NeurIPS Tutorial 2025: Efficient Transformers

Lucas Spangher, Alejandro Queiruga, Zach Gleicher, Ramy Eskander

Efficient Transformers

Introduction to the importance of efficiency in LLMs

NeurIPS 2025

Zachary Gleicher

About me

Zachary Gleicher

Product Manager on Gemini

(1) Gemini Pre-Training
(2) Flash-Lite Post-Training
(3) Gemini Embedding

Efficient Transformers:

State of the art in pruning, sparse attention, and transformer funneling

Agenda

01 Introduction to the importance of efficiency in LLMs

02 Funneling, Distillation, and other Google efficiency Initiatives

03 Attention and Embedding Efficiency

04 Pruning Demo

05 Fireside Chat

Key Takeaways

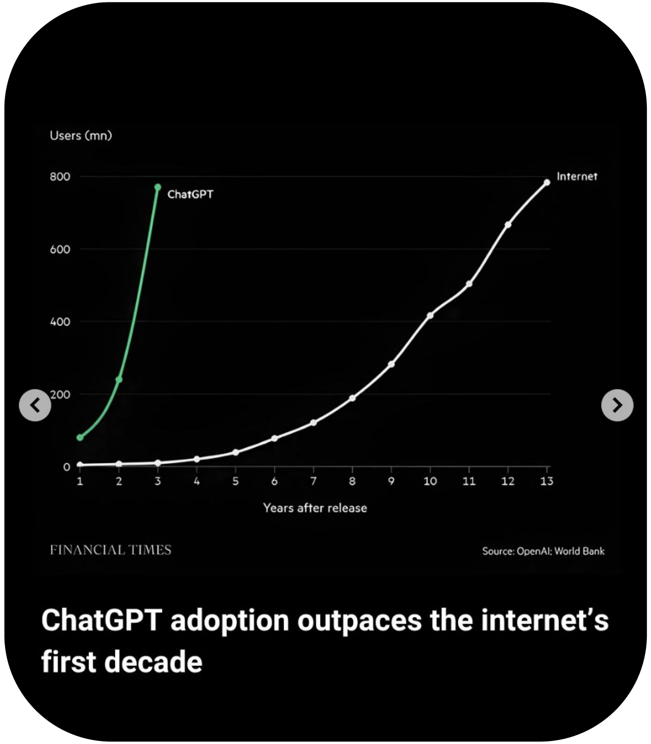
1. **Supply is the Constraint:** There's more token demand than supply - datacenter growth and power are critical bottlenecks
2. **The Opportunity:** Efficiency is critical to unlock more supply
3. **Efficiency has many layers:** Hardware, model, and serving layers

Agenda

Demand for LLMs is surging!

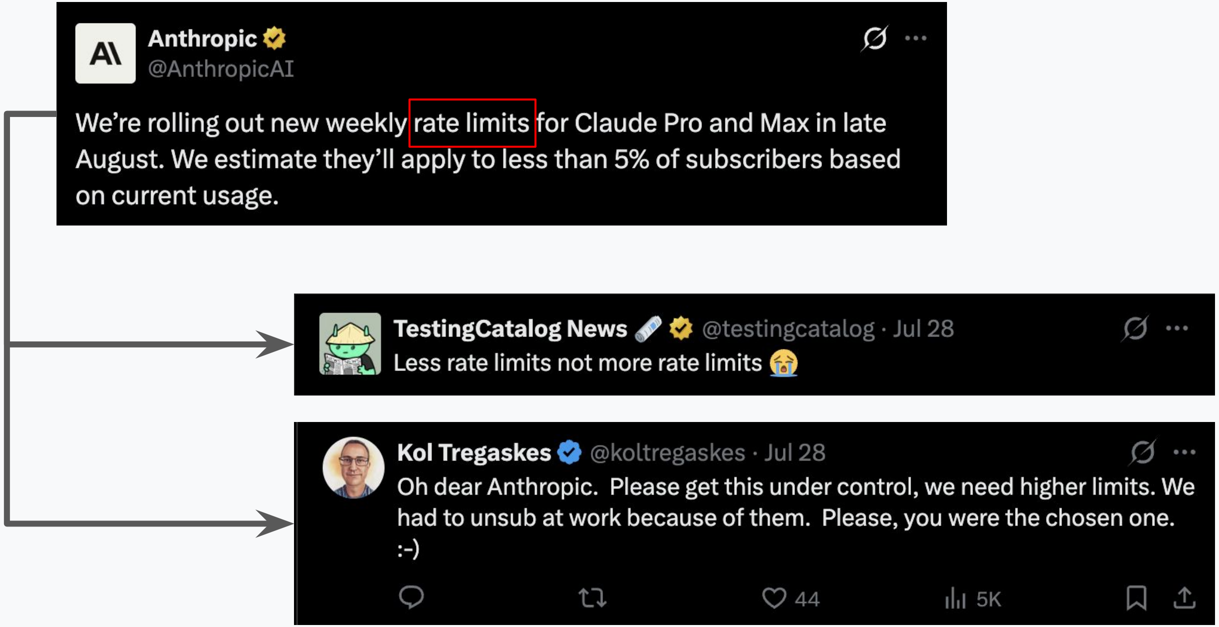
Google processed over
1.3 quadrillion tokens in Oct 2025

In 3 years **ChatGPT** surpassed
800 million weekly users
(~10% of the world's population)



How are LLM providers serving this demand?

They can't! Demand is exceeding supply



The “Hidden Iceberg” of AI costs

GPT-4 was an estimated **$80-$100 million** to train

Inference is cheaper per query but happens billions of times:

- 1.3 Quadrillion tokens in October
- 2.5 Flash is $0.4 input /$2.5 output per 1M tok
- That's **>$1 billion/month** just for inference

How do we meet the demand?

We have two key levers

(1) Increase chip capacity

(2) Improve efficiency

“You should expect OpenAI to **spend trillions** of dollars on datacenter construction in the not very distant future”

- Sam Altman, Aug 2025

He wasn't kidding

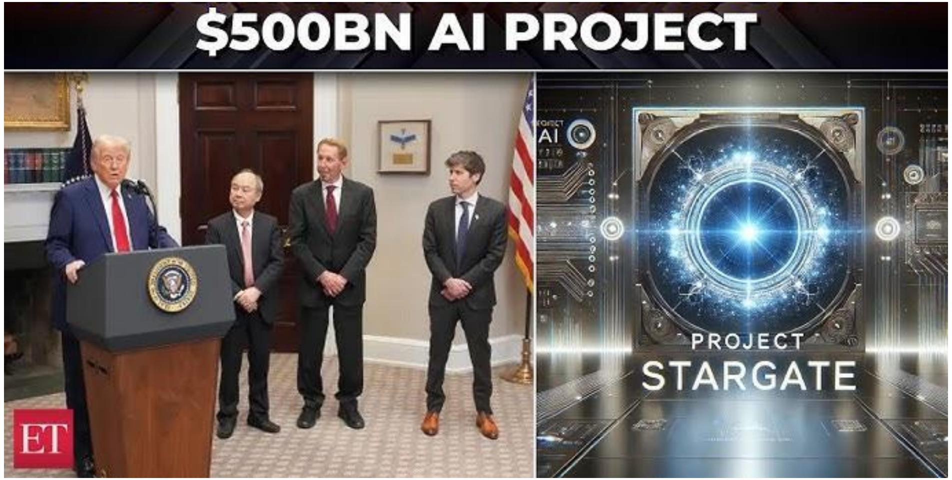
OpenAI and NVIDIA announce strategic partnership to deploy 10 gigawatts of NVIDIA systems

OpenAI and Broadcom announce strategic collaboration to deploy 10 gigawatts of OpenAI-designed AI accelerators

Multi-year partnership enables OpenAI and Broadcom to deliver accelerator and network systems for next-generation AI clusters.

OpenAI | NVIDIA

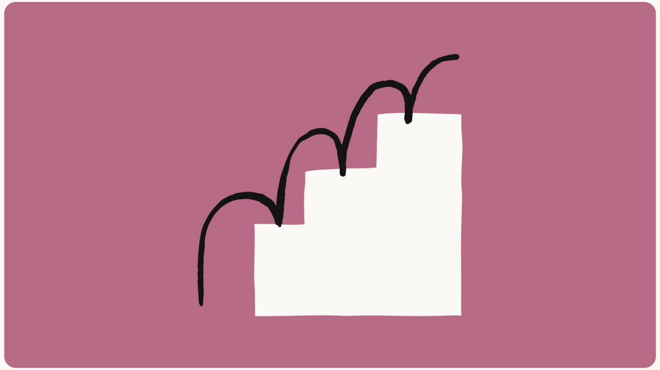
OpenAI | BROADCOM



And that was just OpenAI

Expanding our use of Google Cloud TPUs and Services

Oct 23, 2025 • 2 min read



“Including up to **one million TPUs**…

“worth **tens of billions** of dollars”

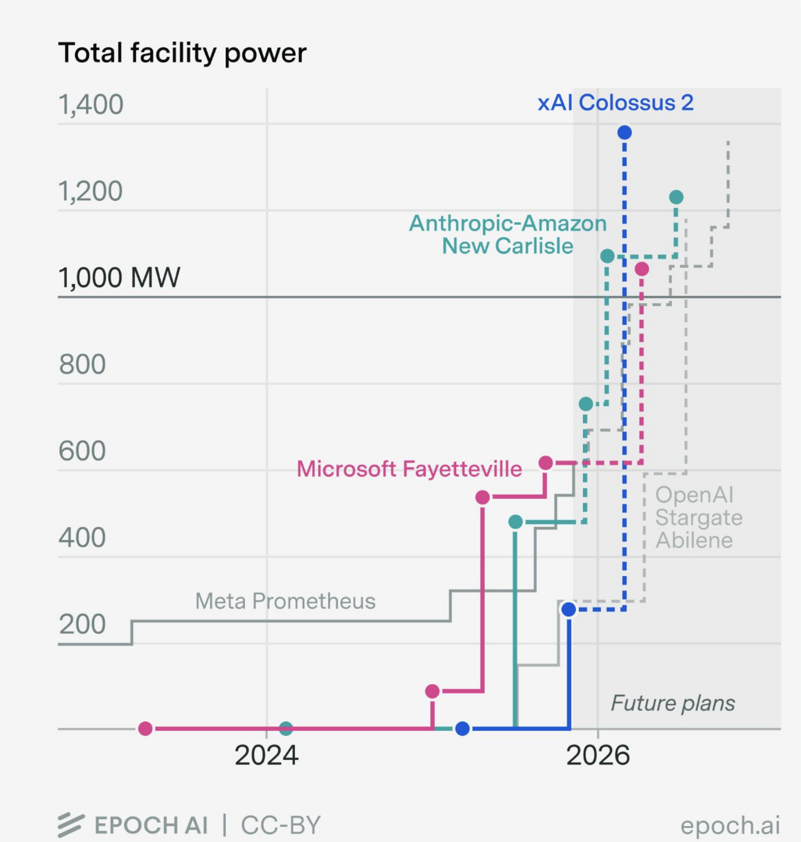
“a **gigawatt** of capacity”

Elon Musk @elonmusk X.com

Having thought about it some more, I think the 50 million H100 equivalent number in 5 years is about right.

Eventually, billions.

5 data centers are set to reach a GW of power in 2026



And climbing towards 5GW data centers in 2028

“The U.S. AI sector needs at least **50GW** of electric capacity by 2028 to maintain global AI leadership”

For context, this is roughly **4x** New York City’s peak electricity demand,

5GW

5GW data center required to train a single model in 2028

x

~5 Frontier AI Labs

x

2x for inference

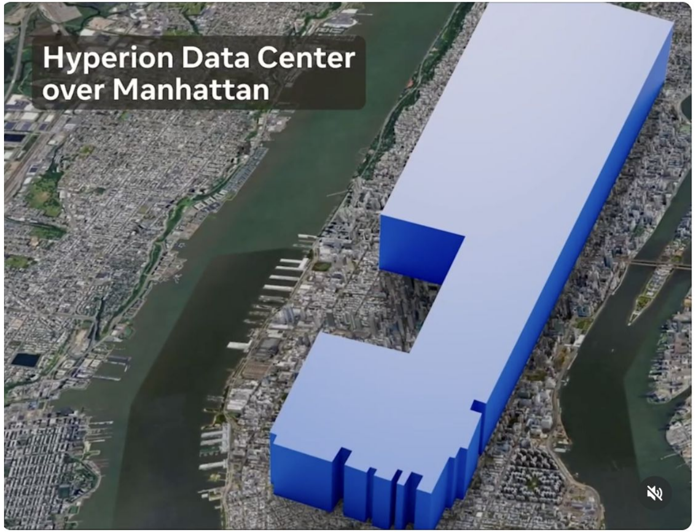
=

50GW

Source: Anthropic, Build AI in America

How big is a 5GW data center?

Size: Meta's Hyperion datacenter (Louisiana) footprint will be large enough to cover most of Manhattan.



Source

Energy: Typical nuclear fission plant is 1GW

source

Do we have enough power?

"The Department of Energy warns that **blackouts** could increase by **100 times** in 2030 if the U.S. continues to shutter reliable power sources and fails to add additional firm capacity"

"Grid growth must match the pace of AI innovation"

Source: Department of Energy

The Challenge and Opportunity

The Challenge - Critical Bottlenecks

1. Data center build outs take time and capital

2. Power constraints

The Opportunity - Users want More

Users are frustrated by rate limits!

Efficiency is critical to unlock more supply

Agenda

01 Background & Motivation

02 Model Efficiency Overview

Two opportunities for improving model efficiency

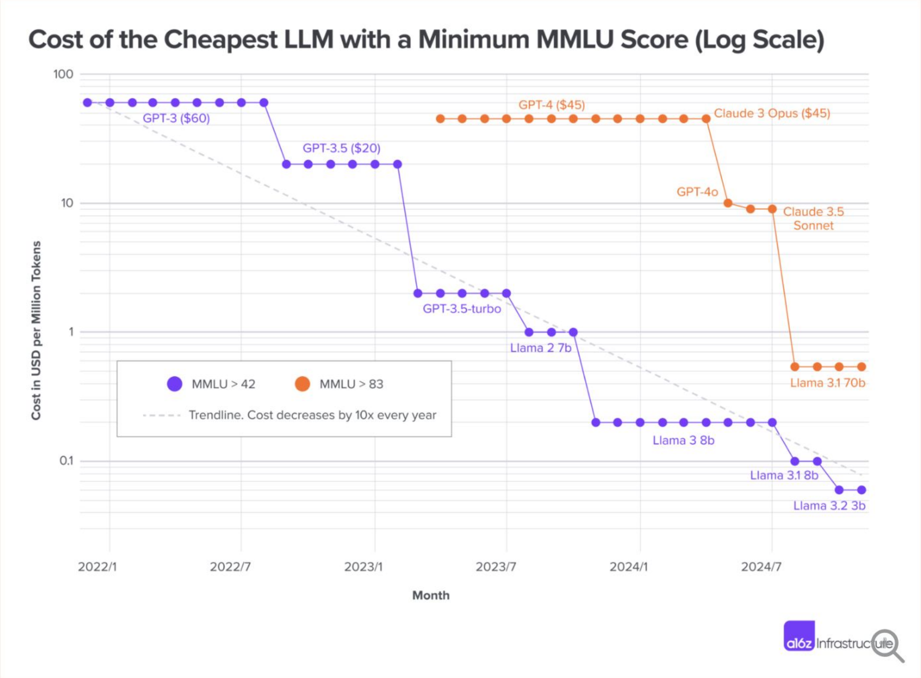
(1) Improve training efficiency

(2) Improve serving efficiency

Some Good News

Serving cost is decreasing by **~10x per year** while remaining quality neutral

- **LLama 3.2 3B** matched the **175B GPT-3** model in ~3 years (1,000x price decrease). Source: a16z
- DeepSeek-V3 (Dec 2024) reduced inference costs by **~36x**, compared with GPT-4o (May 2024) Source: McKinsey



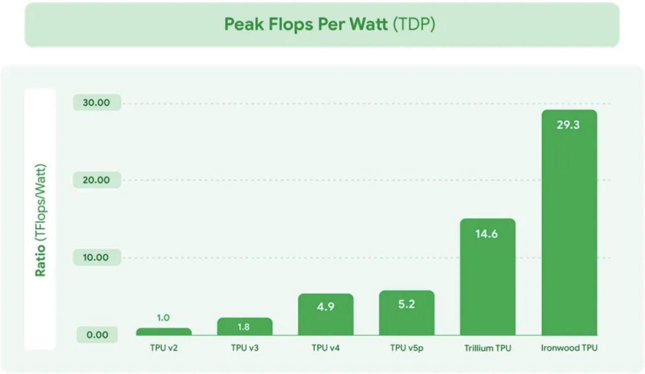
Source: a16z

Where are the efficiency wins coming from?

