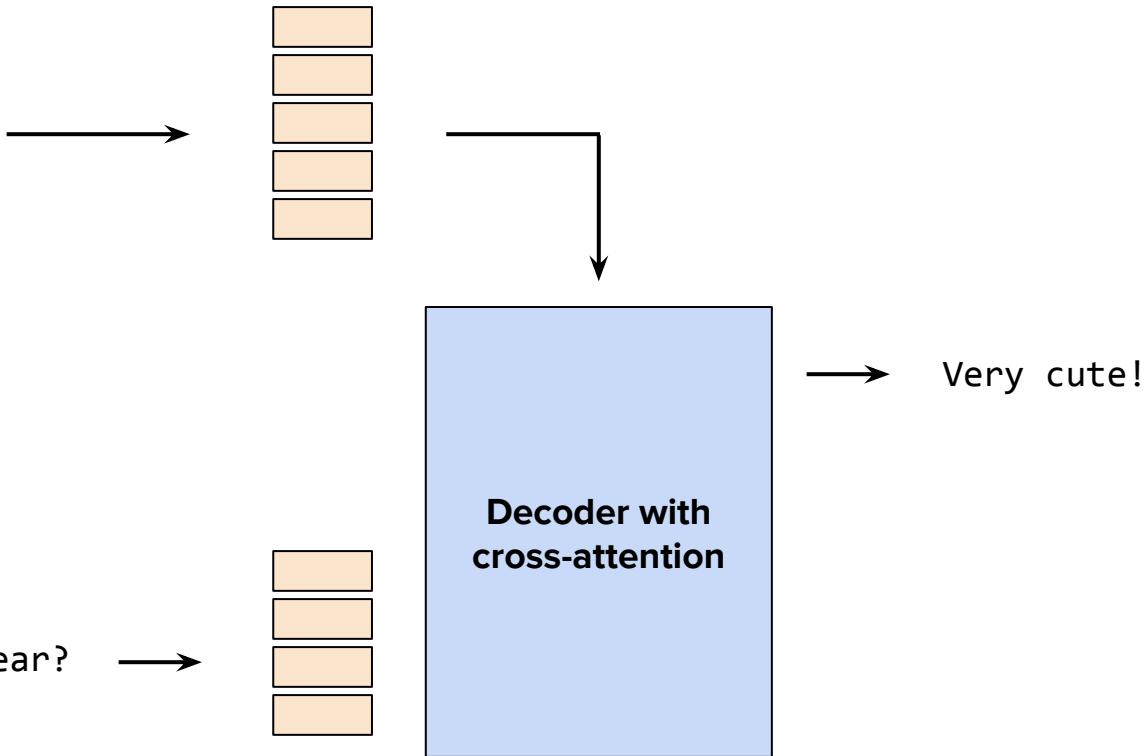
Method 1: recycle decoder-only architecture



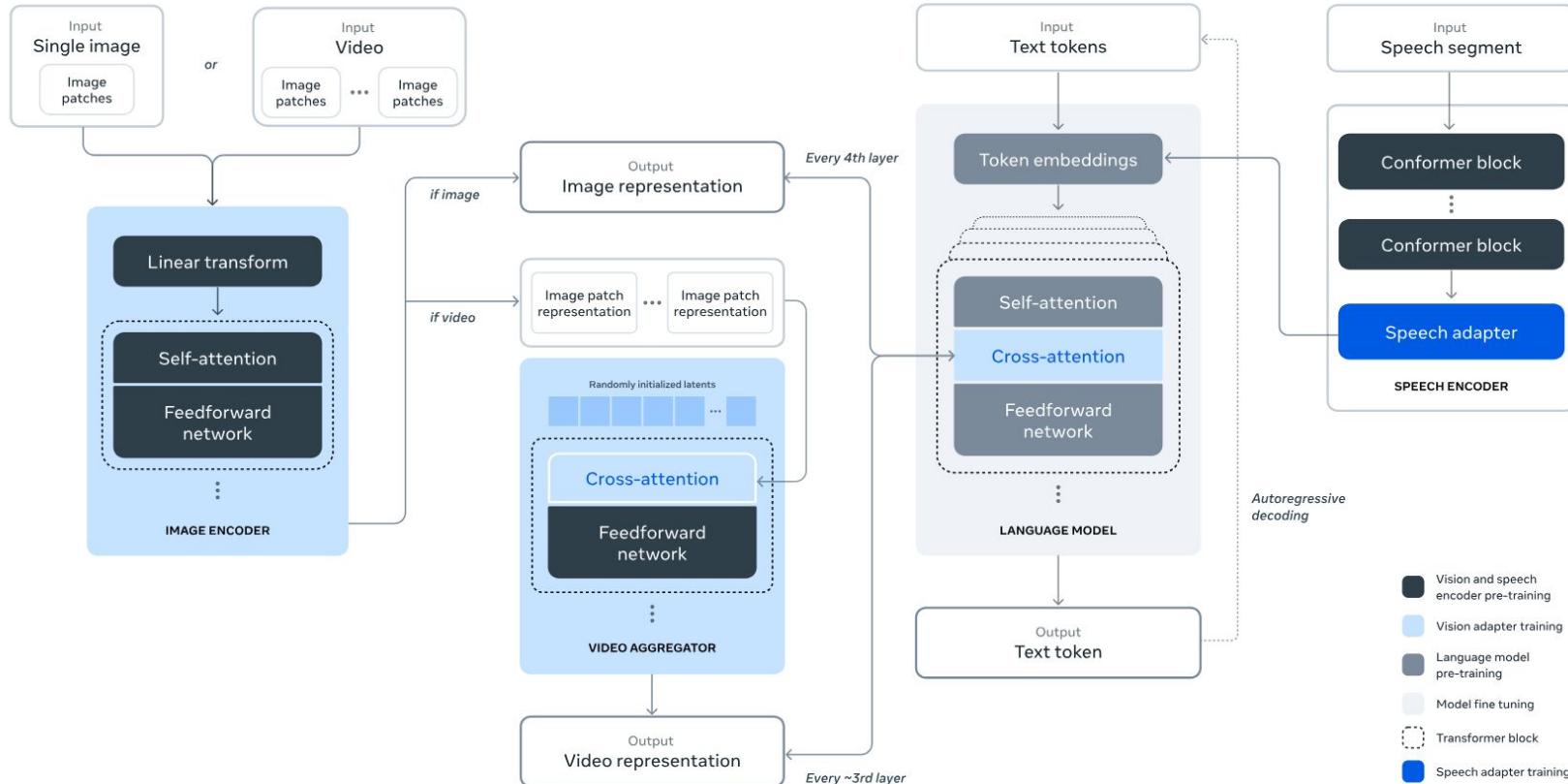
Method 2: leverage cross-attention layer



