Advanced Mathematical Topics in Modern Architectures

Mathematics of Deep Learning — Chapter 21

Yin Yang

Foundation Books Project

Roadmap

Running Example and Decision Audits

Measurement Vocabulary

Recurrence as State Compression

State-Space Models and Selective Scans

Gated Memory: LSTM to xLSTM

Efficient Attention and KV Caches

External Memory and Retrieval

Sparse Computation and Mixture of Experts

Residual Streams and Stable Depth

Inference-Time Algorithms

Worked Architecture Decision Audit

Summary

Running Example and Decision Audits

The Learning Assistant

A small learning assistant must answer:

> *Using the reference notes, explain why my model struggles on long sequences, compare it with a Transformer, and give one practice exercise.*

Constraints:

- Read or retrieve relevant notes from a fixed corpus.
- Stay accurate enough on a fixed set of 100 questions.
- Respond fast enough for live office-hour use.
- Cite the relevant evidence when possible.

The same tradeoffs appear in larger products; a small example keeps the main quantities computable by hand.

What Counts as an Architecture Claim

Advanced architecture claim
Names a fixed task, a baseline, the quantity that should improve, the cost that may get worse, and the changed mechanism (state update, attention pattern, routing rule, precision, decoding algorithm).

Not a claim: "Mamba is newer than a Transformer." A useful claim: "A selective state-space layer may reduce long-sequence memory pressure on our reference-note task while preserving answer quality."

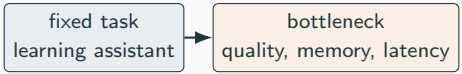
No mechanism, no bottleneck, no measurement $\Rightarrow$ no audit possible.

Decision-Audit Loop, Step by Step

fixed task
learning assistant

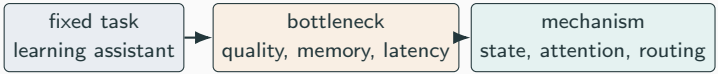
(1) Fix the task, dataset, prompts, scoring rules.

Decision-Audit Loop, Step by Step



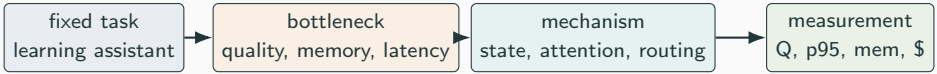
(1) Fix the task, dataset, prompts, scoring rules.
(2) Name the bottleneck before choosing a method.

Decision-Audit Loop, Step by Step



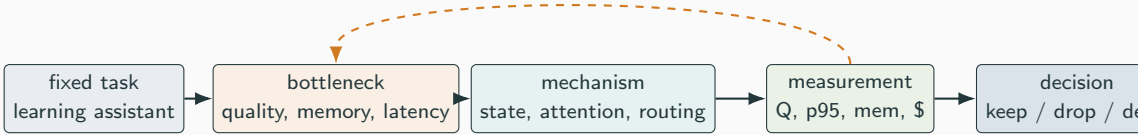
(1) Fix the task, dataset, prompts, scoring rules.
(2) Name the bottleneck before choosing a method.
(3) Identify the mathematical mechanism changed.

Decision-Audit Loop, Step by Step



(1) Fix the task, dataset, prompts, scoring rules.
(2) Name the bottleneck before choosing a method.
(3) Identify the mathematical mechanism changed.
(4) Measure quality and cost on the held-out set.

Decision-Audit Loop, Step by Step



(1) Fix the task, dataset, prompts, scoring rules.
(2) Name the bottleneck before choosing a method.
(3) Identify the mathematical mechanism changed.
(4) Measure quality and cost on the held-out set.
(5) Recommend keep, drop, or defer — with evidence.

A mechanism matters only after it improves a named bottleneck on a fixed task.

Measurement Vocabulary

Serving Token Budget

For one language-model request:

$$T_{\text{total}} = T_{\text{prompt}} + T_{\text{gen}}.$$

- Prompt: system instructions, user question, retrieved passages, formatting.
- Generated: tokens produced after the model starts answering.
- *Prefill* processes the prompt; *decode* produces tokens one step at a time.