(1) Hardware Layer
(2) Model Layer
(3) Serving Layer

Hardware Layer Examples

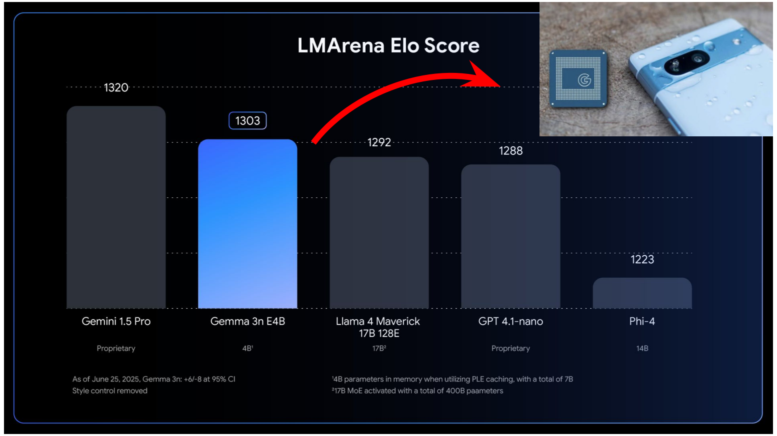
Example 1: Hardware Design



Ironwood perf/watt is **30x more power efficient** than our first Cloud TPU from 2018 (source)

Example 2: On-Device

Over 1 billion smartphones sold a year (source)



Gemma 3n (4B, June 2025) can **run on a mobile phone** (Source)

Where are the efficiency wins coming from?

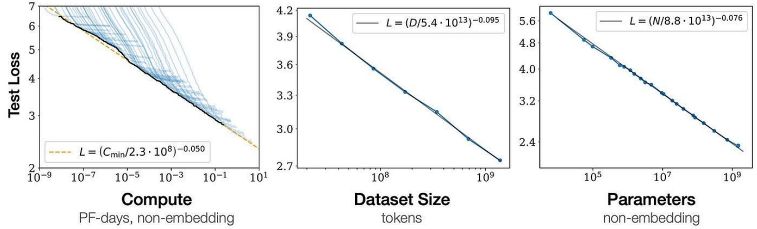
(1) Hardware Layer

(2) Model Layer

(3) Serving Layer

Model Layer Examples

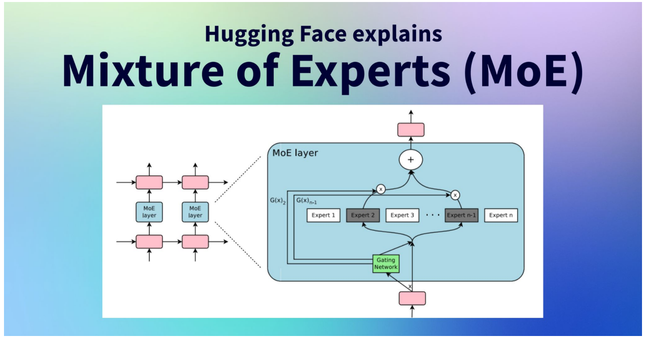
Example 1: Scaling Laws



With 300B tokens, GPT-3 could have been a ~15B model at neutral quality

Source: Training Compute-Optimal Large Language Models, 2022

Example 2: Architecture



Source: Hugging Face

DeepSeek-V3: 671B total params, 37 billion active: a **5.5% activation rate**

Where are the efficiency wins coming from?

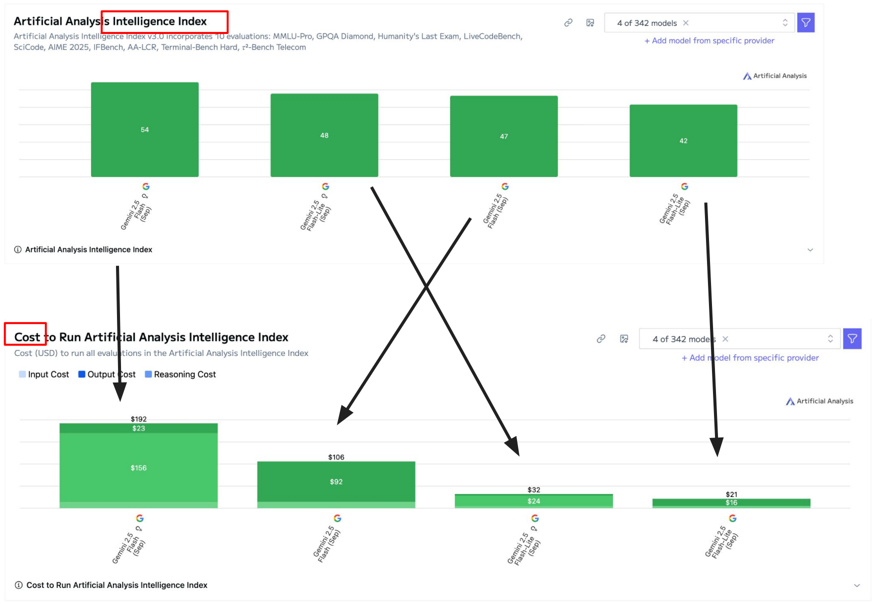
(1) Hardware Layer

(2) Model Layer

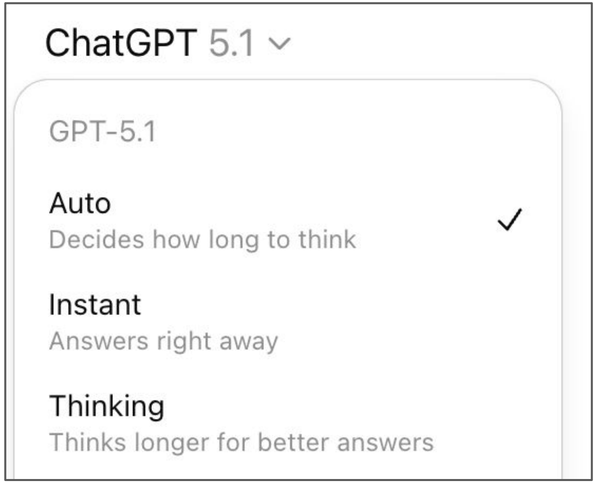
(3) Serving Layer

Serving Layer Examples

Example 3: Reasoning



Example 2: Routing



Key Takeaways

1. **Supply is the Constraint:** There's more token demand than supply - datacenter growth and power are critical bottlenecks
2. **The Opportunity:** Efficiency is critical to unlock more supply
3. **Efficiency has many layers:** hardware, model, and serving layers

Efficient Transformers

KV Compression, Funneling and Sparse Attention for Long Context

Ramy Eskander,
Google, Core ML
reskander@google.com

NeurIPS 2025
12/02/2025

Agenda

- ❖ Introduction
- ❖ KV Compression Multi-Head Latent Attention (MLA)
- ❖ Funnel Transformer
- ❖ Transient Global Attention (TGA)

Introduction

The Problem of Long Context!

Prompt:

What generates the most operating income for company XYZ in 2025?
+ a 100-page business review annual report.

Using Llama 70B and 4 A40 GPUs:

KV Cache Memory: 327K per token

Process Context:	9.5 sec
Process User Query:	0.3 sec
Generate Output (per token)	0.1 sec

The Problem of Long Context!

- **Can we make the model smaller?**
  Distillation, structured pruning, Quantization, …

- **Can we speed up the processing without sacrificing quality?**
  Sharding, **KV Compression**, ….

- **Is there a way to shorten the context?**
  Map-Reduce, Iterative Refinement, Token summarization, **Funnel Transformer** …

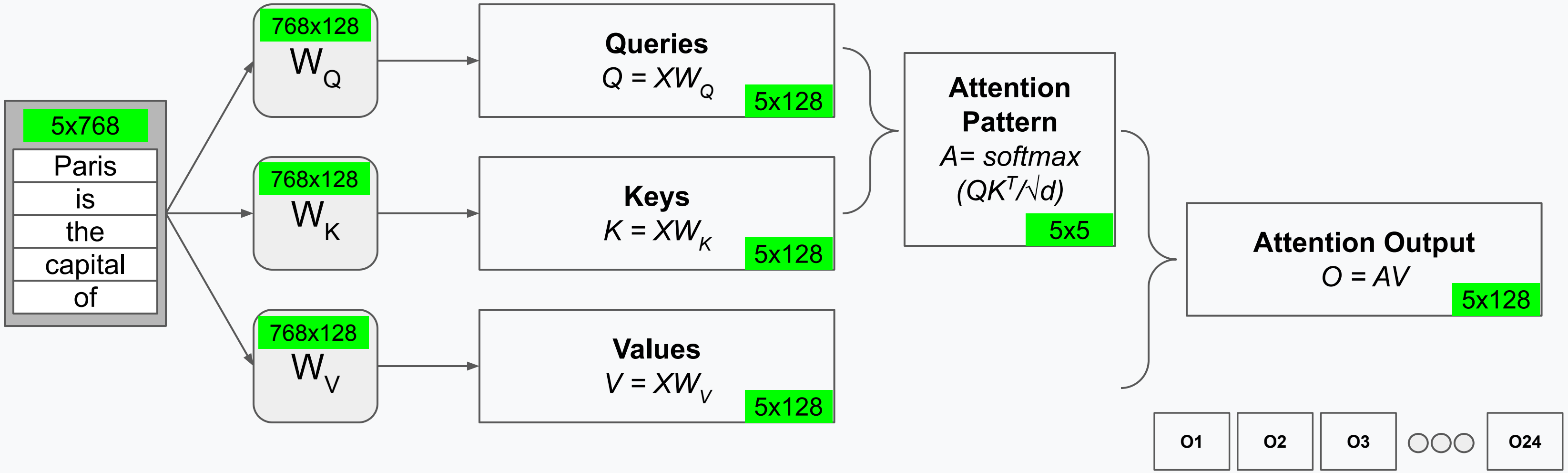
- **Can we speed up Global Self-Attention (n^2d)?**
  Speculative Decoding, Flash Attention, Sparse attention mechanisms, e.g., **Transient Global Attention**, …

KV Compression

Multi-Head Latent Attention (MLA)

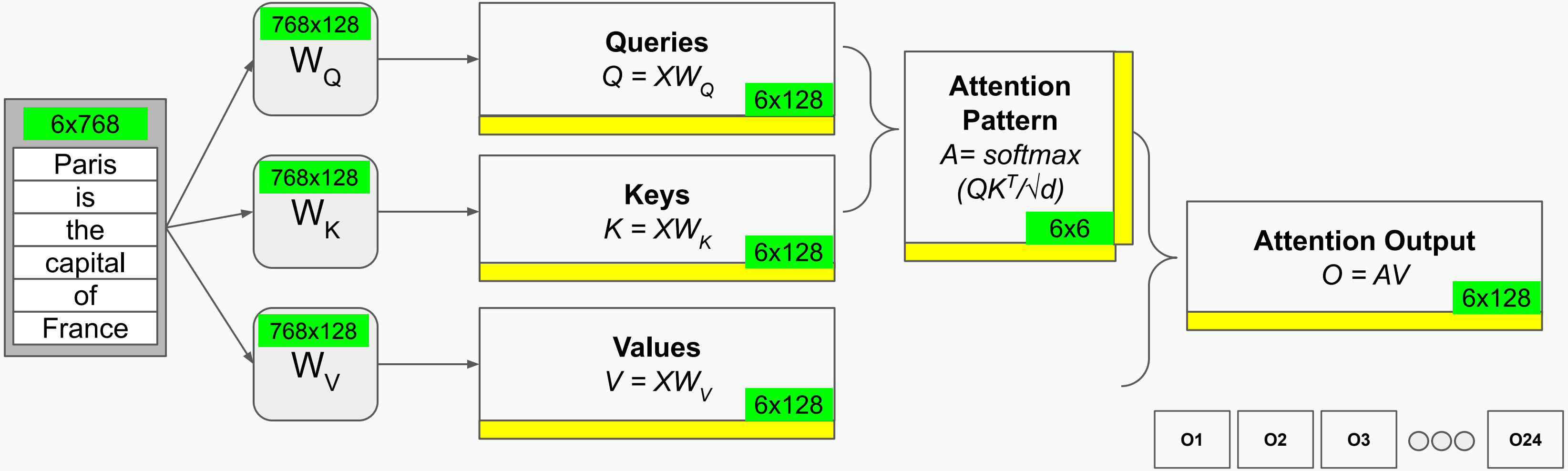
No KV Compression: Architecture

Number of tokens: 5
Embedding dimension = 768
Key size = 128
Number of attention heads: 24



No KV Compression: Architecture

Number of tokens: 5
Embedding dimension = 768
Key size = 128
Number of attention heads: 24



No KV Compression: Size

The Size of the KV cache before compression is defined as: ***2.nh.dh.l*** per token

Where:
n_h = number of attention heads per layer
d_h = key size
l = number of layers

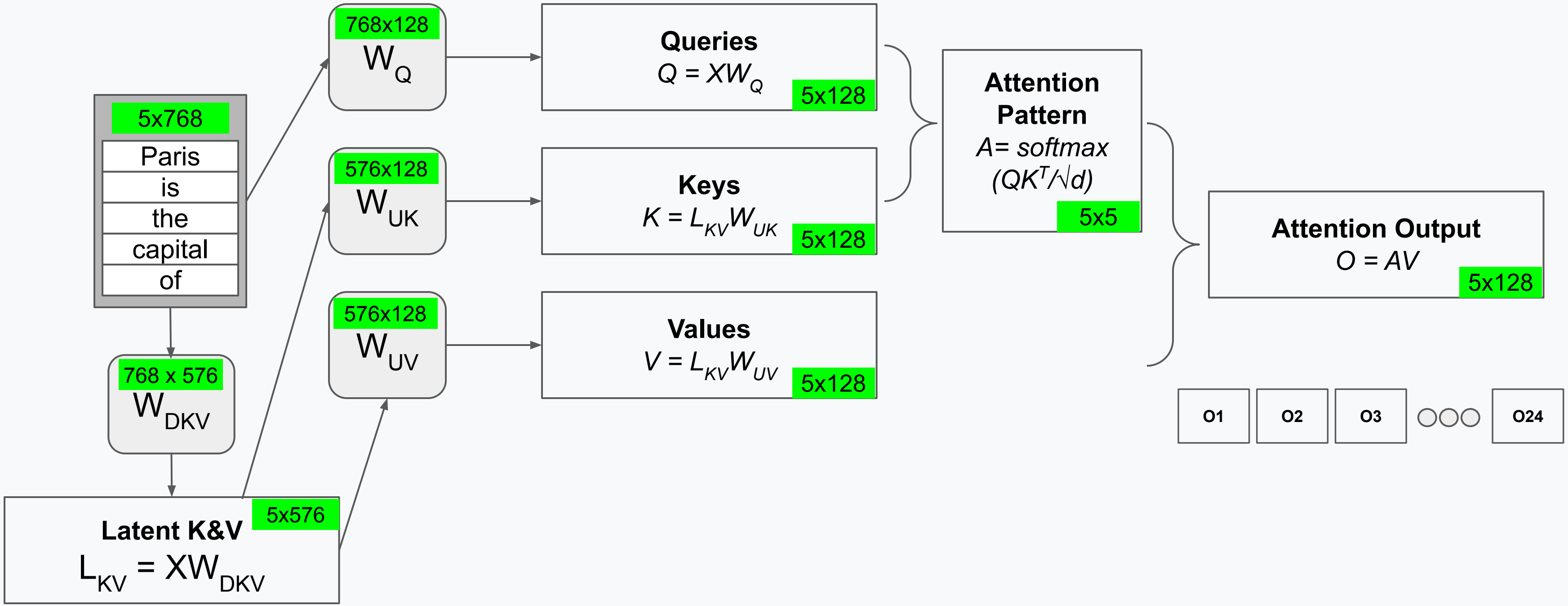
Given n_h = 128, d_h = 128 and l = 61 and 16-point floats (DeepSeek R1/V3 architecture)
The Size of the KV cache = 2 x 128 x 128 x 61 x 2 = 4MB per token

No KV Compression: Size

Attention Mechanism	KV Cache Size (per token)	KV Cache Size (per token)	Quality
Multi-Head Attention (MHA)	$2.n_h.d_h.l$	4MB	High
Multi-Query Attention (MQA)	$2.d_h.l$	31KB	Low
Grouped-Query Attention (GQA)	$2.n_g.d_h.l$	500KB	Medium