Method 2: leverage cross-attention layer



Method 2: leverage cross-attention layer



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

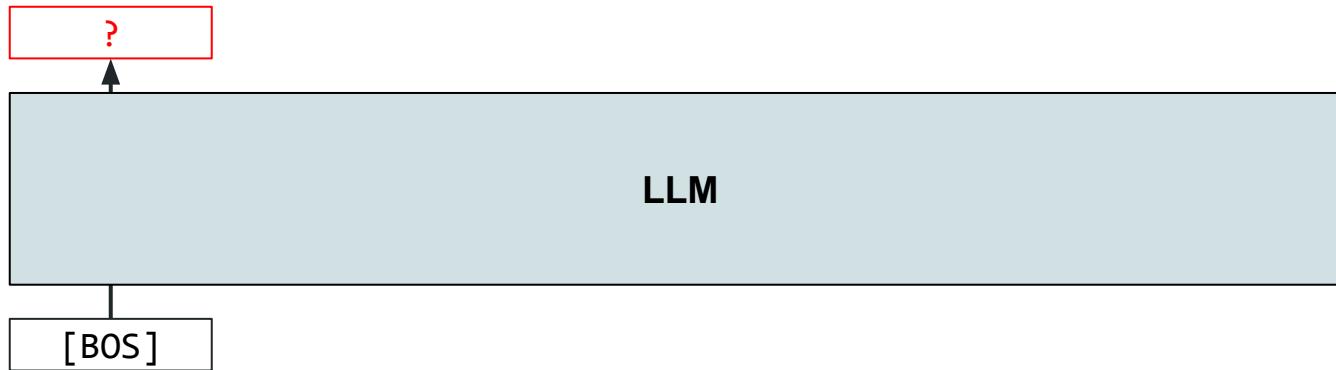
Closing thoughts

Current approach and limitations

LLM

[BOS]

Current approach and limitations



Current approach and limitations

A

LLM

[BOS]

Current approach and limitations

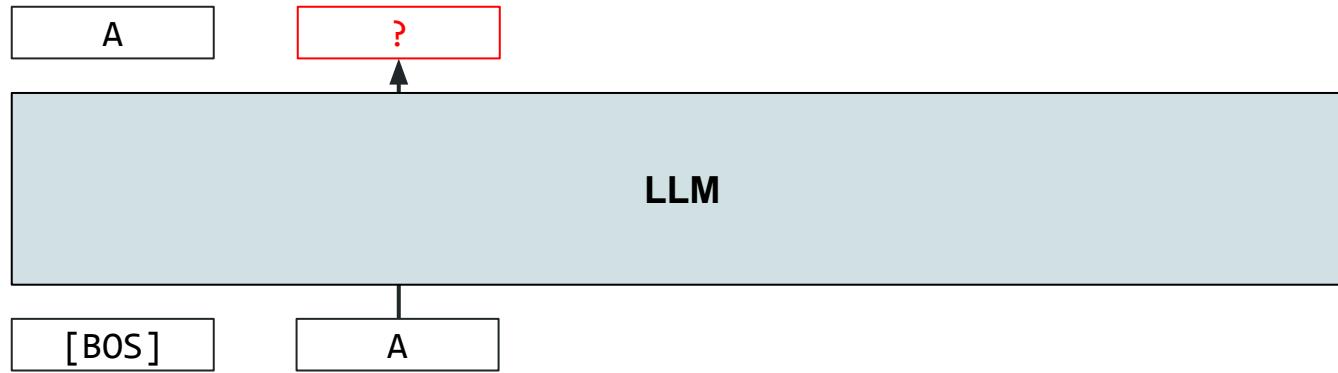
A

LLM

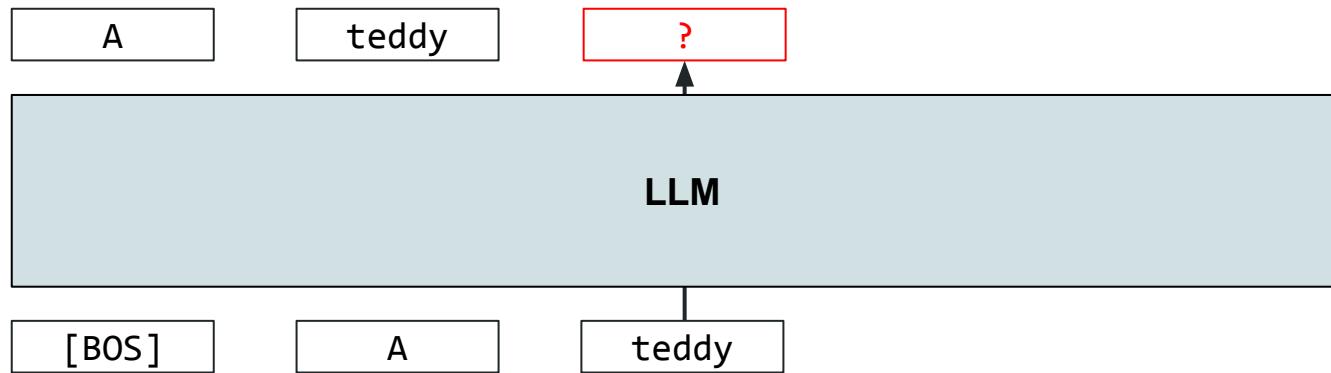
[BOS]