Example: 700 instruction + 1,300 retrieved + 300 generated $\Rightarrow$ $T_{\text{prompt}} = 2{,}000$, $T_{\text{gen}} = 300$, $T_{\text{total}} = 2{,}300$.

Prefill and decode have different cost shapes — one report must show both.

Quality, Latency, Memory, Cost, Pareto

Quantity	What it asks	Reported as
Quality Q	Did it answer correctly?	Rubric score on held-out set
Median latency	Typical wait?	Seconds per request
p95 latency	How bad is the slow tail?	Seconds for the slow 5%
Memory	What must be stored/cached?	Weights + KV cache
Cost	What does good output cost?	$ or GPU-time per 1k good answers

Pareto comparison
A dominates B if A is no worse on every reported quantity and strictly better on at least one. Otherwise, neither dominates.

A comparison is not a ranking until the task says how the quantities trade off.

Pareto: One Concrete Tradeoff

Two candidate learning-assistant systems:

System	Q	Median latency	Memory
A	84/100	0.9 s	6 GB
B	88/100	3.1 s	10 GB

- B is better on quality.
- A is better on latency and memory.
- Neither dominates — the team must weigh four extra correct answers against 2.2 s extra latency and 4 GB.

"Better" depends on which axis the deployment cares about.

Recurrence as State Compression

State Update as a Compressed Summary

State update model

$$h_t = F_\theta(h_{t-1}, x_t), \qquad y_t = G_\theta(h_t),$$

with $x_t \in \mathbb{R}^{d_x}$ and $h_t \in \mathbb{R}^{d_h}$.

The state h_t is a *fixed-size* summary of the prefix $x_1, \ldots, x_t$, not the prefix itself.

- Useful when only some facts must persist (deadlines, policy).
- Risky when exact earlier wording is later needed for citation.
- Streaming: h_t computable from h_{t-1} and x_t alone.

Recurrence is not good or bad — it is a choice about what may be compressed.

How Far Old Tokens Survive

Scalar recurrence $h_t = ah_{t-1} + bx_t$, $h_0 = 0$:

$$h_4 = bx_4 + abx_3 + a^2bx_2 + a^3bx_1.$$

Coefficient of the oldest input x_1 in h_4:

a	Coefficient of x_1	Memory behavior
0.50	$0.125\,b$	Old input fades quickly
0.95	$0.857\,b$	Old input survives longer
> 1	grows	Repeated multiplication is unstable

The same constant controls long-range memory *and* stability — LSTMs replaced it with learned gates.

State-Space Models and Selective Scans

Discrete State-Space Layer

$$s_t = A_t s_{t-1} + B_t x_t, \qquad y_t = C_t s_t.$$

- $A_t \in \mathbb{R}^{d_s \times d_s}$: what is retained.
- $B_t \in \mathbb{R}^{d_s \times d_x}$: what new input enters.
- $C_t \in \mathbb{R}^{d_y \times d_s}$: how the state is read out.

Selective scan
Lets A_t, B_t, C_t depend on x_t. The words “deadline”, “optional”, “example” can affect memory differently.

Selectivity gives the layer a mathematical route to treat different tokens differently.

Affine Scan Composition

Write each update as an affine pair (A_t, u_t) with $u_t = B_t x_t$:

$$s_t = A_t s_{t-1} + u_t.$$

Composing two updates:

$$(A_2, u_2) \circ (A_1, u_1) = (A_2 A_1,\ A_2 u_1 + u_2).$$

Associative: blocks can be composed into larger blocks.

- More than for-loop notation: enables parallel prefix algorithms.
- Selective A_t, u_t still admit the scan structure.
- Hardware-aware implementation is part of the algorithm.

Associativity is what turns a recurrent loop into a parallel scan.

Pair Scores vs. Scan Updates

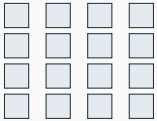
full attention



T^2 pair scores

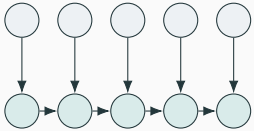
Pair Scores vs. Scan Updates

full attention



T^2 pair scores

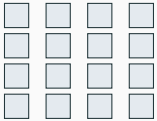
state scan



T recurrent updates

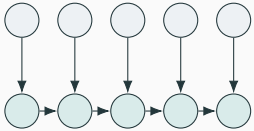
Pair Scores vs. Scan Updates

full attention



T^2 pair scores

state scan



T recurrent updates

$T = 1{,}000$: attention pairs $= 10^6$, scan updates $= 10^3$.