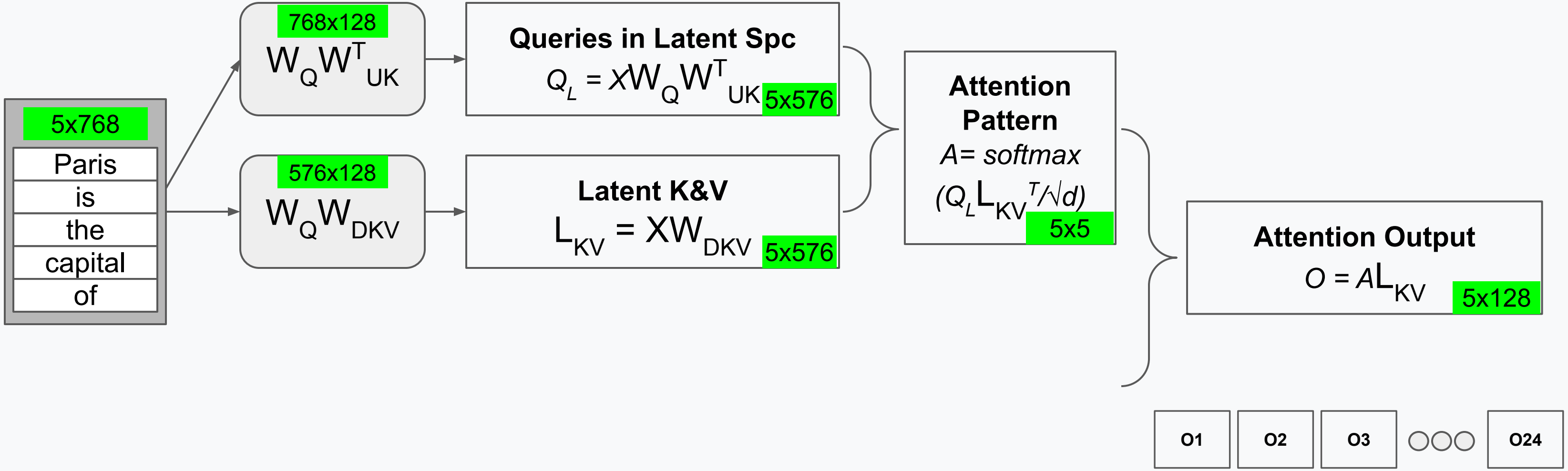
KV Compression (MLA): Architecture

Number of tokens: 5
Embedding dimension = 768
Key size = 128
Number of attention heads: 24



KV Compression (MLA): Architecture

Number of tokens: 5
Embedding dimension = 768
Key size = 128
Number of attention heads: 24



[DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model]

KV Compression (MLA): Size

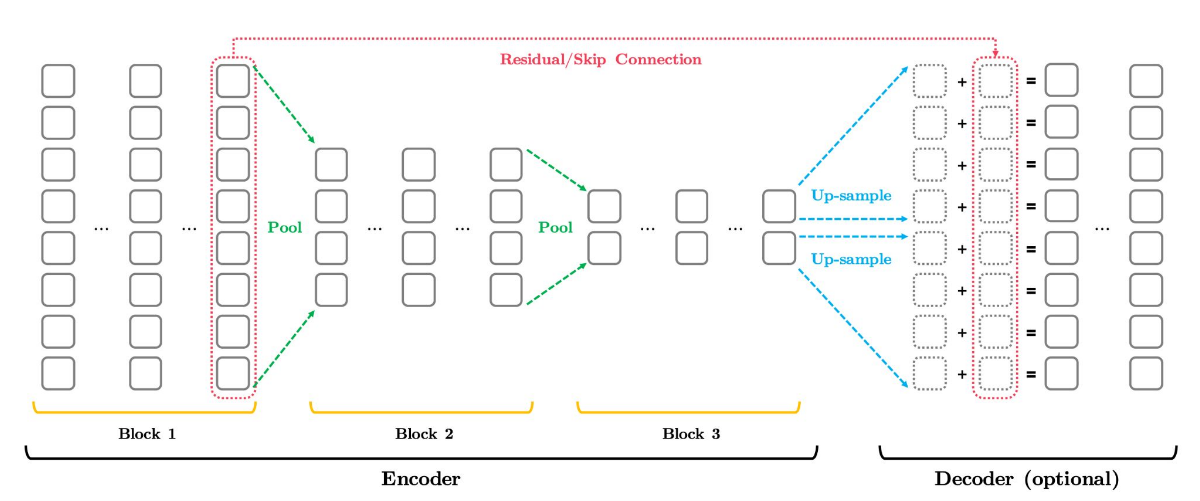
Attention Mechanism	KV Cache Size (per token)	KV Cache Size (per token)	Quality
Multi-Head Attention (MHA)	$2.n_h.d_h.l$	4MB	High
Multi-Query Attention (MQA)	$2.d_h.l$	31KB	Low
Grouped-Query Attention (GQA)	$2.n_g.d_h.l$	500KB	Medium
Multi-Head Latent Attention (MLA)	$d_l.l$	70KB	Best

Funnel Transformer

Funnel Transformer: Overview

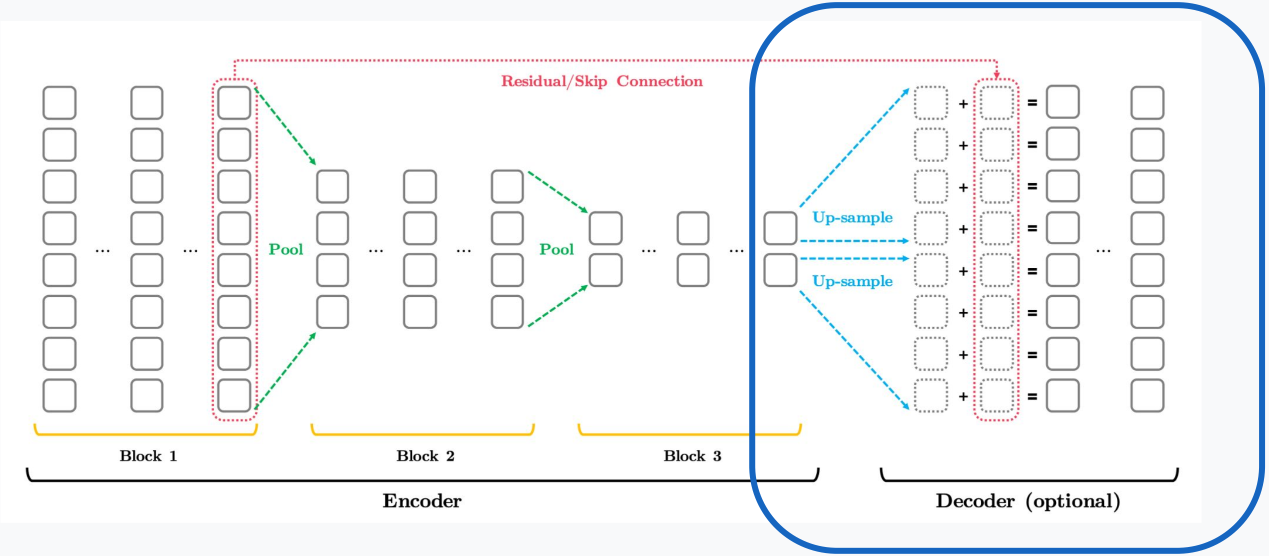
Funnel Transformer is a type of Transformer that progressively compresses the sequence length of hidden states, creating a funnel-like structure.

This compression strategy decreases the computational time and costs. Model capacity can then be further improved by reinvesting the saved FLOPs from length reduction in constructing a deeper or wider model.



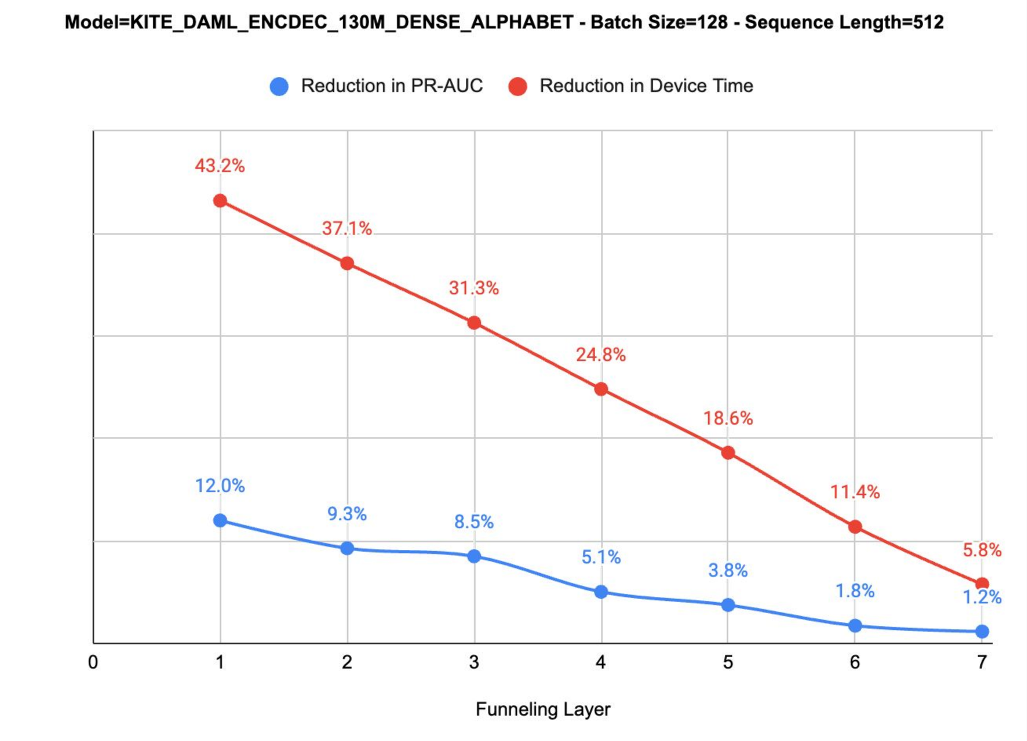
Funnel Transformer: Overview

For tasks involving per-token predictions, a simple decoder is used to reconstruct a full sequence of token-level representations from the compressed encoder output.



Funnel Transformer: Trade-off

The funnel architecture offers a trade-off between latency and quality. The diagram illustrates how compressing the sequence length at different layers (zero-indexed) affects both device time and PR-AUC.



Funnel Transformer: Results

For medium-sized models like a 2B -param model, the Funnel Transformer provides a good latency-quality compromise. For example, applying it at layer 12 results in a significant 40.44% drop in p50 latency and a significant 43.85% increase in QPS/Chip, while lowering MMTEB by just 1.03 points.

Model	Latency (VLP 2x2)						MMTEB
	QPS around TPU utilization 90%	TPU Utilization	QPS / Chip	p50 latency	p90 latency	p99 latency	Mean (Task)
Baseline: 2B_MAXALL	5.82	90.34	1.455	174.56	246.6	291.74	66.03
Funnel 12/36	8.37	83.01	2.093	103.96	152.91	184.5	65.00

Funnel Transformer: Research Questions

[Paper: Revisiting Funnel Transformers for Modern LLM Architectures with Comprehensive Ablations in Training and Inference Configurations]

Funnelling Strategy:
Develop a generalized strategy for selecting the proper Funnel parameters, e.g., strides.

Model Type Influence:
Does the Funnel transformer exhibit different performance characteristics in Dense versus MoE models?

Scaling Effects:
How does the Funnel behavior vary across different model sizes?

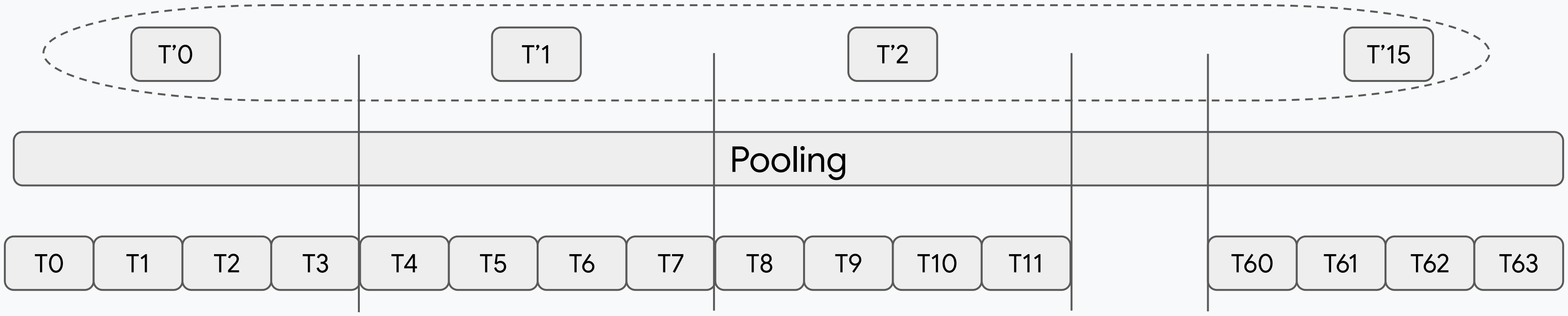
Pre-training Impact:
How does funnel-aware pre-training change Funnel performance (vs. post-training Funnel)?

Enhanced Decoding:
Research efficient decoding approaches towards restoring the full input length for token-level tasks.

Transient Global Attention

Transient Global Attention: Overview

The main idea of Transient Global Attention is to synthesize the global tokens on the fly (as aggregations of groups of tokens in the input), at each attention layer, resulting in noticeable drop in latency, especially with very large sequence lengths.



Transient Global Attention: Results

TGA provided significantly higher efficiency when applied with a long sequence length, without any drop in quality (MMTEB). Tested with a 2B-param model and a sequence length of 8192, TGA results in:

- 12.37% increase in QPS
- 9.99% drop in p50 latency
- 6.78% drop in p90 latency
- 8.93% drop in p99 latency

Model	***Latency (VLP 2x2) Sequence Length = 8192***						***MMTEB Mean (Task)***
	QPS around TPU utilization 90%	***TPU Utilization***	***QPS / Chip***	***p50 latency***	***P90 latency***	***p99 latency***	
Baseline: 2B_MAXALL	***5.82***	***90.34***	***1.455***	***174.56***	***246.6***	***291.74***	***66.03***
TGA	***6.54***	***90.29***	***1.635***	***157.13***	***229.87***	***267.27***	***66.07***

Thank you!