A

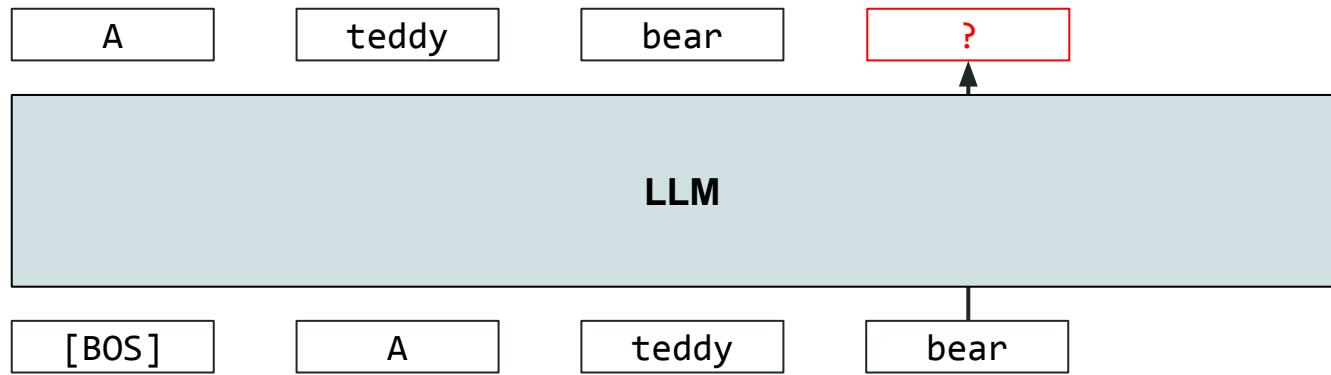
Current approach and limitations



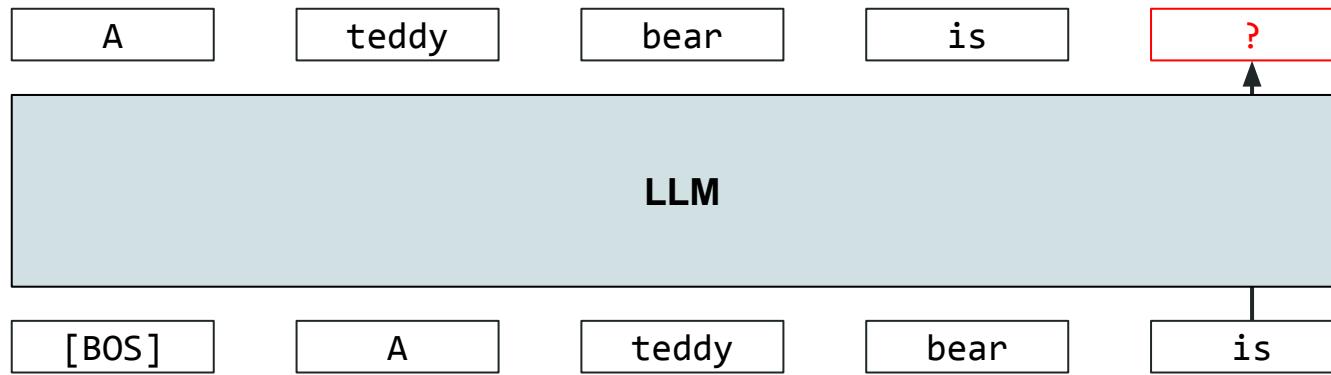
Current approach and limitations



Current approach and limitations

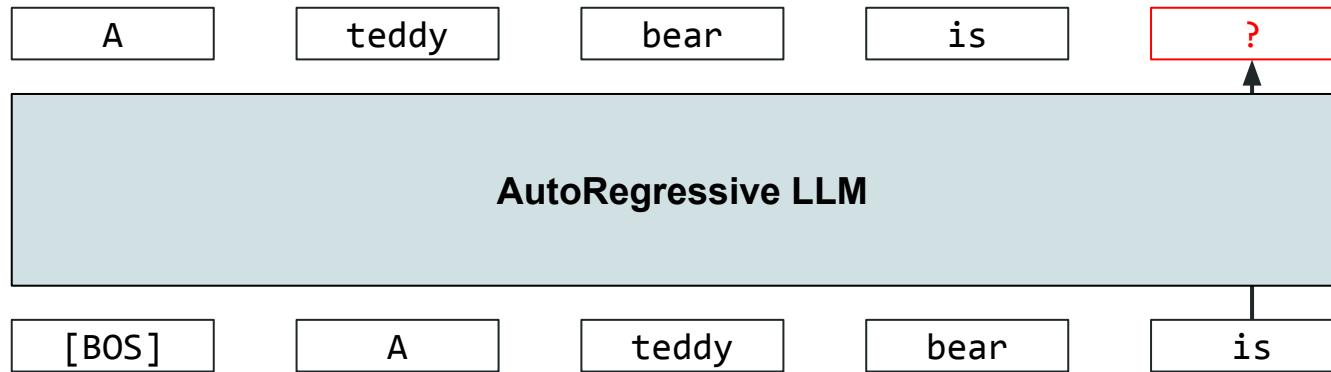


Current approach and limitations



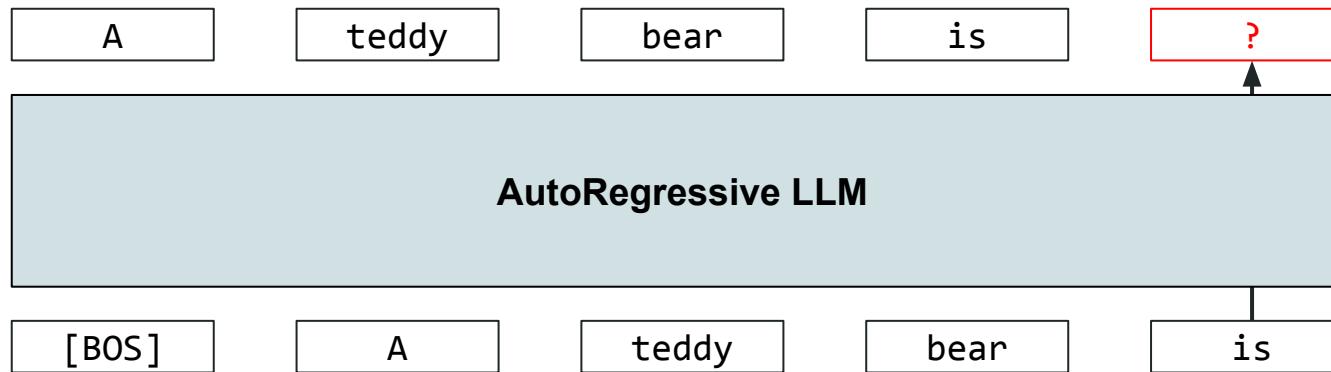
Current approach and limitations

Paradigm. AutoRegressive Modeling (ARM)



Current approach and limitations

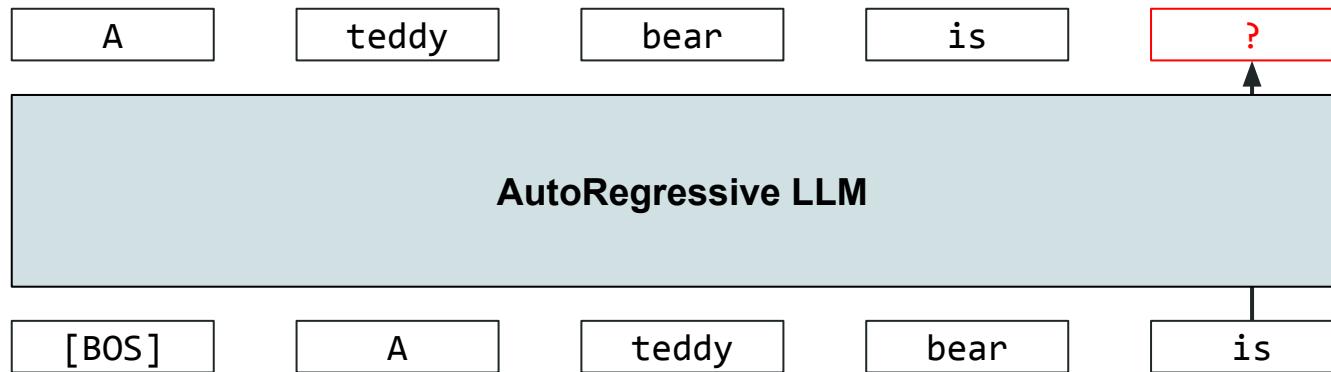
Paradigm. AutoRegressive Modeling (ARM)



Problem. Inference-time generation is not parallelizable

Current approach and limitations

Paradigm. AutoRegressive Modeling (ARM)



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

Alternative approach in the news

Inception raises \$50 million to build diffusion models for code and text

Announcement on TechCrunch, November 6th 2025.