$T = 10{,}000$: attention pairs $= 10^8$, scan updates $= 10^4$.

Counts name the bottleneck; wall-clock time still depends on the hardware and kernel.

Gated Memory: LSTM to xLSTM

Gates: Learned Multiplicative Control

Gate
A learned quantity $g_t \in [0, 1]$ that scales information by multiplication: $g_t u_t$. Near 0 blocks, near 1 passes.

Forget-update memory

$$c_t = f_t \, c_{t-1} + i_t \, \widetilde{c}_t,$$

forget gate f_t, input gate i_t, candidate content $\widetilde{c}_t$.

Gates turn the single decay constant a of a plain recurrence into a data-dependent decision.

Remembering and forgetting become learned actions, not a fixed rate.

Two-Step Gated Memory

Scalar memory with $c_0 = 0$, $\widetilde{c}_1 = 10$, $\widetilde{c}_2 = 3$:

Step	Gates	Update	Result
1	$f_1 = 0,\ i_1 = 1$	$c_1 = 0 \cdot 0 + 1 \cdot 10$	$c_1 = 10$
2	$f_2 = 0.9,\ i_2 = 0.2$	$c_2 = 0.9 \cdot 10 + 0.2 \cdot 3$	$c_2 = 9.6$
alt	$f_2 = 0.2,\ i_2 = 0.2$	$c_2 = 0.2 \cdot 10 + 0.2 \cdot 3$	$c_2 = 2.6$

Same $\widetilde{c}_t$; different gates $\Rightarrow$ very different memory. xLSTM revisits gated memory at modern scale.

Gates fit sparse persistent info; exact citations remain hard for compressed memory alone.

Efficient Attention and KV Caches

Scaled Dot-Product Attention

For $Q \in \mathbb{R}^{T_q \times d_k}$, $K \in \mathbb{R}^{T_k \times d_k}$, $V \in \mathbb{R}^{T_k \times d_v}$:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V.$$

The matrix $QK^\top \in \mathbb{R}^{T_q \times T_k}$ holds pairwise scores.

- $T_q = T_k = 4$: 16 scores.
- $T = 8{,}000$: 64,000,000 scores per head per layer.

FlashAttention computes *exact* attention with better memory movement; sequence length, hardware, and latency still belong in the report.

Prompt processing and per-token decoding have separate cost shapes.

KV Cache: Storing Keys and Values

KV cache
During autoregressive decoding, the model stores previously computed key and value vectors so they need not be recomputed each step.

Stored scalar count:

$$N_{\mathrm{KV}} = 2\, L\, H_{\mathrm{kv}}\, T\, d_h.$$

L layers, H_{kv} key-value heads, T cached length, d_h per-head dimension. Factor 2 for keys *and* values. Bytes $= b \cdot N_{\mathrm{KV}}$ with b bytes per scalar.

Long context, more layers, or more KV-heads all multiply the serving footprint.

KV Cache Size: A Concrete Example

With $L = 24$, $H_{\text{kv}} = 8$, $T = 2{,}000$, $d_h = 64$:

$$N_{\text{KV}} = 2 \cdot 24 \cdot 8 \cdot 2{,}000 \cdot 64 = 49{,}152{,}000 \text{ scalars.}$$

At 2 bytes per scalar that is about 98 MB *per request*.

- Double $T \Rightarrow$ double the cache (linear in T).
- Multi-query attention: one KV head shared across query heads.
- Grouped-query attention: an intermediate sharing.
- Rotary position embeddings handle position by rotating coords.

Many simultaneous users multiply this footprint — p95 latency and memory pressure track together.

External Memory and Retrieval

Retrieval as External Memory

External memory
A stored collection of documents, chunks, embeddings, or tool outputs accessed at inference time — represented outside the network.