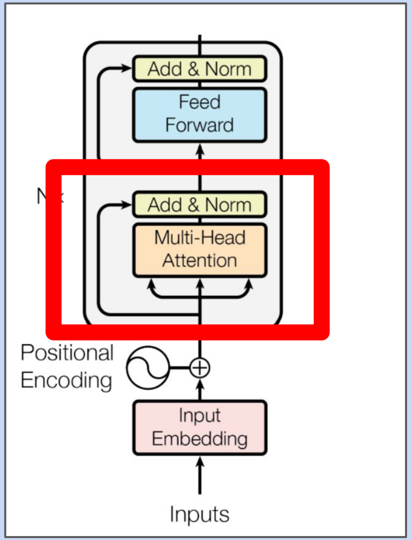
Architectural Sparsity:

Attention, Cascades, and FFN pruning

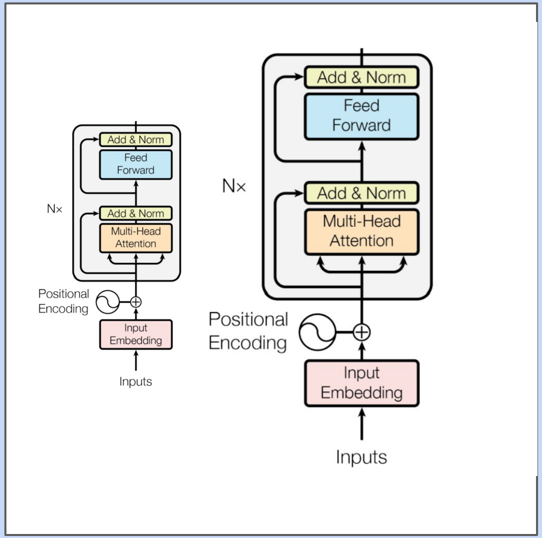
Lucas Spangher

Sections of this talk

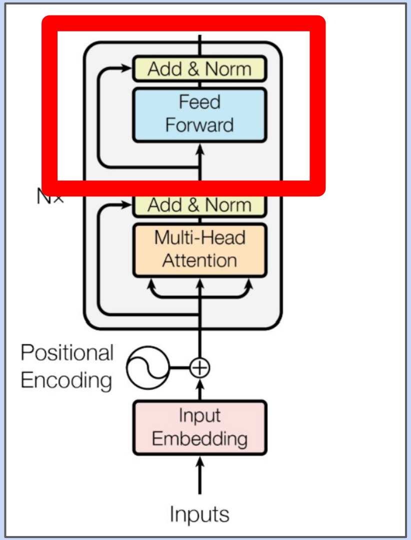
1. Attention Sparsity



2. Model Cascades

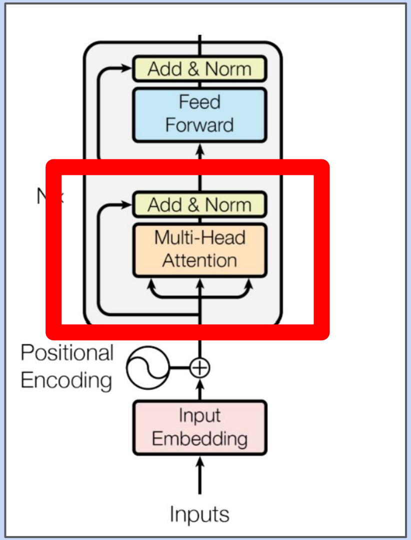


3. FFN Pruning

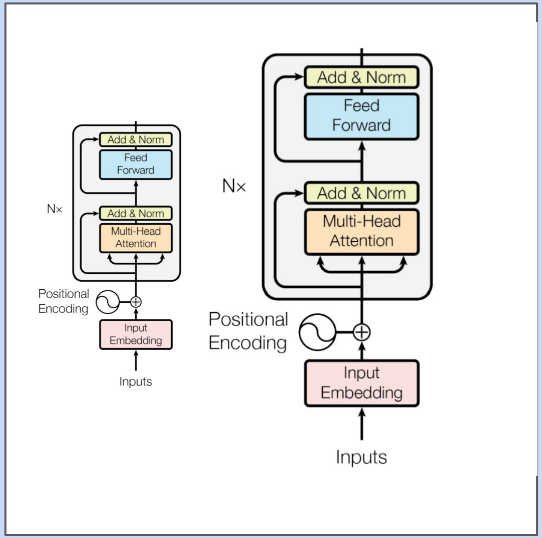


Sections of this talk

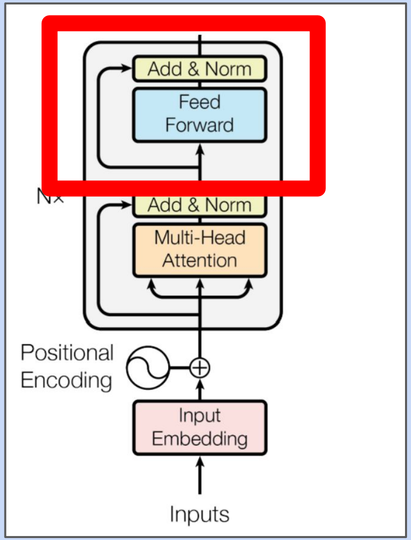
1. Attention Sparsity



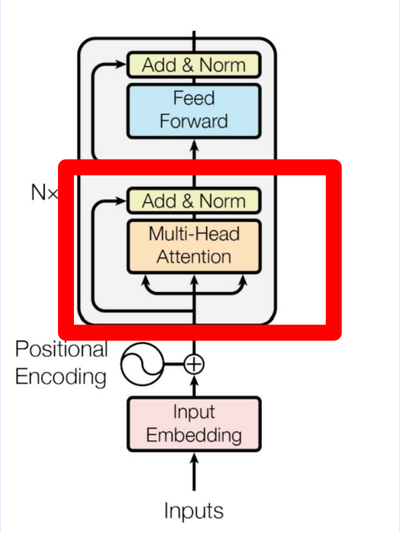
2. Model Cascades



3. FFN Pruning



1. Attention (is sometimes more than you need)



Outline:

- Introduction – attention masks and KV cache
- Flash attentions
- Attention mask interpretation
- Sparse attention mechanisms
- LLM alternatives

Paper list:

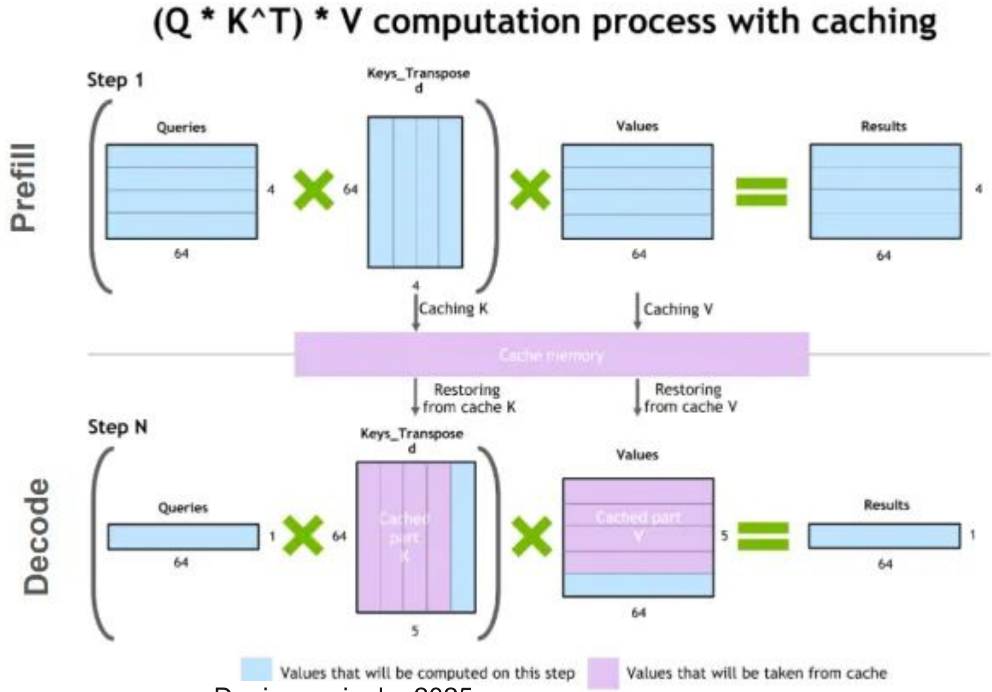
- Flash Attention, 2022
- Flash Attention 2, 2023
- DisruptionBench, 2025
- Sparse Transformer, 2019
- LongFormer, 2020
- Reformer, 2020
- Routing Transformer, 2021
- Autoformer, 2021

Intro: Vanilla Attention optimization with KV cache.

Attention mechanism: **KV cache**

- Separate prefill (population of the KV cache, $O(T^2)$) from decode (compute new K and V from query, grows w every token, $O(T)$)

- Global attention

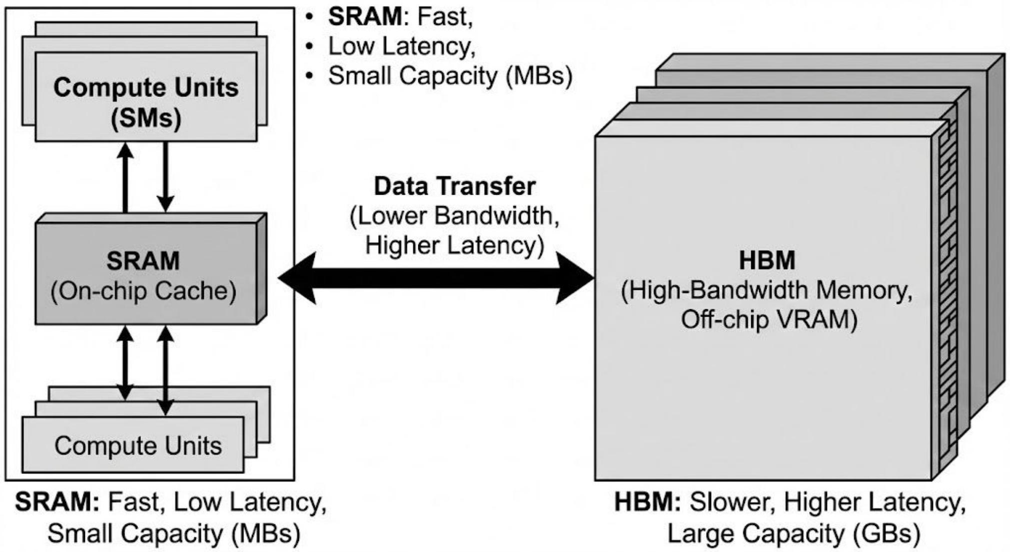


Degirmencioglu, 2025

Intro: Hardware Aware Era

Issue: Wasn't just O(N^2) compute that was slowing us down, it was O(N^2) memory access (HBM to SRAM).

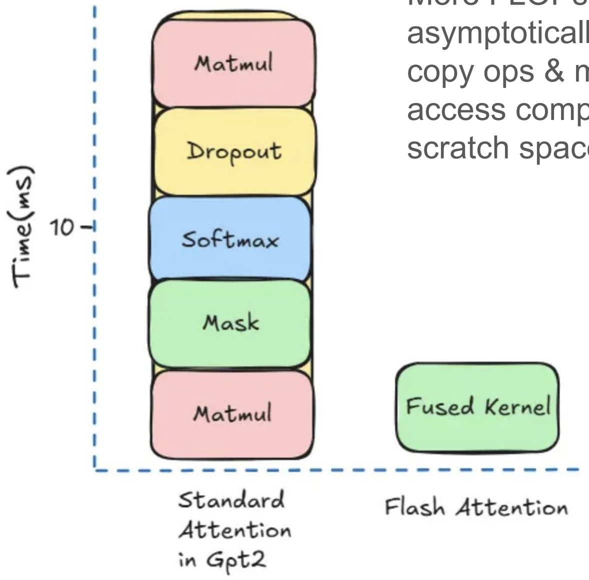
Standard Attention computes $S = QK^T$ (NxN) to High Bandwidth Memory (HBM), reads it back to apply Softmax, writes it again, and reads it again to multiply by V.



The Hardware Aware Era

Flash Attention (togetherAI)

1) **Tiling (Block-wise Computation):** It loads blocks of *Q, K, V* into SRAM, computes the attention scores for that specific block, updates the output, and discards the raw scores.
2) **Online Softmax:** (based on the Safe Softmax trick), keeps running statistics (max and sum) for each block. As it processes new blocks, it rescales the previous partial results to match the new global max.

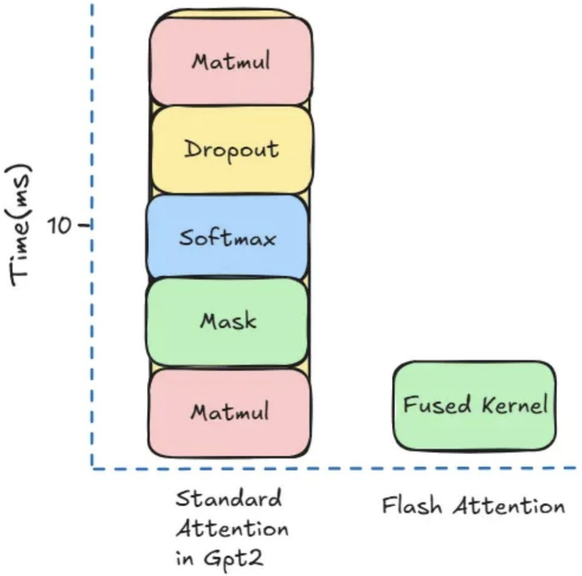


More FLOPs, but asymptotically smaller copy ops & memory access comp. Get rid of scratch space.

The Hardware Aware Era

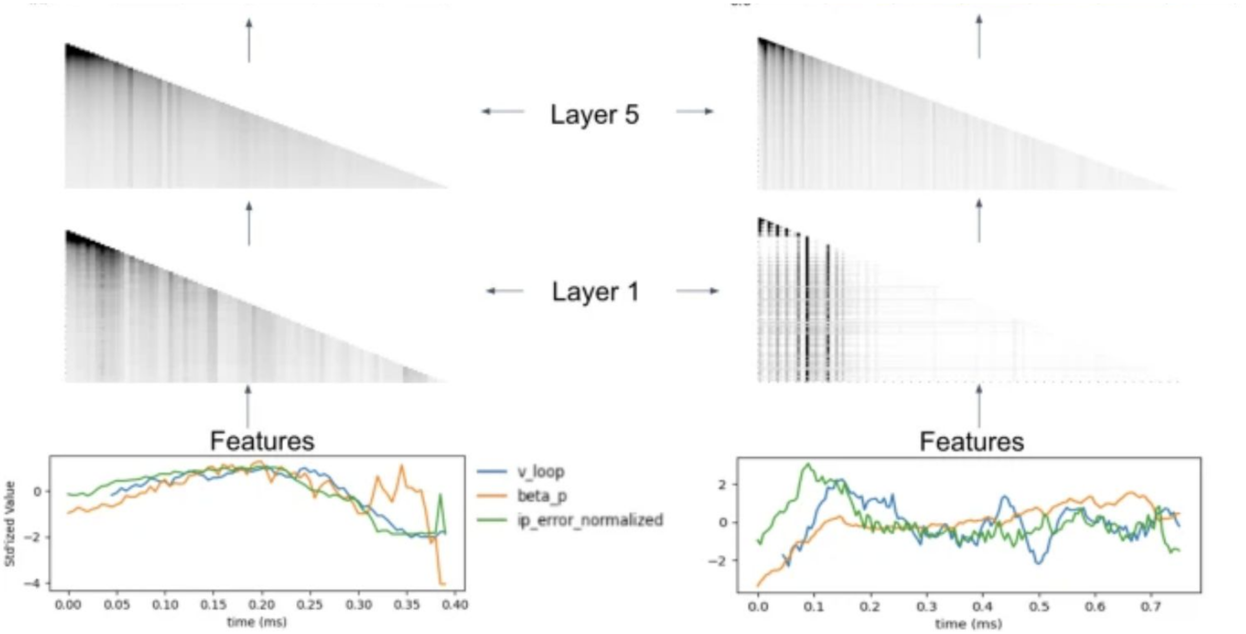
Flash Attention 2 (2023)

1) **Parallelizes over sequence length** in addition to heads (instead of just batch and heads.) Saturates GPU with batch=1.
2) **Reduces non-mat mul FLOPS** (i.e. sum, exp, div)
3) (Hardware specific) – **Work partitioning (warp)**



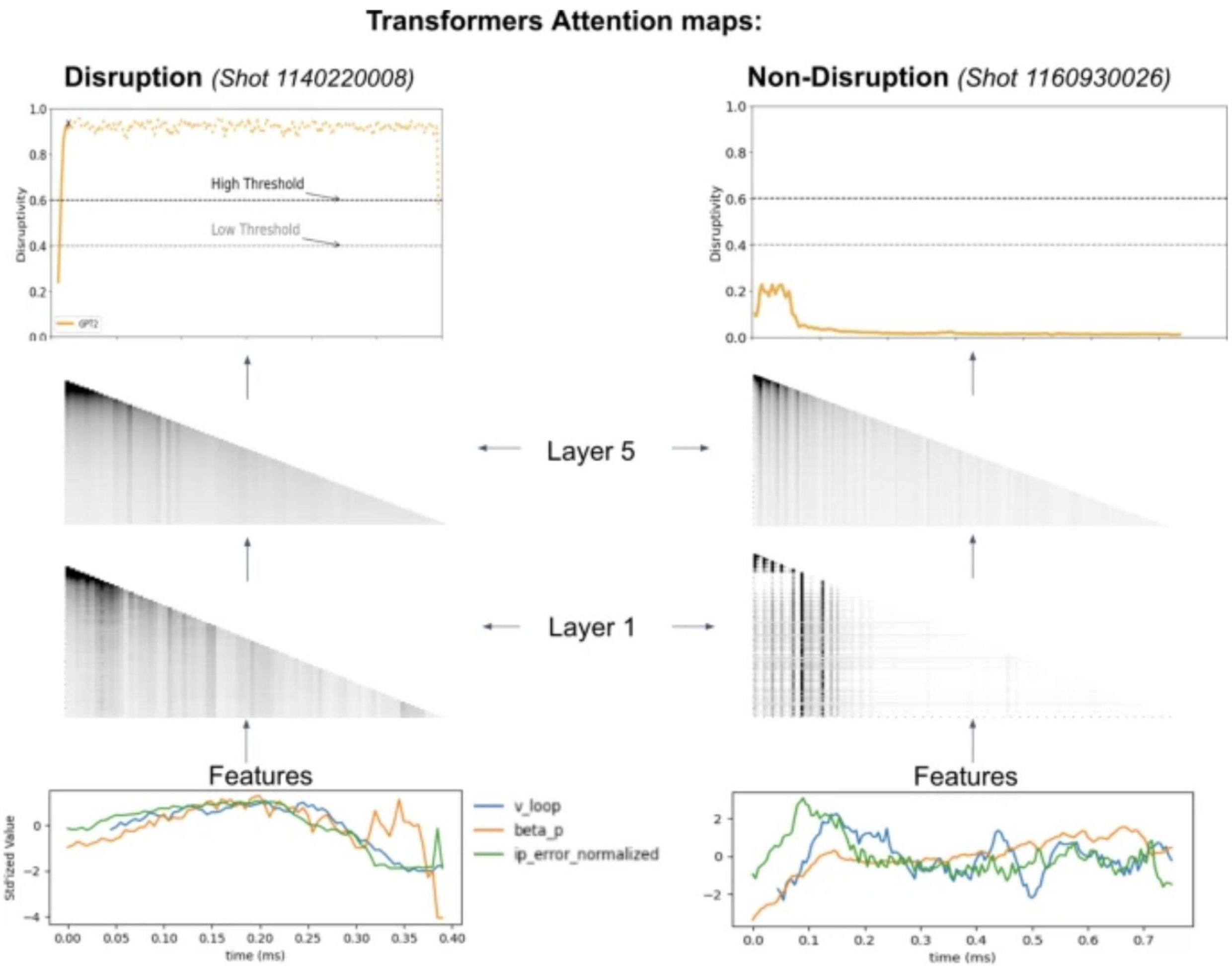
Interpretation: One can look at the attention matrix of the lower levels for model intuition