Announcing our \$50M Seed Round, led by Menlo Ventures | >
The Fastest LLMs Ever Built

Diffusion LLMs: A Breakthrough for Speed and Quality

Inception website, screenshot taken on November 26th 2025.

Gemini Diffusion

Our state-of-the-art, experimental text diffusion model

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

July 31, 2025

Seed Diffusion Preview

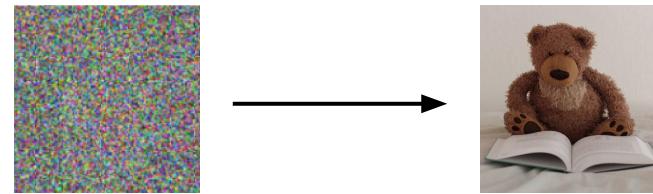
A large scale language model based on discrete-state diffusion, specializing in code generation, achieves an inference speed of 2,146 token/s, a 5.4x improvement over autoregressive models of comparable size.

Read Paper ↗

Try Now ↗

ByteDance announcement, July 31st 2025.

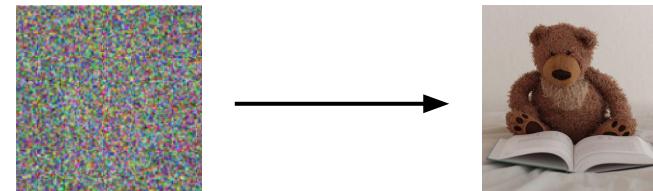
Context on image generation models



Context on image generation models

Intuition

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



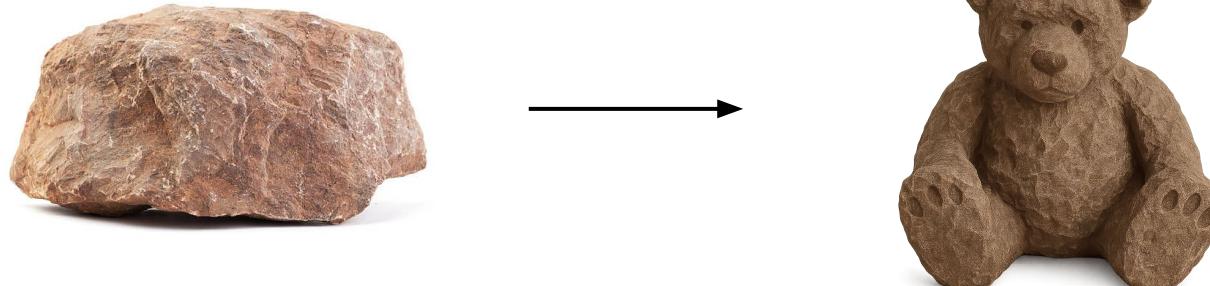
Context on image generation models

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

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

Analogy. Building a sculpture



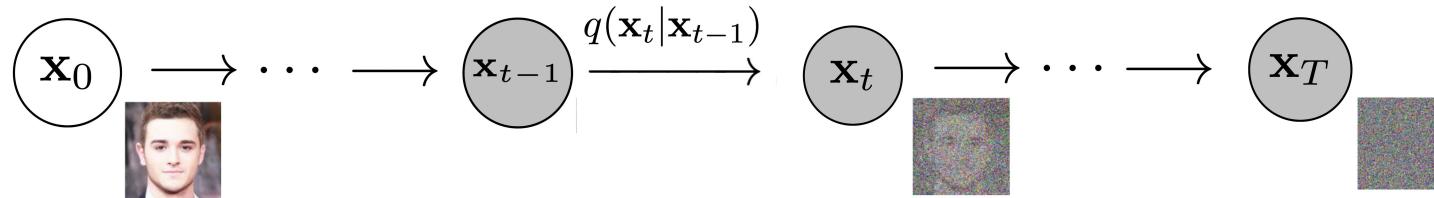
High-level goal of image generation models

Goal

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

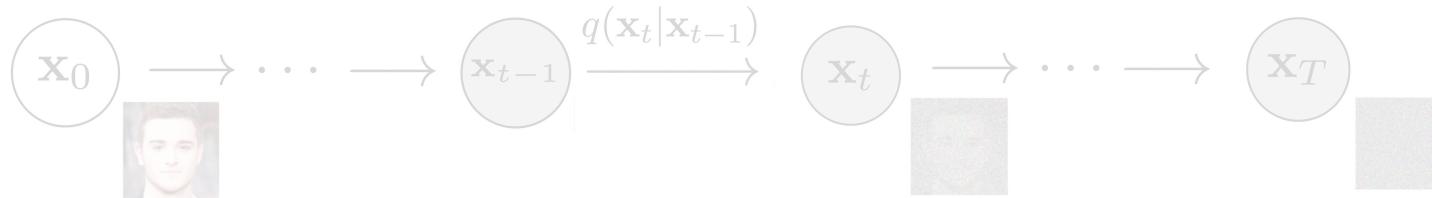
Intuition behind diffusion for images

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

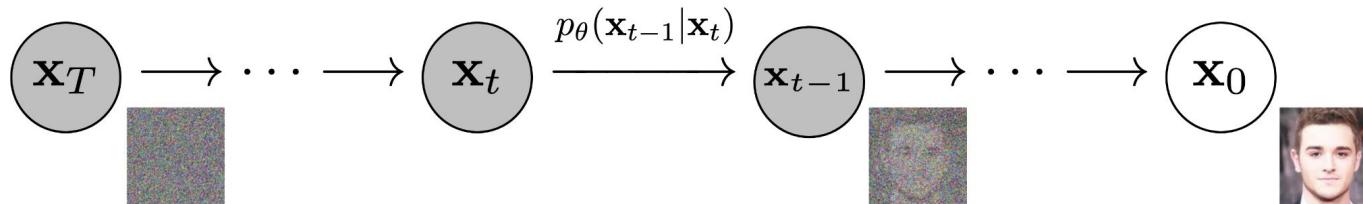


Intuition behind diffusion for images

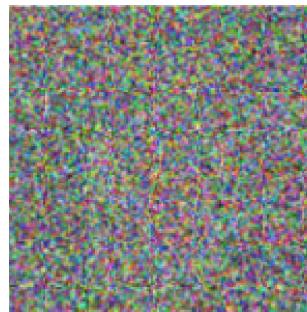
1. Add noise to image (**forward process**)