Top-k retrieval
With encoder e and score s,

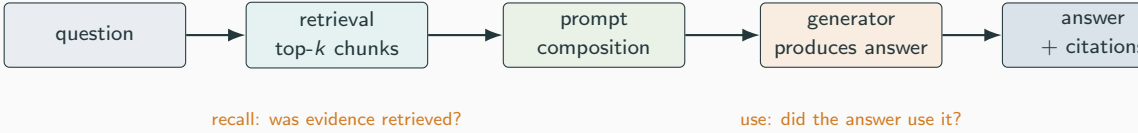
$$R(q) = \mathrm{TopK}_{d \in \mathcal{D}}\ s(e(q),\, e(d)).$$

Retrieved chunks are candidates for the prompt.

Cosine similarity, $u = (1,1)$, $v_1 = (1,0)$, $v_2 = (0,1)$, $v_3 = (2,2) \Rightarrow v_3$ nearest with similarity 1.

Retrieval is ordinary geometry applied to stored text vectors.

Retrieval vs. Generation: Audit Separately



- Retrieval recall: was the needed passage in top-k?
- Context budget: did it fit in T_{prompt}?
- Citation quality: does the answer point to the right source?

If the passage was never retrieved, changing the sequence layer solves the wrong problem.

Sparse Computation and Mixture of Experts

Total vs. Active Parameters

Dense vs. sparse
Dense layers use the same weights for every token. Sparse MoE layers have P_{total}, but per token use only $P_{\text{active}}(x_t) \ll P_{\text{total}}$.

Top-k expert routing
Router scores $r_\theta(x_t)$, selected set $S_t = \text{TopK}(r_\theta(x_t), k)$,

$$y_t = \sum_{e \in S_t} \alpha_{t,e}\, E_e(x_t).$$

MoE separates stored capacity from per-token compute — but a large P_{total} alone does not imply a better system.

Expert Calls and Load Balance, Step by Step

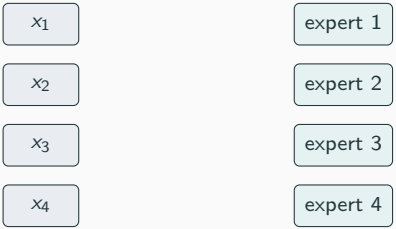
x_1

x_2

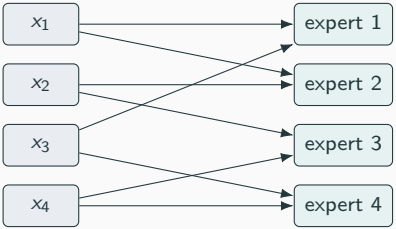
x_3

x_4

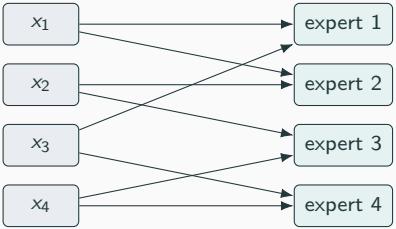
Expert Calls and Load Balance, Step by Step



Expert Calls and Load Balance, Step by Step

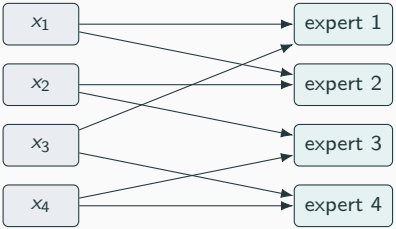


Expert Calls and Load Balance, Step by Step



Top-2 routing, T tokens:
total calls $= Tk$.
With E experts, average load $= Tk/E$.
Some expert gets at least $\lceil Tk/E \rceil$ calls.

Expert Calls and Load Balance, Step by Step



Top-2 routing, T tokens:
total calls $= Tk$.
With E experts, average load $= Tk/E$.
Some expert gets at least $\lceil Tk/E \rceil$ calls.

Example: $T = 1{,}000$, $k = 2 \Rightarrow 2{,}000$ calls.
Imbalance shows up as slow tail and idle hardware.

Report P_{total}, P_{active}, quality, p95 latency, and routing imbalance.