(Spangher, 2023)



One can look at the attention matrix of the lower levels for model intuition

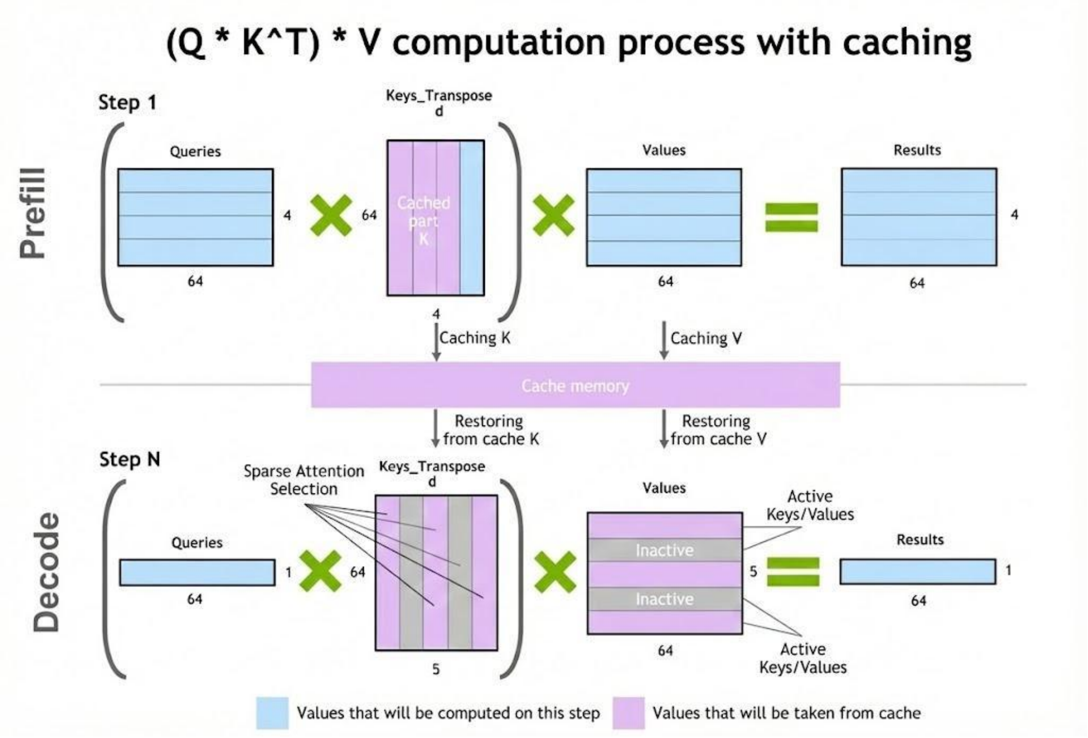
(Spangher, 2023)



Interpretation based improvements: The “window” era: fixed and local sparsity

Concept: Limit windows to local neighborhoods or strided patterns

Sparse Transformer (Child, 2019)

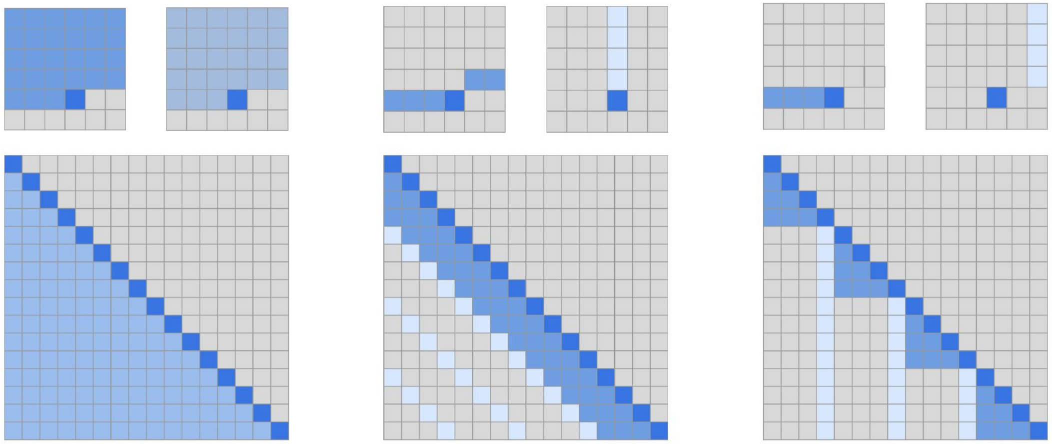


The “window” era: fixed and local sparsity

Concept: Limit windows to local neighborhoods or strided patterns

Sparse Transformer (Child, 2019)

Problems: context fragmentation. Difficulty passing sequence start info to later layers.



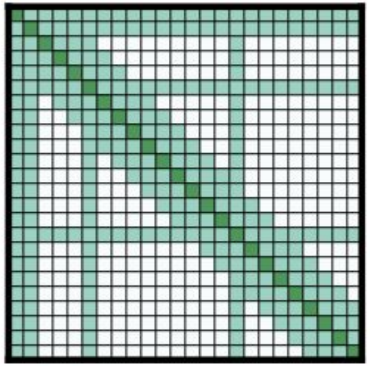
(a) Transformer (b) Sparse Transformer (strided) (c) Sparse Transformer (fixed)

Figure 3. Two 2d factorized attention schemes we evaluated in comparison to the full attention of a standard Transformer (a). The top row indicates, for an example 6x6 image, which positions two attention heads receive as input when computing a given output. The bottom row shows the connectivity matrix (not to scale) between all such outputs (rows) and inputs (columns). Sparsity in the connectivity matrix can lead to significantly faster computation. In (b) and (c), full connectivity between elements is preserved when the two heads are computed sequentially. We tested whether such factorizations could match in performance the rich connectivity patterns of Figure 2.

The “window” era: fixed and local sparsity

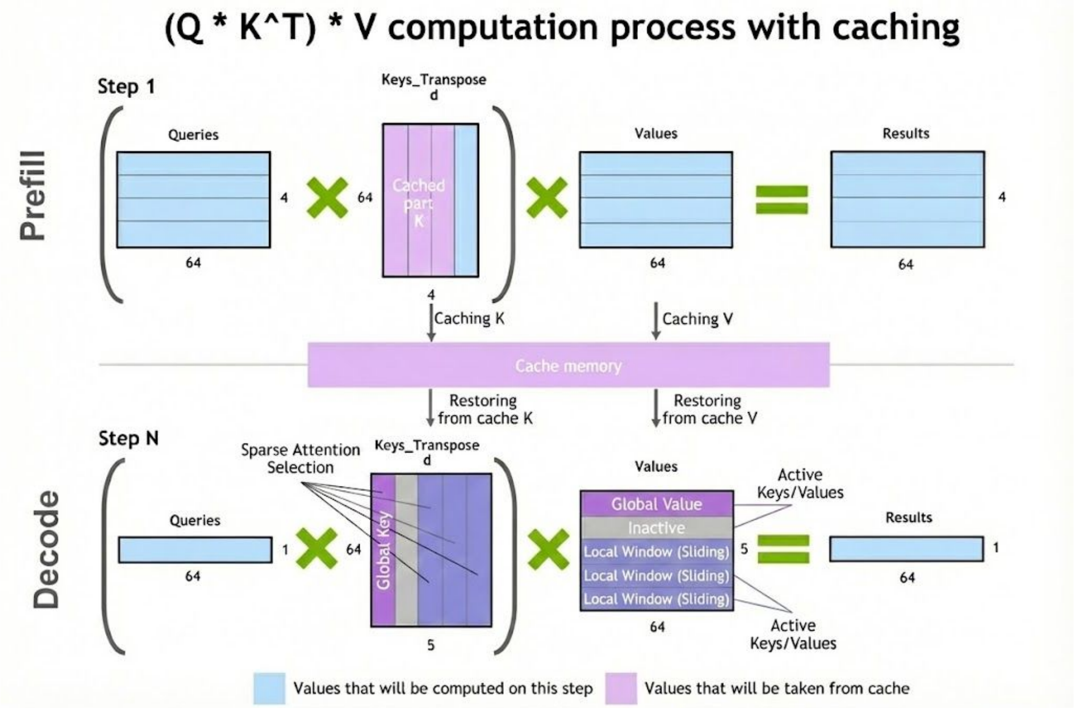
Concept: Limit windows to global and local neighborhoods or strided patterns

Longformer (2020)



(d) Global+sliding window

(Beltagy, 2020)



Content-Based and Learnable Sparsity

What if we only attend to important tokens based on query-key similarity?

Reformer (2020) computed embedding “similarity hashes” of similar words

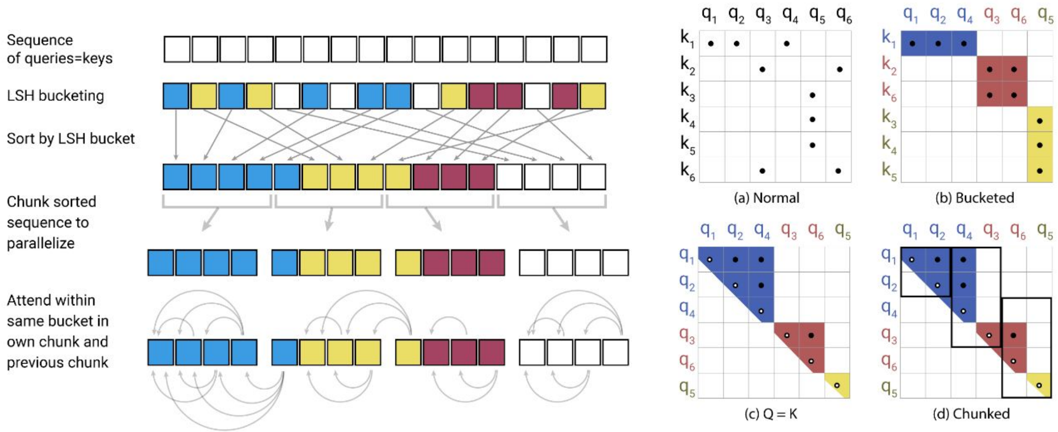
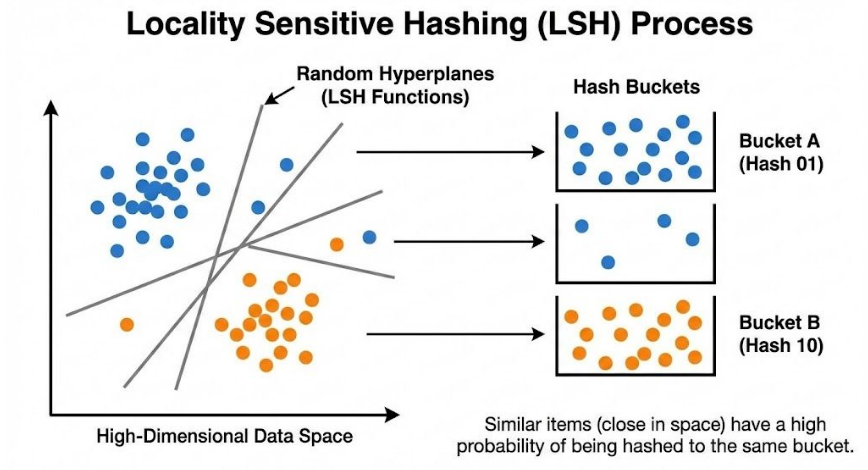


Figure 2: Simplified depiction of LSH Attention showing the hash-bucketing, sorting, and chunking steps and the resulting causal attentions. (a-d) Attention matrices for these varieties of attention.

Reformer – locally sensitive hashing

LSH – “Locally Sensitive Hashing” – an indexing. Since softmax dominates the output, we only care about the highest q . k pairs. Closeness in the projection on random high dimensional planes is used as heuristic for similarity.



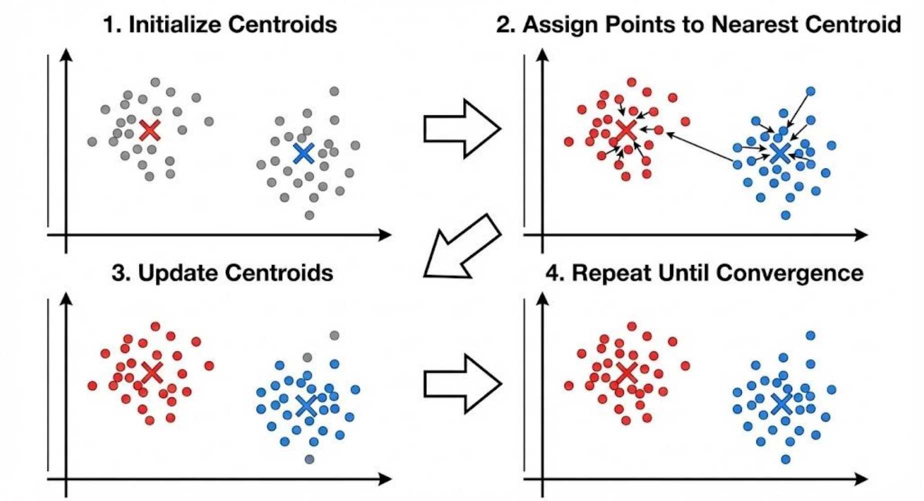
(dumb, fast SVD)
(standard algo in retrievals)

Huge breakthrough, but the possibility of missed hashes causes error.

Routing transformer (2021)

What if we only attend to important tokens based on query-key similarity?

Routing Transformer (2021) solves the hash misses by learning the clustering in k-means matching.



Miscellaneous Note: Attention variants to tend only to the type of attention that you need

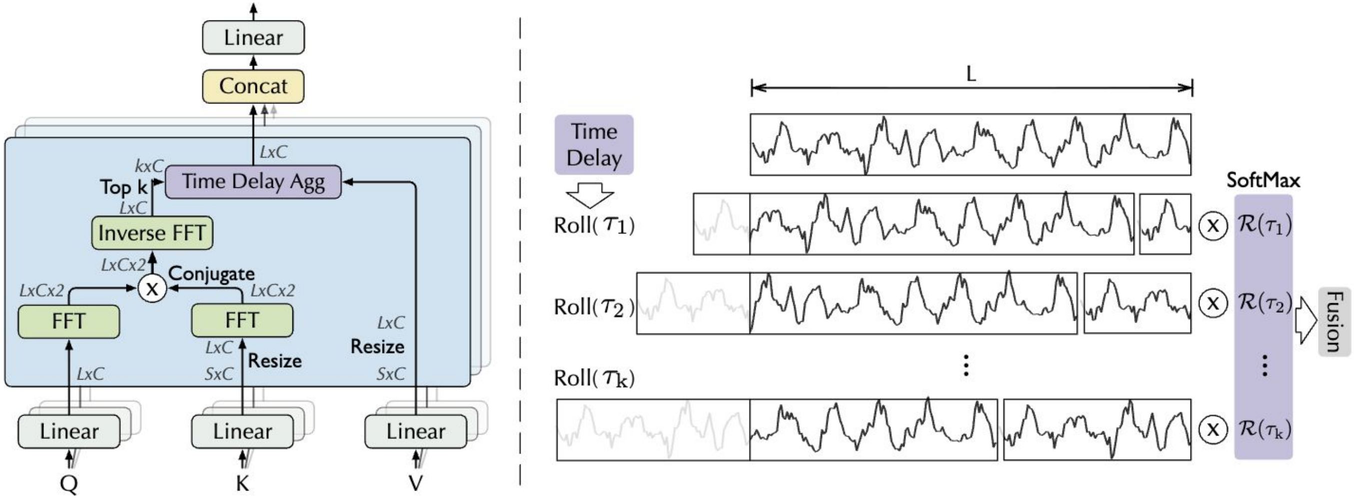
Before modeling, ask yourself:

- Dataset composition? Dataset size - O(100M)?
- Modeling needs – does it truly need an LLM?
- Is your time-series truly discrete, or is it continuous?