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



From image to text



Image

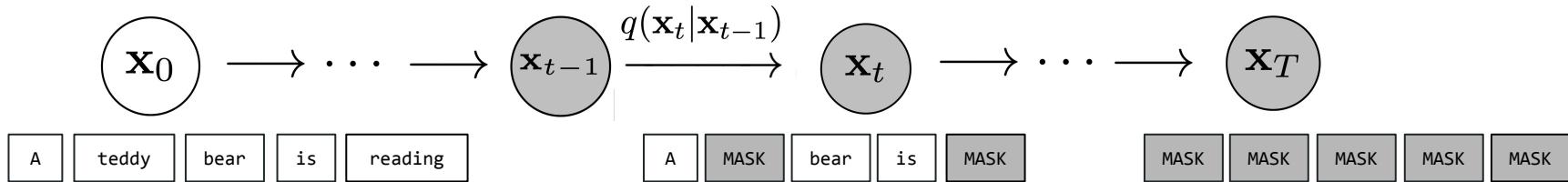


MASK

Text

Intuition behind diffusion for text

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

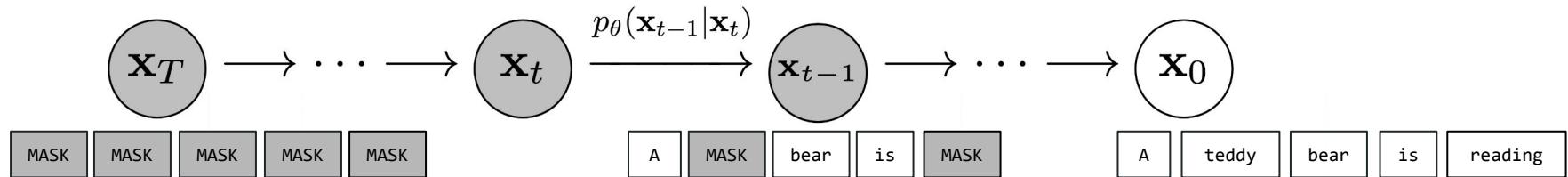


Intuition behind diffusion for text

1. Mask tokens with some probability (forward process)



2. Learn to unmask tokens (reverse process)

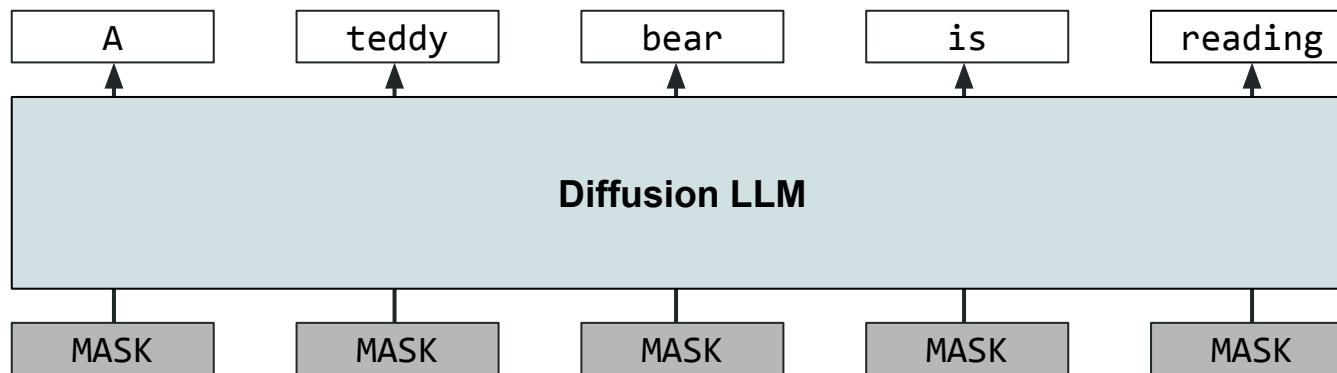


Diffusion in LLMs

MDM = Masked Diffusion Model

Diffusion in LLMs

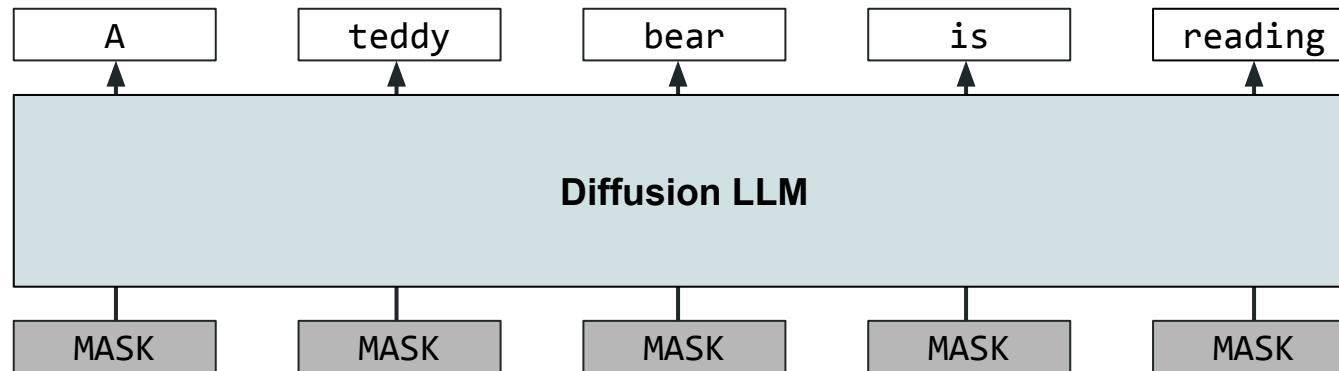
MDM = Masked Diffusion Model



Diffusion in LLMs

MDM = Masked Diffusion Model

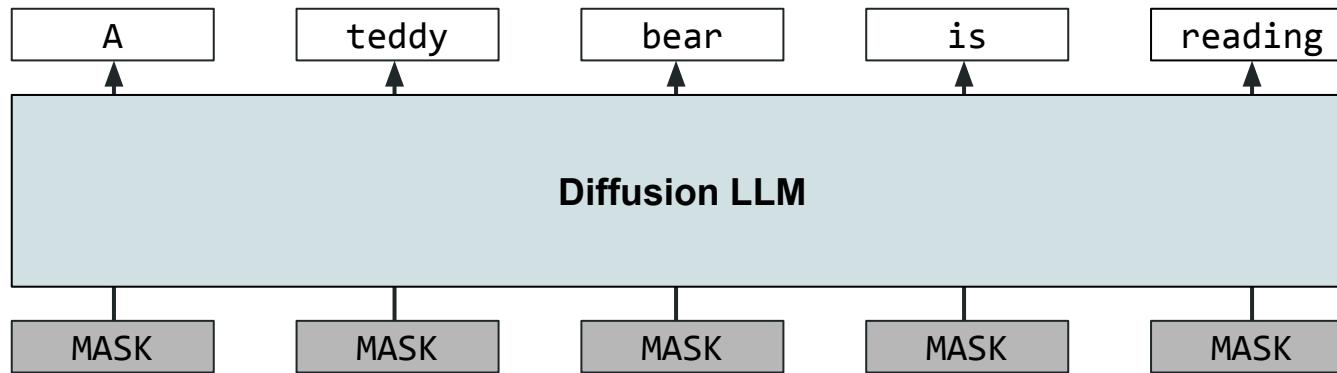
Decoding done in fewer forward passes!