Residual Streams and Stable Depth

The Residual Stream

Residual stream
The sequence of representations carried through residual blocks. In a Transformer, attention and feed-forward sublayers *read from* and *write updates into* this stream.

Scalar example, 10 layers:

$$z_{\ell+1} = 0.9\, z_\ell \;\Rightarrow\; z_{10} \approx 0.35\, z_0,$$

$$z_{\ell+1} = 1.1\, z_\ell \;\Rightarrow\; z_{10} \approx 2.59\, z_0.$$

Neither is automatically good; depth changes magnitudes quickly.

Residual design, normalization, and initialization are the practical controls on this growth.

Pre-Normalization Block

A pre-normalization residual block:

$$z_{\ell+1} = z_\ell + F_\ell\big(\mathrm{Norm}(z_\ell)\big).$$

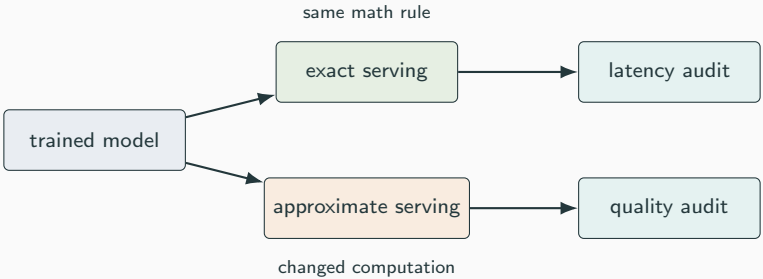
- Identity path lets later layers *modify* an existing state.
- Normalization controls the scale fed into F_ℓ.
- Initialization controls how fast accumulated updates change scale.
- Hyper-Connections / MHC explore richer connection patterns.

Learning assistant: evidence retrieved from notes must *survive* many transformations. A fluent but uncited answer can signal a representation that lost the source.

Deep composition is only useful when information can flow through it stably.

Inference-Time Algorithms

Exact vs. Approximate Acceleration



- Exact: FlashAttention, KV-cache reuse, batching of identical work.
- Approximate: quantization, low-rank, distillation.
- Speculative decoding: exact *under its acceptance rule*.

State which case applies before comparing numbers — approximate serving always needs a quality audit.

Quantization Error Bound

For a scalar quantizer that maps to a nearest grid value with step Δ:

$$|w_i - Q(w_i)| \le \frac{\Delta}{2}, \qquad E_Q(w) = \sum_{i=1}^{n} (w_i - Q(w_i))^2 \le \frac{n\Delta^2}{4}.$$

Example, grid $\{-1, 0, 1\}$:

w_i	$Q(w_i)$	Squared error
0.8	1	0.04
-0.2	0	0.04
-0.9	-1	0.01

Quantization reduces memory; calibration and rare-case failures must still be measured.

Speculative Decoding: One-Step Acceptance

Draft distribution q, target p. Draw $V \sim q$, accept with

$$a(V) = \min\left\{1,\ \frac{p(V)}{q(V)}\right\}.$$

On rejection, sample from residual $r(v) \propto \max\{p(v) - q(v),\ 0\}$. Final distribution: p.

Three-token vocabulary, $p = (0.50, 0.30, 0.20)$, $q = (0.40, 0.40, 0.20)$:

Token	$p(v)$	$q(v)$	accepted	residual
a	0.50	0.40	0.40	0.10
b	0.30	0.40	0.30	0.00
c	0.20	0.20	0.20	0.00

Accepted mass plus residual mass exactly recovers the target p — speculative decoding can be exact.

Batching: Throughput vs. Wait

Batching
Processes multiple requests together. Improves hardware utilization and throughput, but can increase per-request wait.

- Tokens/second can rise even as one user's wait rises.
- Office-hour peak: maybe accept queueing for throughput.
- Single live tutor: latency dominates.

Always report *both* throughput and latency. A method that looks excellent in tokens per second may still feel slow if users wait in a queue.

Throughput and latency are different axes of the same serving decision.