Autoformer – the time series transformer

Introduces blocks:

Autocorrelation: Time delay:



Autoformer – the time series transformer

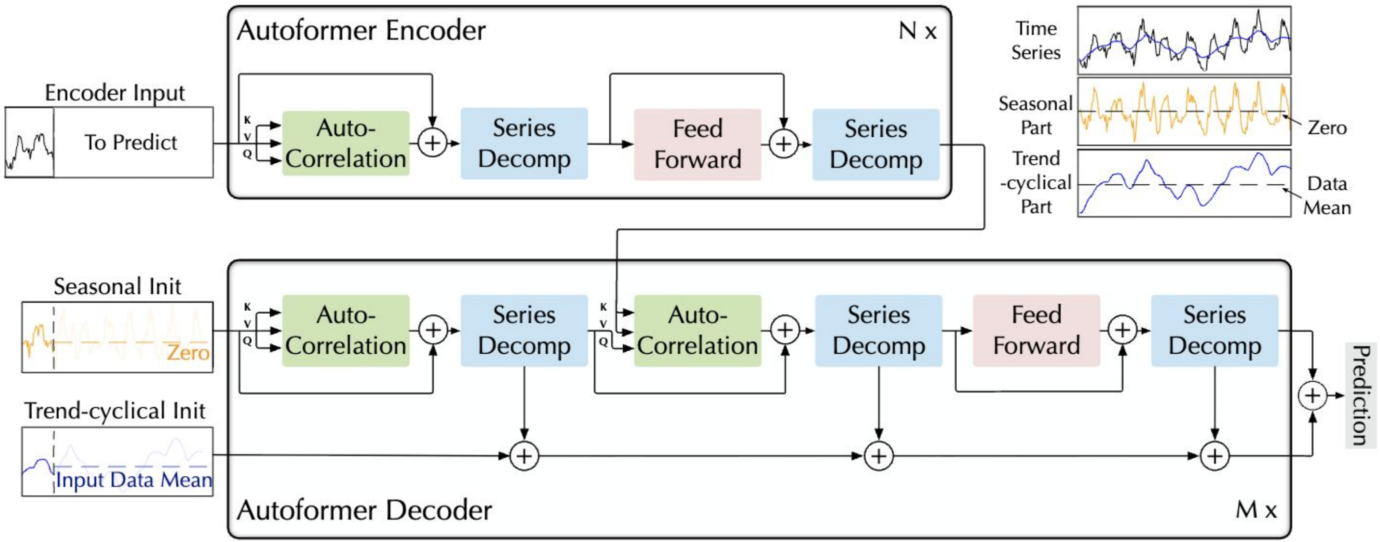
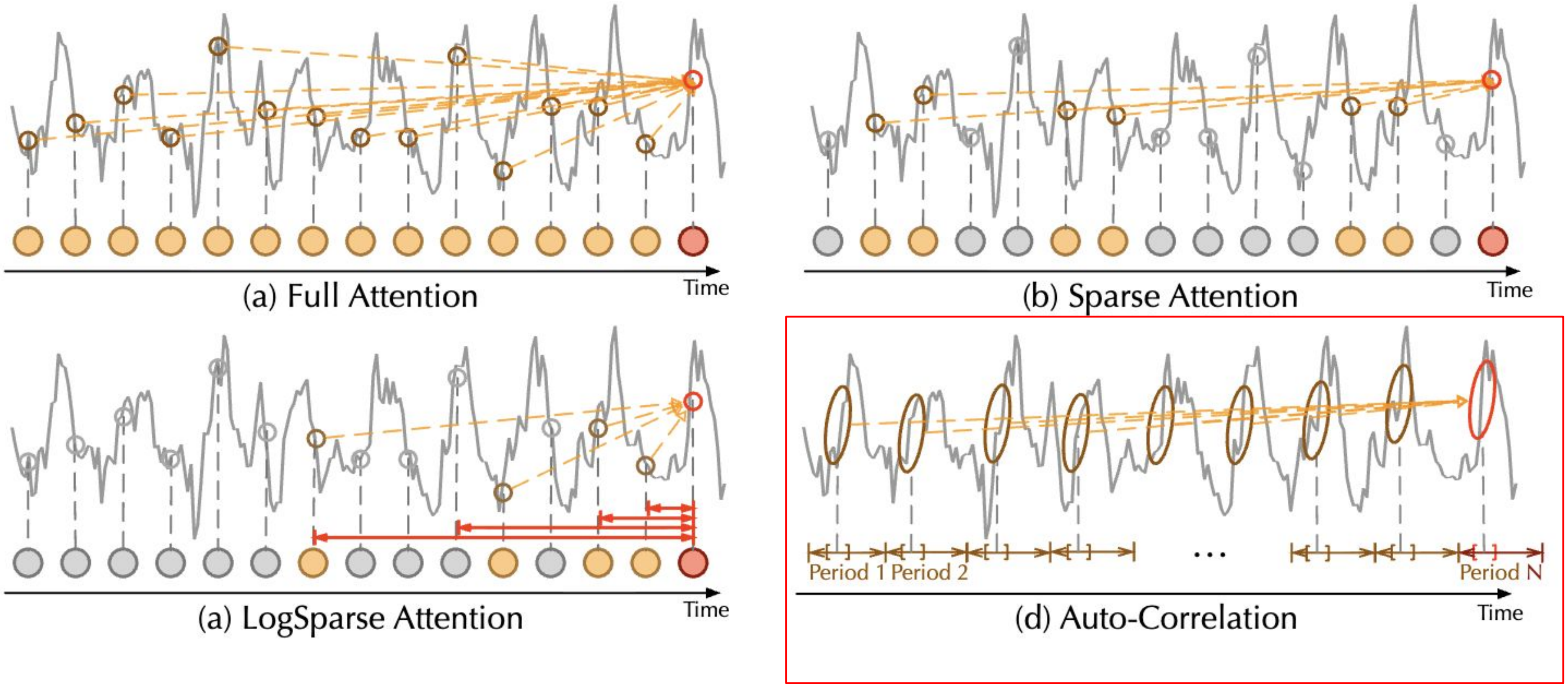


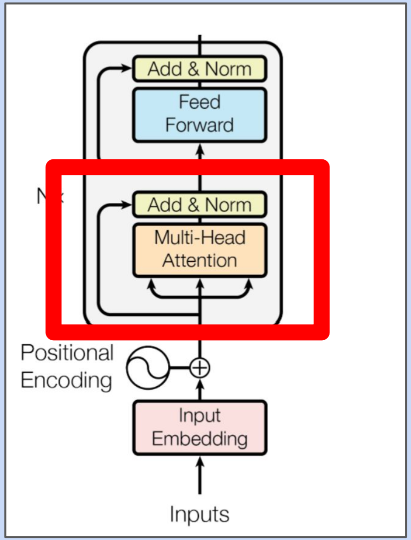
Figure 1: Autoformer architecture. The encoder eliminates the long-term trend-cyclical part by series decomposition blocks (blue blocks) and focuses on seasonal patterns modeling. The decoder accumulates the trend part extracted from hidden variables progressively. The past seasonal information from encoder is utilized by the encoder-decoder Auto-Correlation (center green block in decoder).

Autoformer – the time series transformer

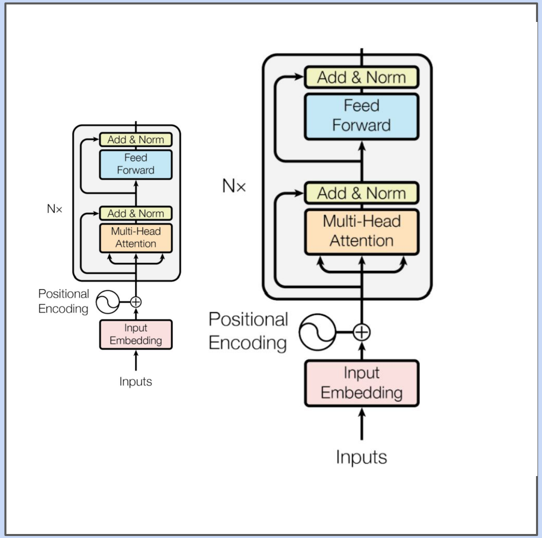


Sections of this talk

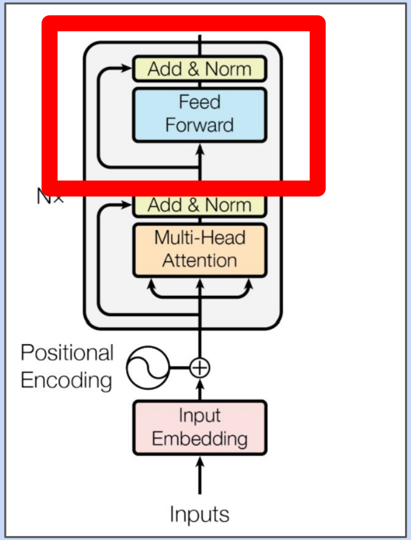
1. Attention Sparsity



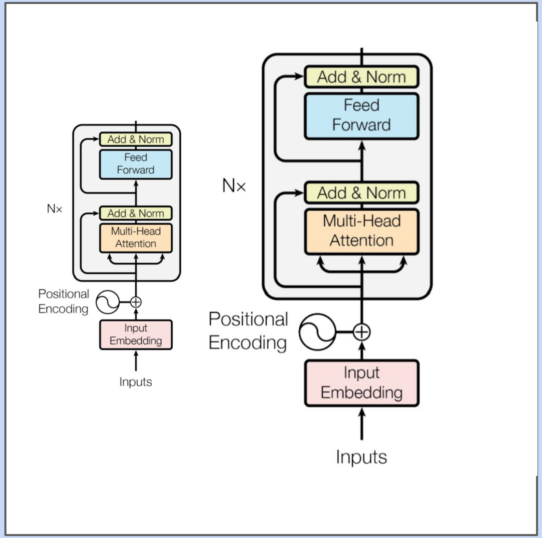
2. Model Cascades



3. FFN Pruning



2. Model Cascades



Outline

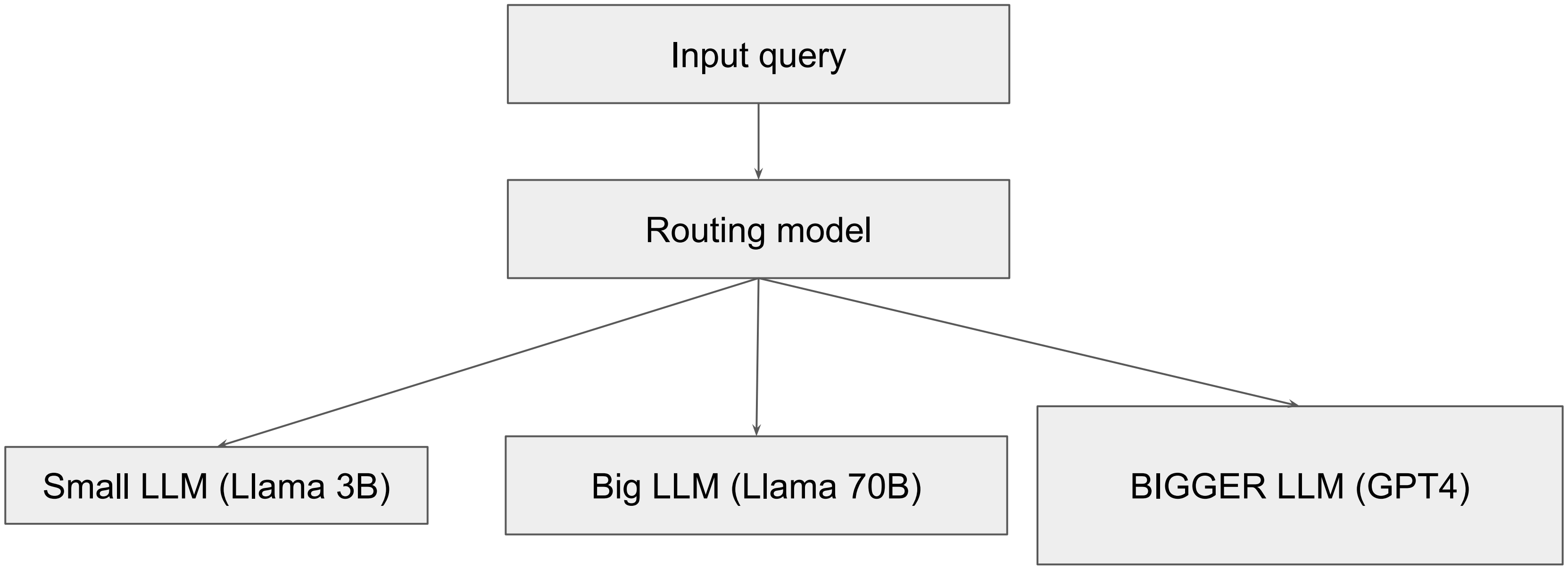
- Routing Cascades
- Sequential Cascades
- Early Exit Cascades

Paper list

- RouteLLM, 2024
- FrugalGPT, 2025
- BranchyNet, 2017
- Early Exit Networks, 2024

Routing Model Cascades

RouteLLM: Learning to Route LLMs with Preference Data (Ong et al., 2024).



Sequential Cascades

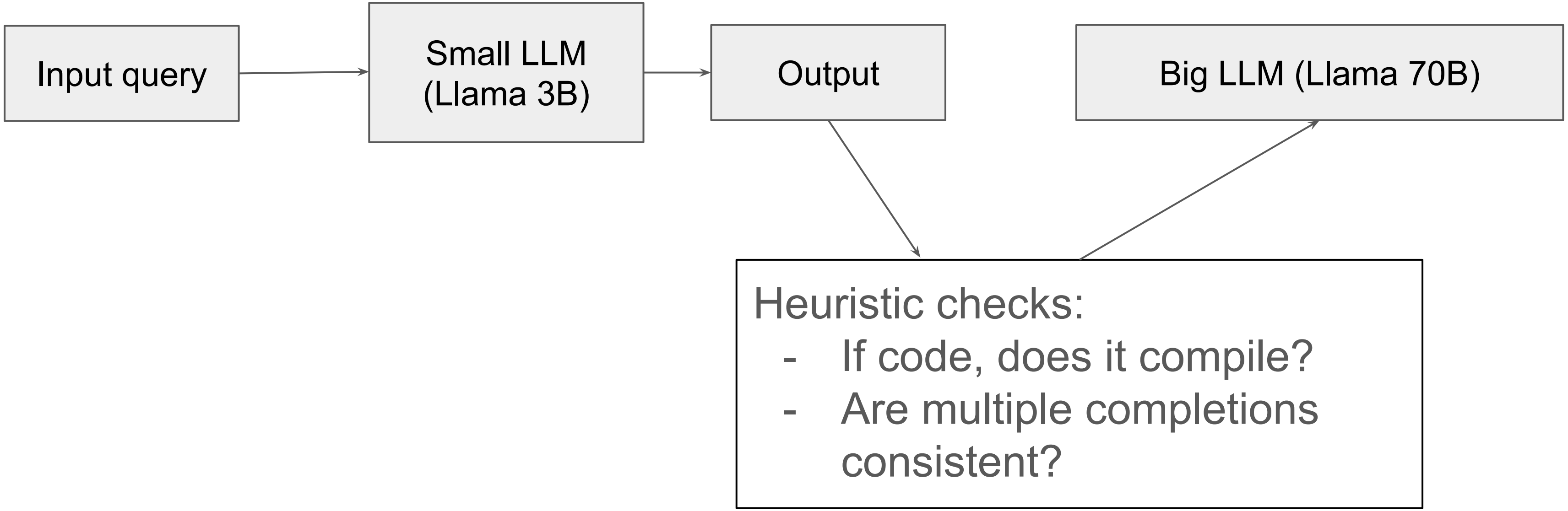
FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance (Chen et al., 2023).

Mechanism: Sequential calling:

(1) Call a cheap model
(2) Evaluate the response quality
(3) Threshold: if the score < tau, call expensive model

Early Exit Model Cascades

FrugalGPT (early exits)



Early Exit Model Cascades

BranchyNet or Early-Exit Networks (e.g., DeeBERT).

Attach small "classification heads" to intermediate layers of a Transformer (e.g., after Layer 4, Layer 12, and Layer 24).

Inference: If the head at Layer 4 is 99% confident (high softmax entropy), stop the forward pass there.