Diffusion in LLMs

MDM = Masked Diffusion Model

Decoding done in fewer forward passes!



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.

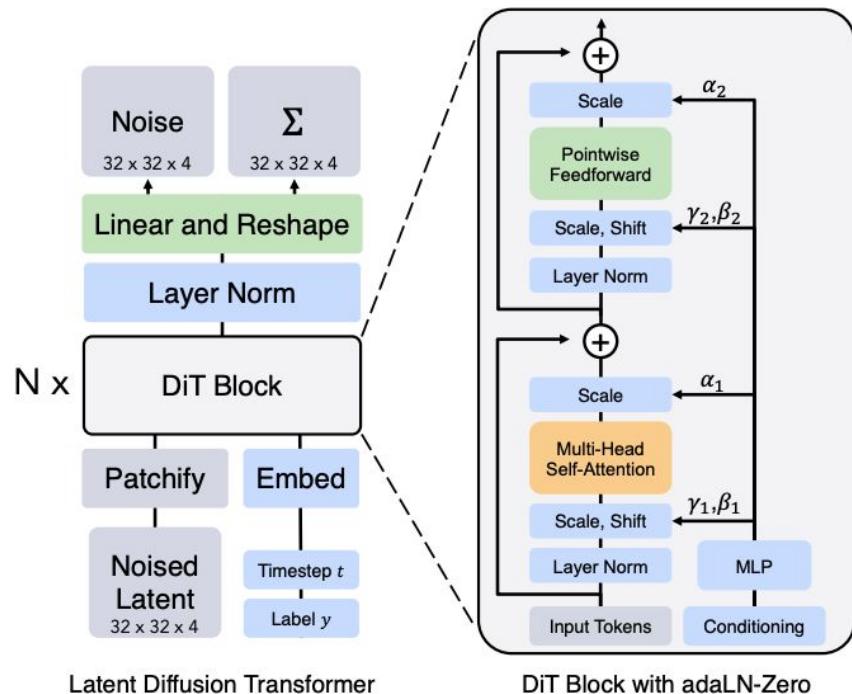
- "Diffusion" training for text output

Cross-pollination between modalities

Trend. Modalities trade ideas on concepts that work

Architecture.

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



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

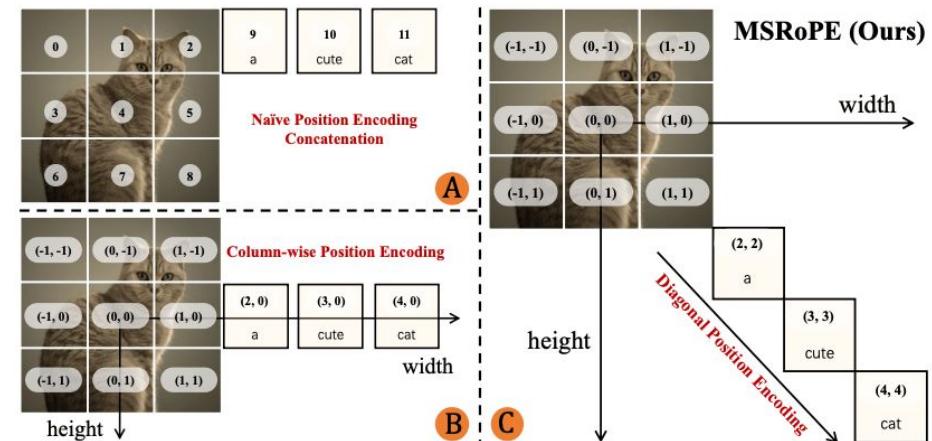
Cross-pollination between modalities

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)

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

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?

Back to the basics

Foundational research still very much ongoing.

Microscopic level. Design choices still widely very across papers.

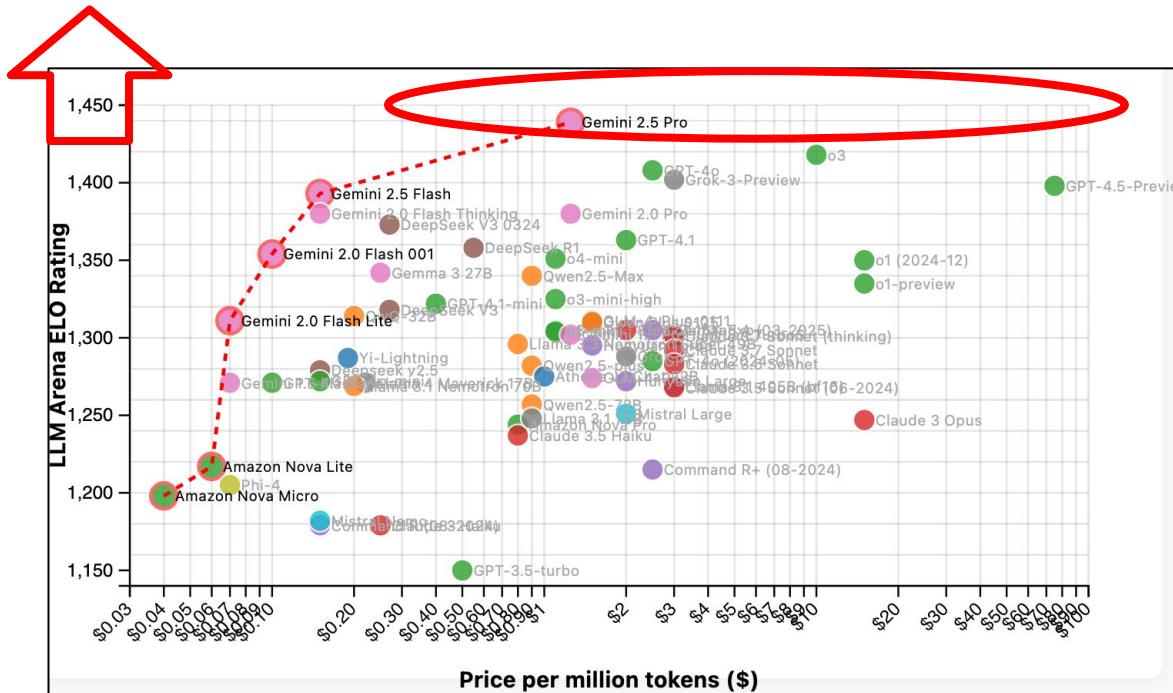
- Optimizers: AdamW vs. MuonClip vs. others
- Normalization?
- MHA/MQA/GQA design choices
- Activation functions
- MoE vs. not
- Number of layers

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

Taking a step back. Transformer = best architecture?

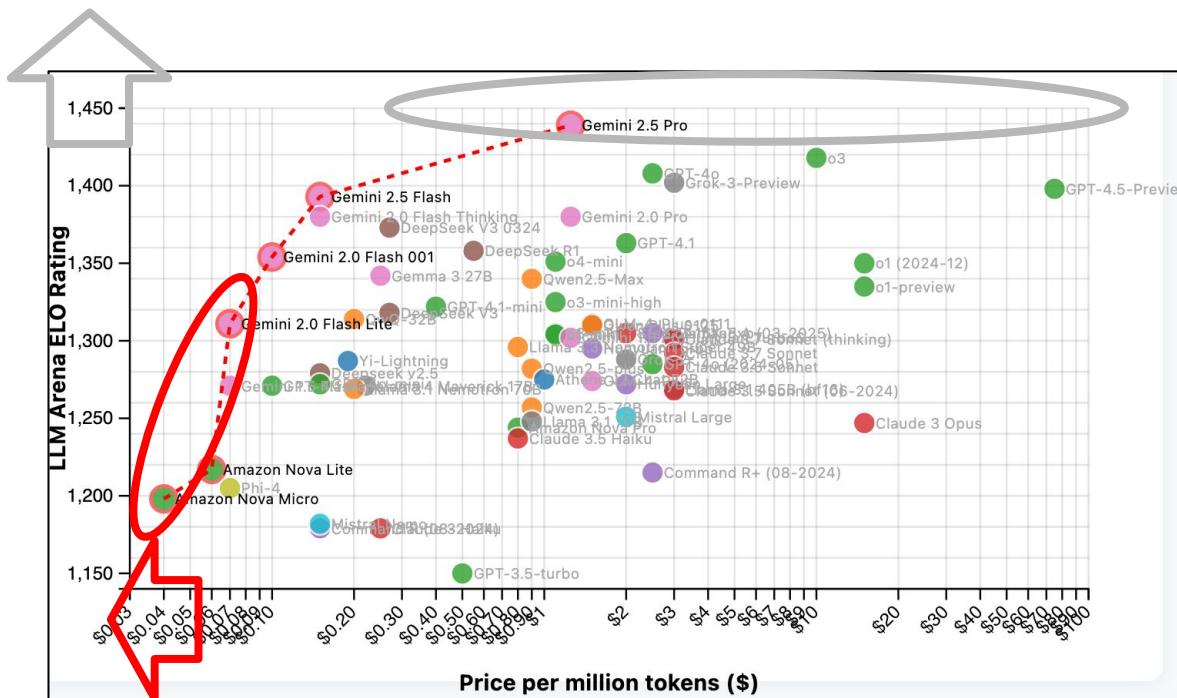
Interest in pushing through new frontiers

Until now. Focus on best performance.



Interest in pushing through new frontiers

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



Hardware optimization

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

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

Hardware optimization

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

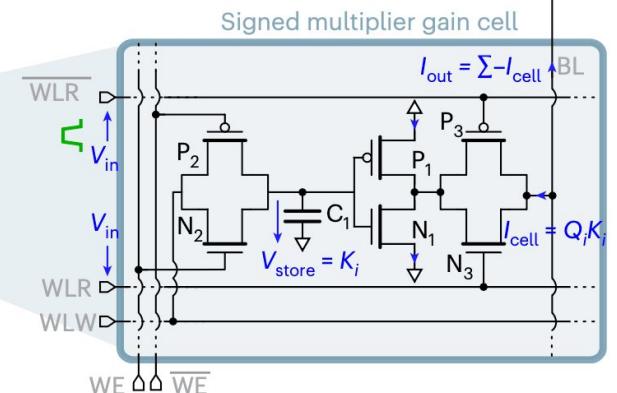
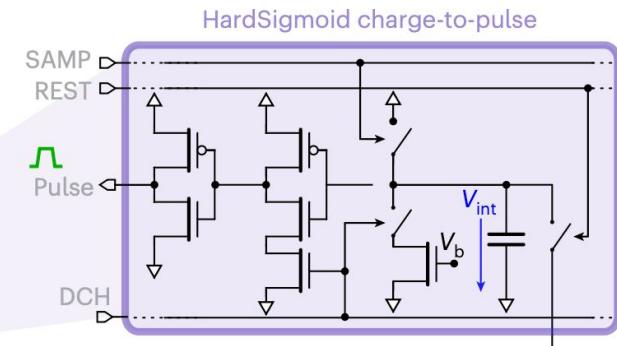
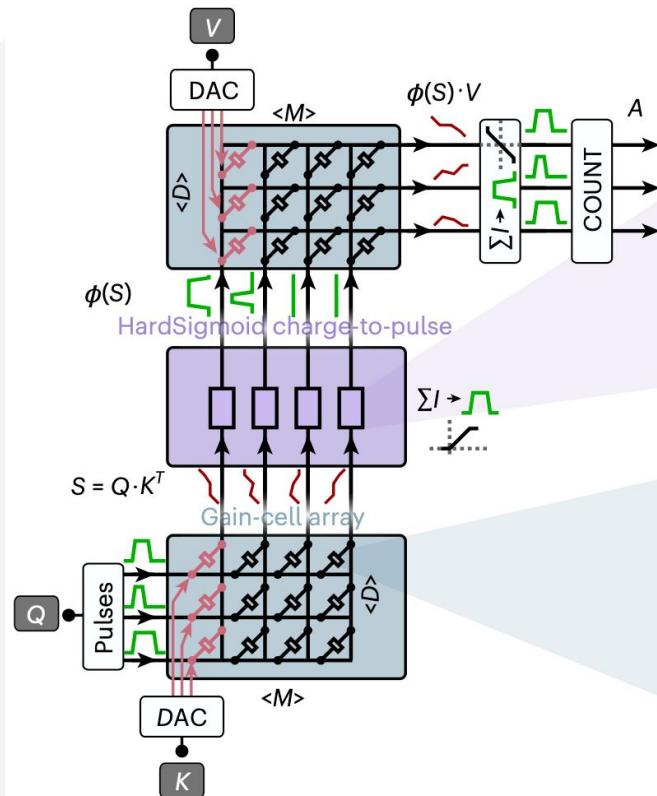
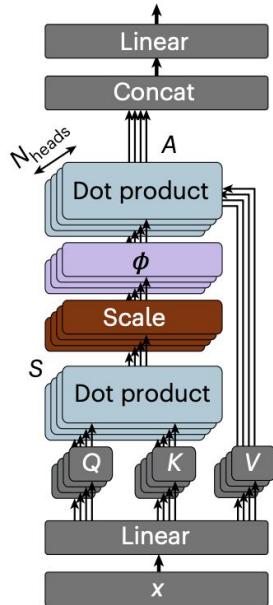
Challenges. Transformer blocks have special needs:

- 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

Hardware optimization



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.