The winner?

1. Routing cascades?
2. System cascade?
3. Early exits?

(hold up a number of fingers pls)

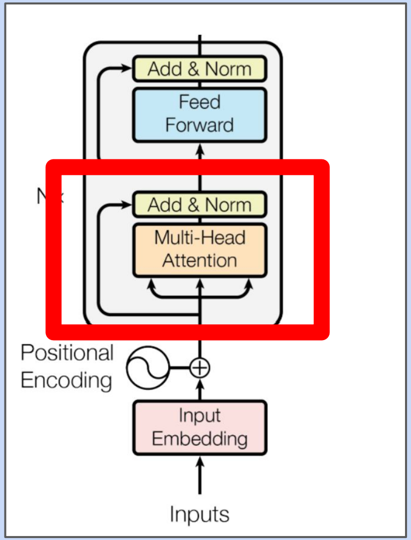
The winner?

1. Routing cascades
2. **System cascade**
3. Early exits

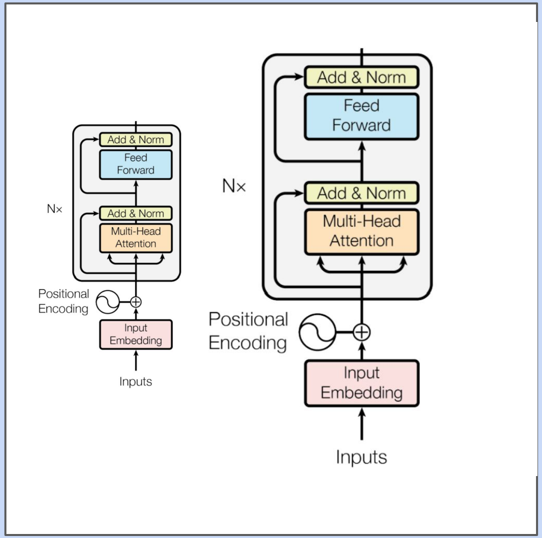
- Routing is actually pretty hard (active research)
- Early exits are tough to manage across batches (active research.)

Sections of this talk

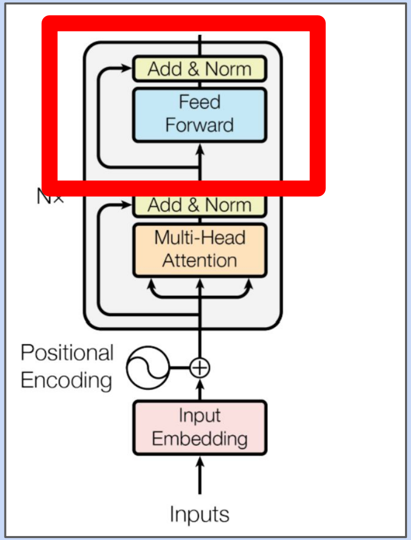
1. Attention Sparsity



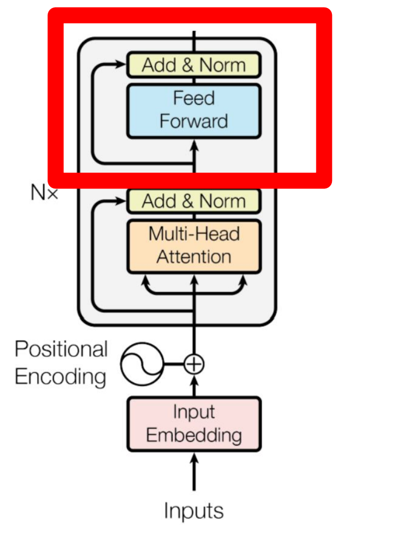
2. Model Cascades



3. FFN Pruning



2. FFN Pruning



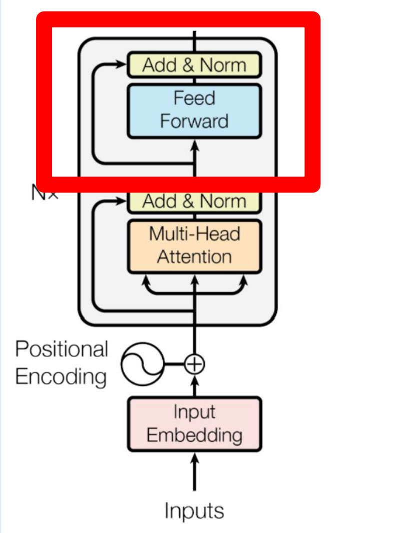
Outline

- Architecture intro
- Pruning intro
- Matryoshka embeddings
- Setup for Alejandro's talk

Paper list

- Structured Pruning, 2020
- Lottery Ticket, 2020
- BranchyNet, 2017
- Early Exit Networks, 2024

What are we pruning?



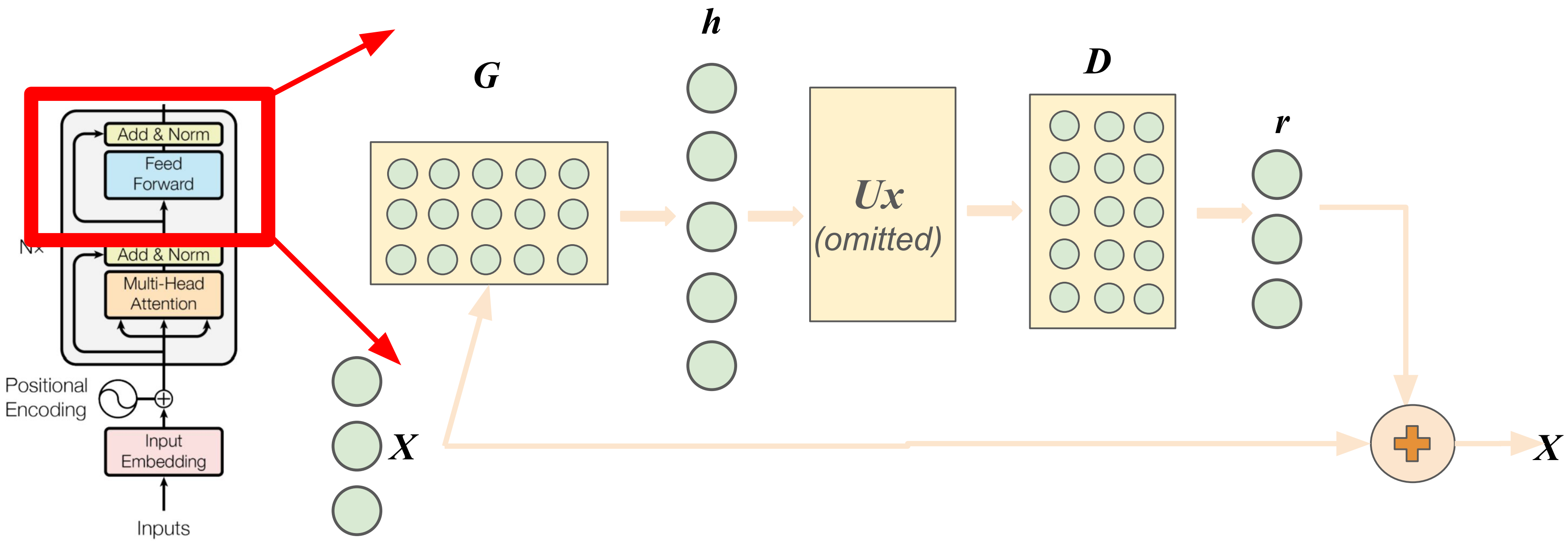
Feed forward networks in LLMs account for 50% of computation cost conservatively.

This is a juicy part of model that can be trimmed! (But not the only part)

An overview of pruning techniques

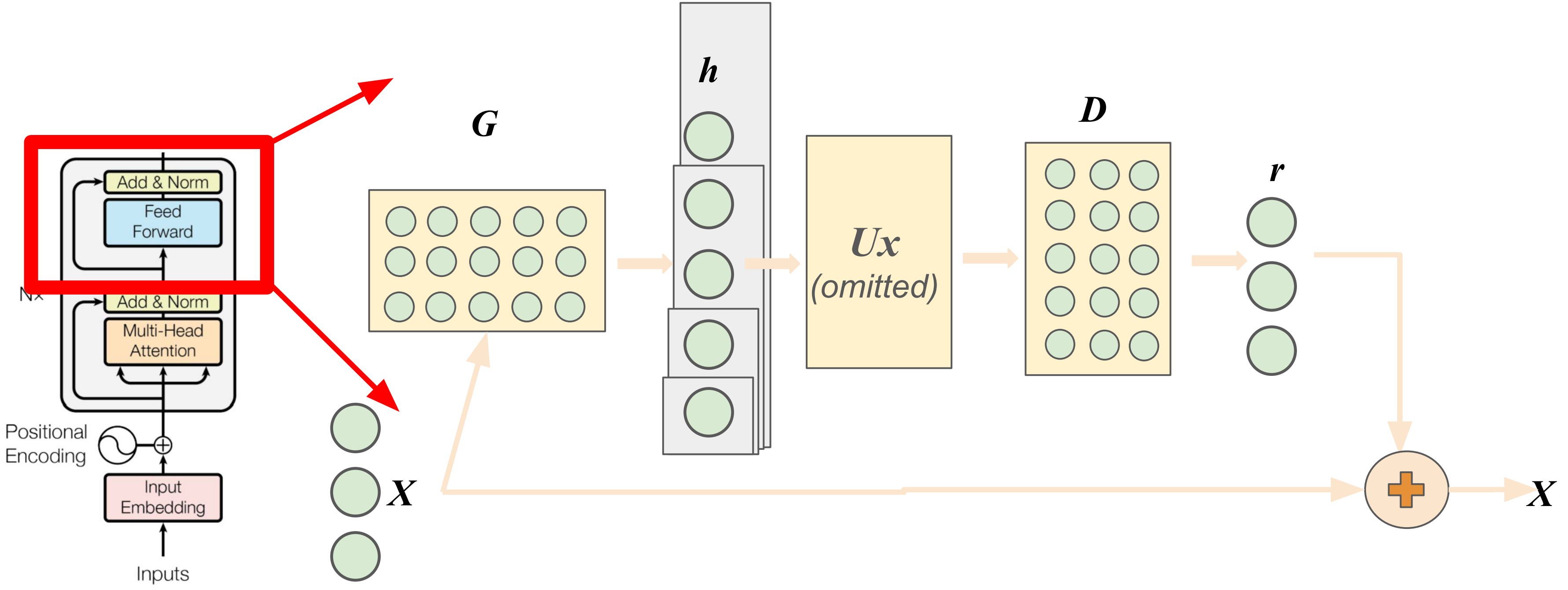
- Structured pruning
  - Block pruning (Structured Pruning, 2020 EMNLP)
  - Layer, head pruning
  - Lottery Tickets (2020) demonstrated that sparse subnetworks are activated per prompt.
  - SliceGPT (2024) downprojects weight matrices onto PCA components, rotating them to take the top k.
- Unstructured pruning
  - Optimal Brain Damage (Lecun, 1989) pruned FFN weights based on their saliency as determined by the Hessian (assumed diagonal hessian)
  - Optimal Brain Surgery (Hassibi, 1993) used an inverse Hessian
  - ShearedLlama (downprojects a larger model onto a pre-determined smaller size using constrained optimization.)

A (somewhat standard) FFN block



The Gemma 3 Feedforward Blocks are GLUs. We want to decrease the width by finding unutilized neurons.

MatFormer: Matryoshka Embeddings



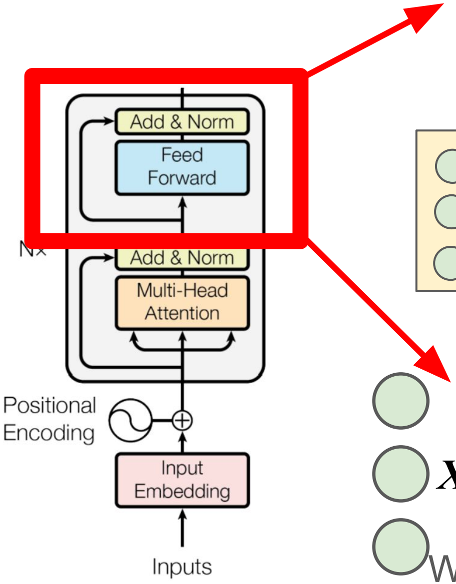
Train the model concurrently with increasing h-dim sizes; forces the model to be good at concentrating info in the smallest. **Infer** sequentially.

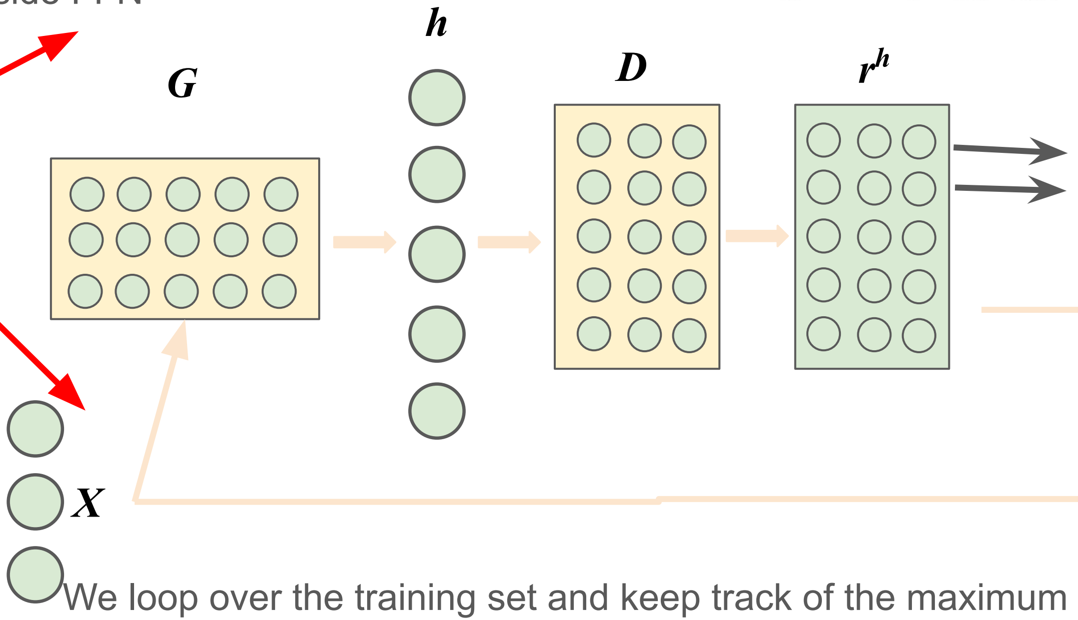
How do we prune?

Break residual contribution by neurons *h:*

$$r_i^h = \sigma(G_{hj}x_j)U_{hj}x_jD_{ih}$$

Pruning inside FFN





Compute norms for each neuron's residual contribution

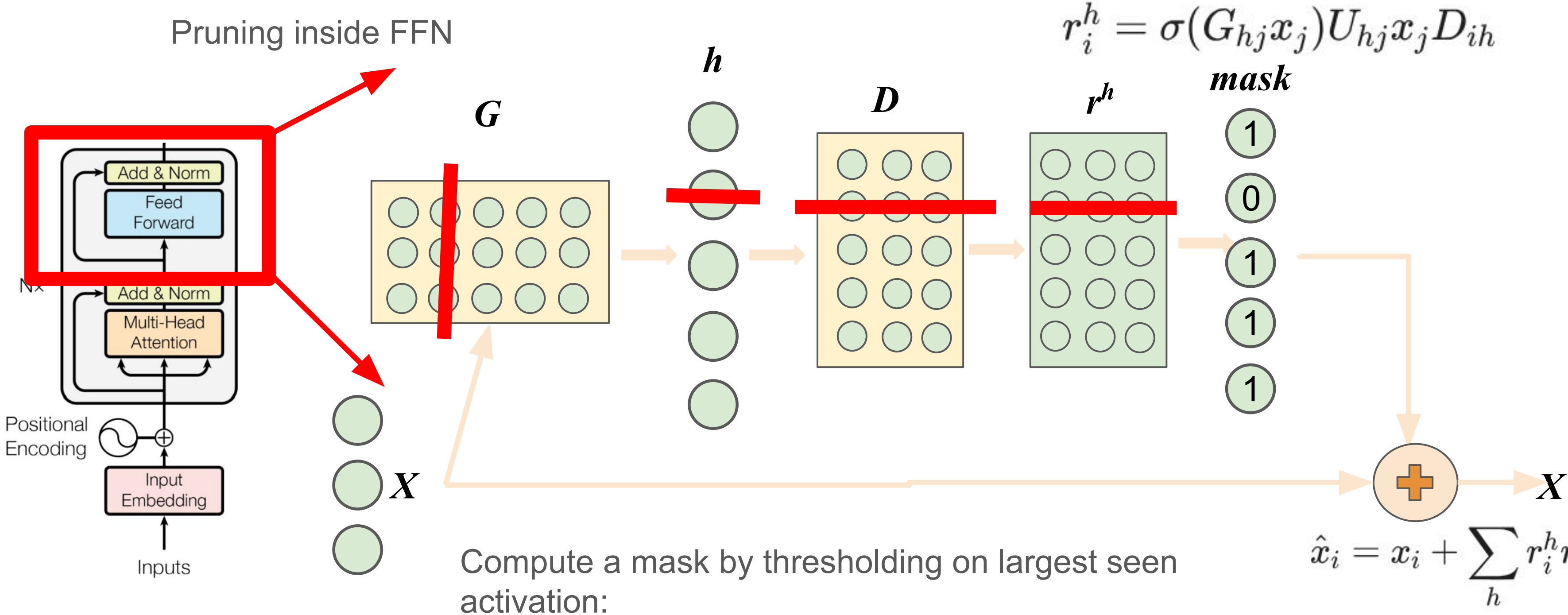
We loop over the training set and keep track of the maximum seen contribution from every neuron:

$$r^h_{max} = \max_{x \in \text{Train Set}} \|r_i^h\|_2$$

How do we prune?