Daily life has already dramatically changed by LLMs

Today. Many uses already.

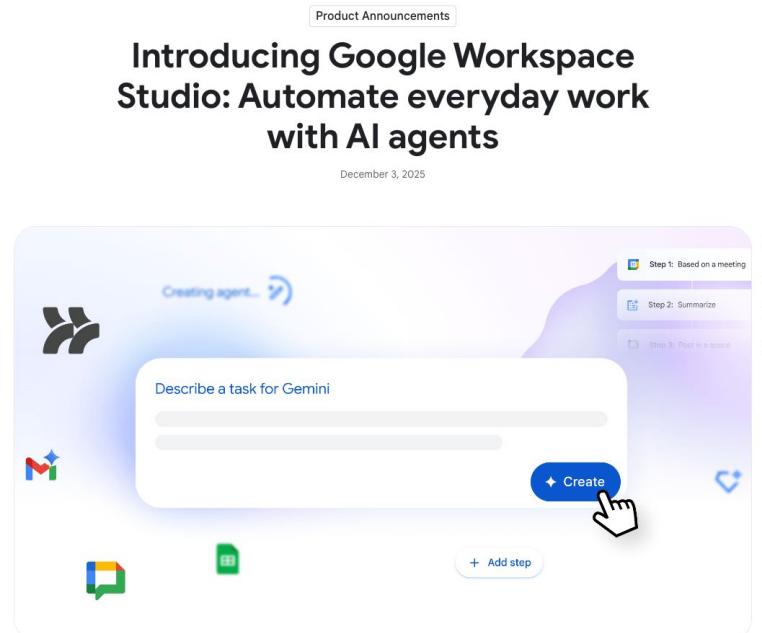
Some key applications:

- 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)



More use cases to be expected

Tomorrow.

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

Near term.

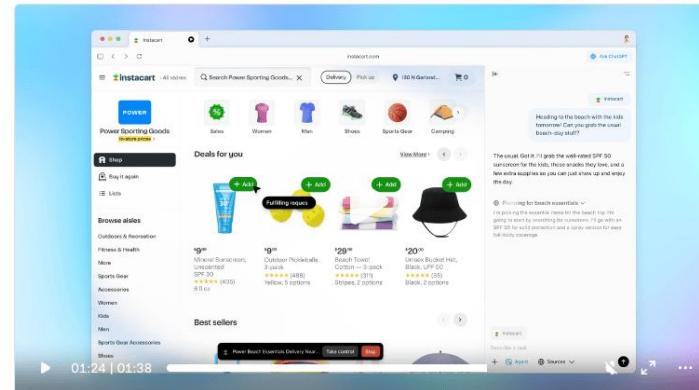
Browser? (e.g. OpenAI's Atlas launch). Looking further: OS-level LLM?

October 21, 2025 Product

Introducing ChatGPT Atlas

The browser with ChatGPT built in.

[Download for macOS ↗](#)



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 (\neq 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, that 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 at an arbitrary level of granularity.

this teddy bear is really cute → T → this[UNK] teddy[UNK] bear[UNK] is[UNK] really[UNK] cute[UNK] [PAD] ... [PAD]

Here are the main types of tokenizers:

Type	Pros	Cons	Illustration
Word	<ul style="list-style-type: none">Easy to interpretShort sequence	<ul style="list-style-type: none">Large vocabulary sizeWord variations not handled	
Subword	<ul style="list-style-type: none">Word roots leveragedIntuitive embeddings	<ul style="list-style-type: none">Increased sequence lengthTokenization more complex	
Character	<ul style="list-style-type: none">No out-of-vocabulary concernsSmall vocabulary size	<ul style="list-style-type: none">Much longer sequence lengthPatterns hard to interpret because too low-level	
Byte			

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

1.2 Embeddings

Definition – An embedding x 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:

$$\text{similarity}(t_1, t_2) = \frac{\mathbf{t}_1 \cdot \mathbf{t}_2}{\|\mathbf{t}_1\| \|\mathbf{t}_2\|} = \cos(\theta) \in [-1, 1]$$

The angle θ characterizes the similarity between the two tokens:

Similar

Dissimilar

Independent

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 .

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.

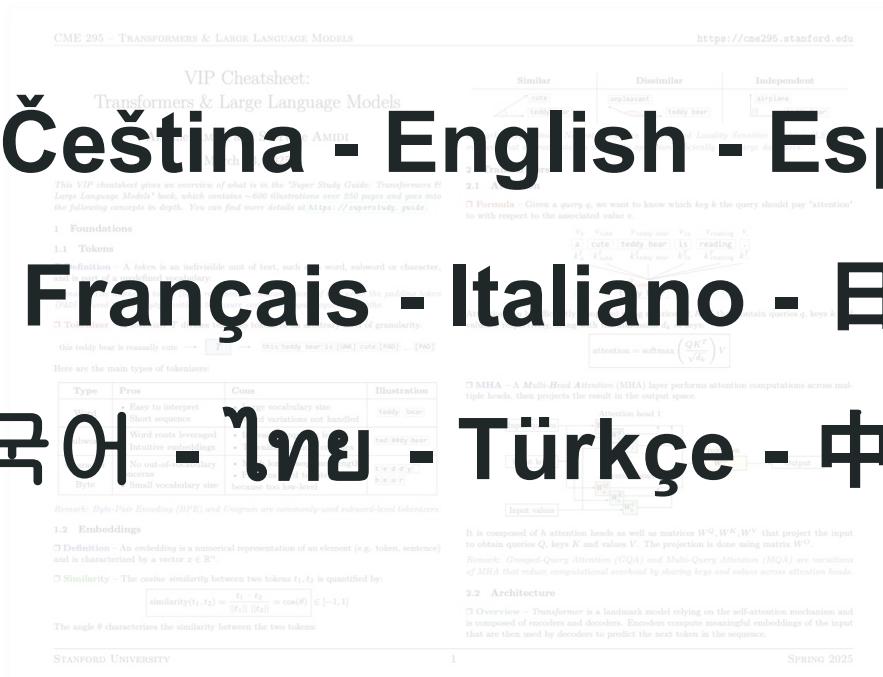
Architecture

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.

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

Maybe useful for after the class

VIP Cheatsheet

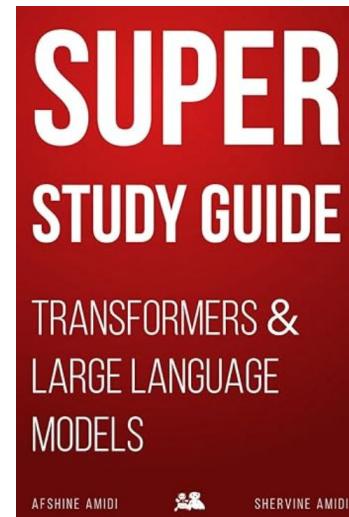


Thank you for this quarter!



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!