Applying a mask deletes width and parameter matrix rows

$$r_i^h = \sigma(G_{hj}x_j)U_{hj}x_jD_{ih}$$

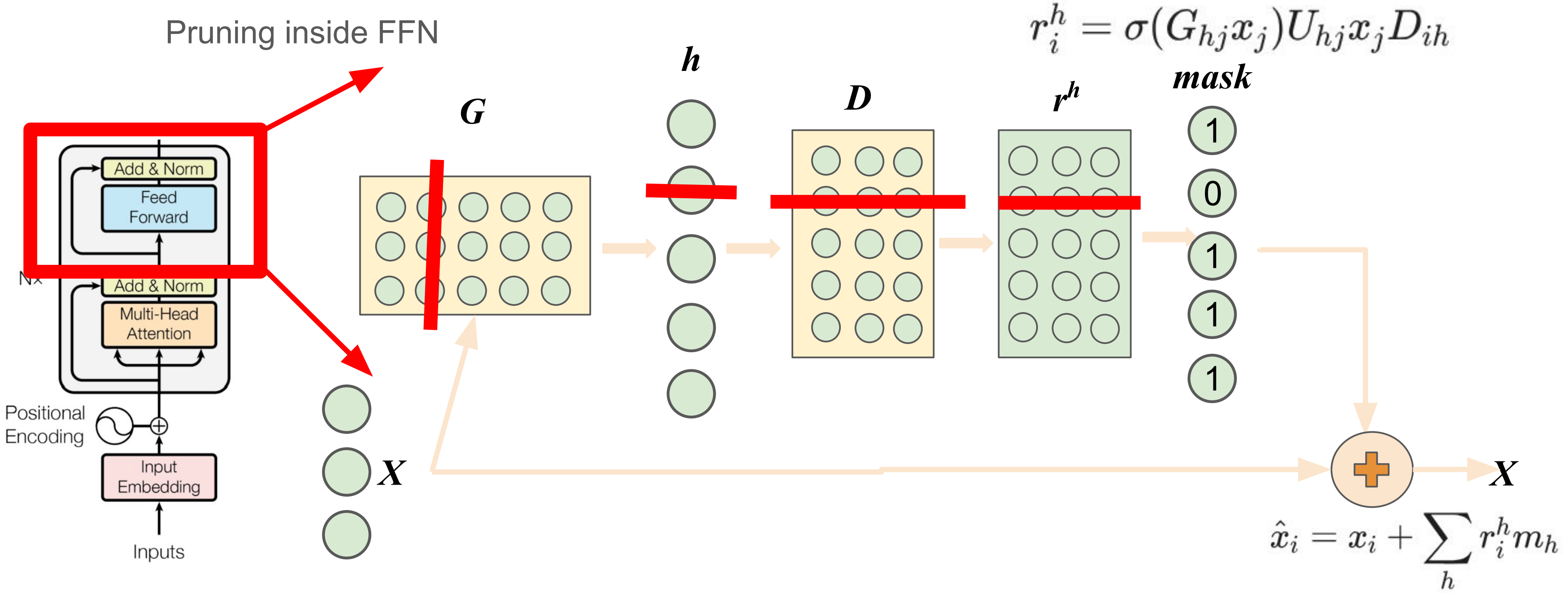


$$\hat{x}_i = x_i + \sum_h r_i^h m_h$$

Compute a mask by thresholding on largest seen activation:

$$mask_h = \begin{cases} 1; & r_{max}^h > c \text{ (keep neuron)} \\ 0; & r_{max}^h \leq c \text{ (delete neuron)} \end{cases}$$

How do we prune?



Most similar to **LLM-Pruner: On the Structural Pruning of Large Language Models** (Ma et al., NeurIPS 2023)

And now we'll show you this pruning….

In an "importance" selection method.

Feel free to get in touch!

Lucas Spangher

spangher@google.com

lucas_spangher@berkeley.edu

Google, MIT.

Interest in LLM efficiency, RL for green energy, and nuclear fusion disruption detection.

My apologies – at this moment I can’t accommodate requests for interns, job referrals.

Live Demo of Neuron Pruning in Gemma 3:

Alejandro F Queiruga

Feel free to get in touch!

Alejandro Quieruga

afq@google.com

Google, UC Berkeley.

Interest in LLM tinkering of all sorts, robotic hands, ML theory, AI for science, advertising.

Also can’t accommodate interns or referrals

And now on to the interactive portion…

We've prepared an interactive colab:

Neuron Pruning in Gemma 3 Demo

http://bit.ly/4oMp6B3

Colab Link

Appendix

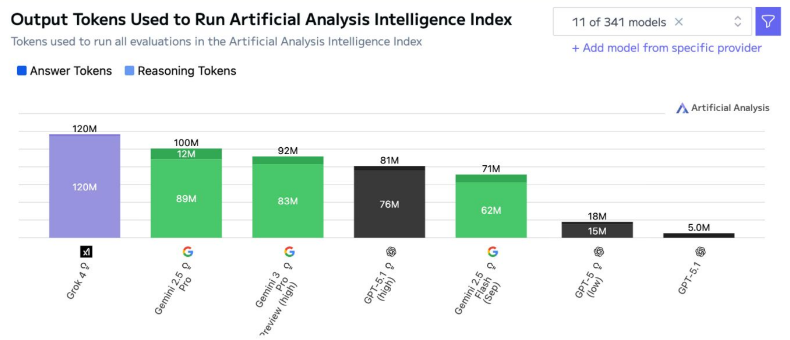
01

Appendix

Model Layer: Post-Training Examples

Example 1: Verbosity

The **verbosity** of a model has a dramatic effect on efficiency



Source: Artificial Analysis

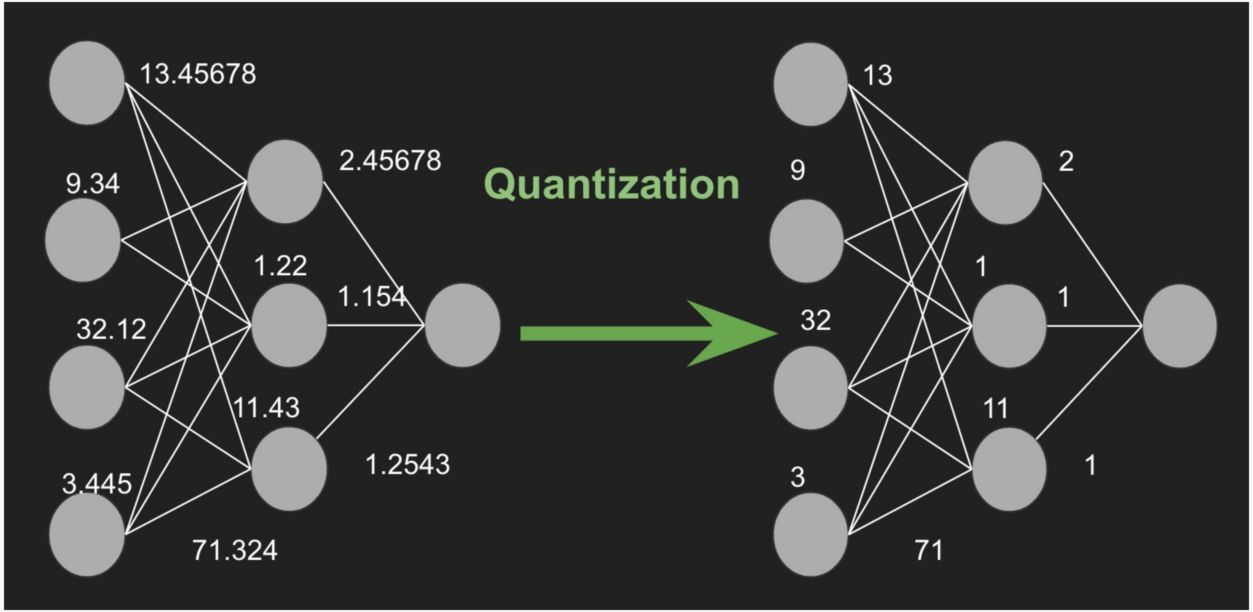
Example 2: Specialized Models

Cursor trained a specialized "fast apply" model

- Surpasses the performance of GPT-4 and GPT-4o,
- Achieves speeds of ~1000 tokens/s

CURSOR

Model Layer: Quantization



Source: [Digital Ocean]

Measuring contest: planned ~1GW scale clusters coming online in 2026

- Cluster size increasingly becomes a defining trait amongst American labs, particularly useful during recruitment. Should valuations follow cluster size instead of adoption or fiscal metrics, a larger bubble could begin to form.

GW Scale Cluster Rankings

Code Name	IT Power at YE 2026	Chip Type	Number of Chips	Total TFLOPS	Provider
xAI - Colossus 2	**1,200 MW**	GB200/300	550,000	3,488,148,649	xAI
Meta - Prometheus	**1,020 MW**	GB200/300	500,000	3,171,044,226	Meta
OpenAI - Stargate	**880 MW**	GB200/300	400,000	2,469,594,595	Oracle
Anthropic - Project Rainier	**780 MW**	Tranium 2	800,000	1,040,000,000	AWS

*Google DeepMind has also spun up many noteworthy clusters in Iowa, Nebraska, and Ohio. However, the distributed nature of these projects and lack of available information led to this omittance from the table.

Companies are racing to develop 5GW datacenters

Cluster Name	Company	Location	Capacity	Online Date
OpenAI Stargate (Multi-site)	OpenAI/SoftBank/Oracle	Multiple US sites (Texas, Ohio)	~10 GW	2028-2030
Microsoft Fairwater (Multi-site)	Microsoft	Wisconsin & Atlanta	2+ GW	2026-2028
Anthropic-Amazon New Carlisle	Anthropic/Amazon	New Carlisle, IN	2.2 GW	Jan 2026
Meta Hyperion	Meta	Richland Parish, LA	2 GW (Phase 1) 5 GW (full)	2030 (2 GW) Post-2030 (5 GW)
xAI Colossus 2	xAI	Memphis, TN	1.6+ GW	Feb 2026
Google Data Centers	Google	Iowa, Ohio, Texas	1+ GW	2026-2027

They cant! Demand is exceeding supply

“We almost always prioritize giving the GPUs to research over supporting the product...

We’re here to build AGI and research gets the priority.”

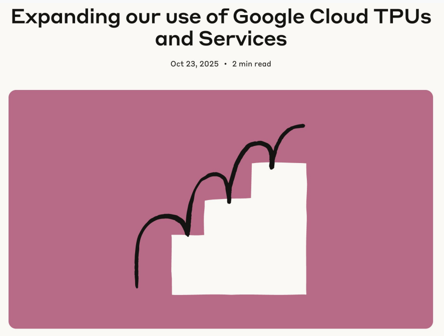
Demand is Soaring!

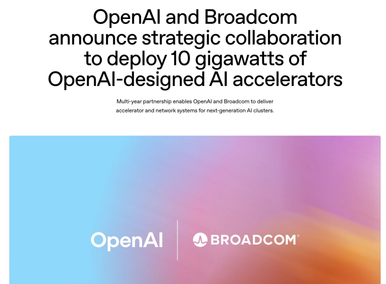
Google processed over 1,300,000,000,000,000 tokens in Oct 2025

That's 500M tokens a second or 1.8 Trillion tokens an hour

Motivating huge build outs

The Stargate Project is a new company which intends to invest $500 billion over the next four years building new AI infrastructure for OpenAI in the United States.





including up to one million TPUs, dramatically

AI energy consumption

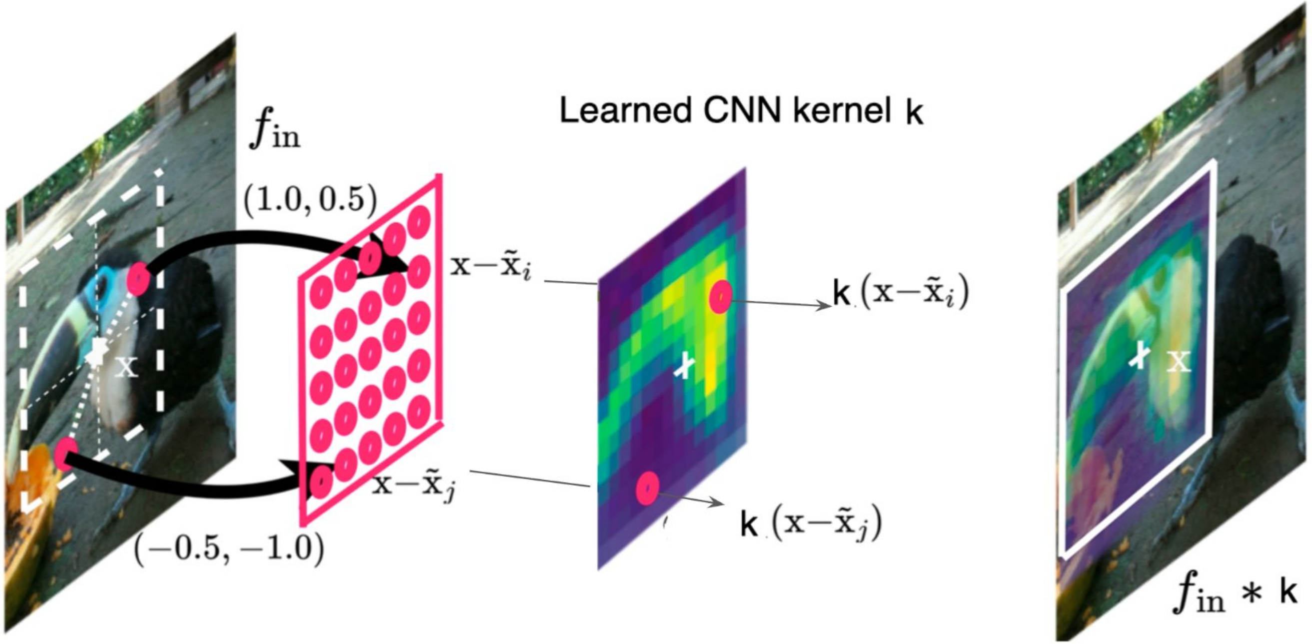
- https://kanoppi.co/search-engines-vs-ai-energy-consumption-compared/
- https://www.polytechnique-insights.com/en/columns/energy/generative-ai-energy-consumption-soars/
- https://www.polytechnique-insights.com/en/columns/energy/generative-ai-energy-consumption-soars/
-

Background

Abstract: Transformer architectures consume the lionshare of computational budgets associated with today's most powerful language and vision models, making research into greater computational efficiency a hot and essential direction. Our proposed tutorial surveys the bleeding edge of three complementary research threads that together comprise a significant part of the current industrial toolkit for achieving *computational efficiency* in Transformers: **(1) pruning**, the structured or unstructured removal of weights, layers and heads; **(2) sparse attention & routing**, including block, sliding-window, locality-sensitive hashing; and **(3) funneling**, which pools intermediate representations to shorten sequences through depth. We will then feature an expert industrial and academic panel of speakers from Caltech, MIT, Anthropic, Google Deepmind, and Microsoft, hearing about the latest trends seen in top industrial labs. Attendees will leave with actionable recipes for building sub-10 B-parameter models that match or exceed dense baselines on language, vision and multi-modal benchmarks.

Continuous Kernel CNN

Why is a classical CNN suited for discrete time-steps? The local filters are not resolution independent.



Continuous Kernel CNN

Network's goal: learn phi, a network that generates continuous kernels, which are then discretized for learning/inference.