Worked Architecture Decision Audit

An Audit Table (Hypothetical Numbers)

Method	Q	Med.	p95	Mem./cost
Baseline Transformer	84	0.9 s	2.8 s	6 GB / 1.0x
Longer-context attention	88	1.4 s	5.9 s	13 GB / 1.7x
Retrieval + shorter context	87	1.0 s	3.1 s	7 GB / 1.2x
State-space / recurrent	82	0.7 s	2.0 s	5 GB / 0.8x
Serving optimization	84	0.6 s	1.7 s	6 GB / 0.7x

Same 100 validation questions, same corpus, same scoring rule.

No row dominates — each row tests a different bottleneck.

Reading the Audit, Step by Step

Baseline: reference numbers — no change.

Recommendation example:

Keep retrieval + shorter-context if it improves citation recall without breaching live-help p95. Defer longer-context attention until the quality gain is large enough to justify memory and tail latency.

Reading the Audit, Step by Step

Baseline: reference numbers — no change.

Longer context: +4 quality, but p95 doubles and memory 2×.

Recommendation example:

Keep retrieval + shorter-context if it improves citation recall without breaching live-help p95. Defer longer-context attention until the quality gain is large enough to justify memory and tail latency.

Reading the Audit, Step by Step

Baseline: reference numbers — no change.

Longer context: +4 quality, but p95 doubles and memory 2×.

Retrieval: +3 quality with small p95 hit — test if it solves recall.

Recommendation example:

Keep retrieval + shorter-context if it improves citation recall without breaching live-help p95. Defer longer-context attention until the quality gain is large enough to justify memory and tail latency.

Reading the Audit, Step by Step

Baseline: reference numbers — no change.

Longer context: $+4$ quality, but p95 doubles and memory $2\times$.

Retrieval: $+3$ quality with small p95 hit — test if it solves recall.

State-space: faster and smaller, but -2 quality.

Recommendation example:

Keep retrieval + shorter-context if it improves citation recall without breaching live-help p95. Defer longer-context attention until the quality gain is large enough to justify memory and tail latency.

Reading the Audit, Step by Step

Baseline: reference numbers — no change.

Longer context: +4 quality, but p95 doubles and memory 2×.

Retrieval: +3 quality with small p95 hit — test if it solves recall.

State-space: faster and smaller, but −2 quality.

Serving optimization: latency win, quality preserved (check it stayed exact).

Recommendation example:

Keep retrieval + shorter-context if it improves citation recall without breaching live-help p95. Defer longer-context attention until the quality gain is large enough to justify memory and tail latency.

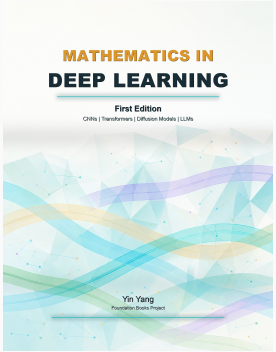
Wording is narrow on purpose — the evidence applies to this task, this set, this setup.

Summary

Chapter in One Slide

- **Measurement first:** Q, median & p95 latency, memory (incl. KV cache), cost. Pareto comparison, not ranking.
- **Recurrence & SSMs:** fixed-size state compresses the prefix; selective scans add input-dependent updates with cost $\sim T$ instead of T^2.
- **Gates:** learned multiplicative control of remember / forget — LSTM, xLSTM.
- **Efficient attention & KV cache:** $N_{\mathrm{KV}} = 2LH_{\mathrm{kv}}Td_h$; MQA / GQA share KV heads.
- **Retrieval:** external memory — audit recall before blaming the generator.
- **MoE:** P_{total} vs. P_{active}; Tk expert calls and load balance.
- **Residual streams:** let information flow through depth with controlled magnitude.
- **Inference algorithms:** exact (FlashAttention, speculative under acceptance rule, batching) vs. approximate (quantization).
- **Decision audit:** fix the task, name the bottleneck, identify the mechanism, measure, recommend narrowly.

An advanced architecture claim is useful when it names the mechanism, the bottleneck, the measurement, and the cost it accepts.



Mathematics in Deep Learning

Chapter overview slides for the book by Yin Yang.

Book page	`foundation-books.github.io/mathematics-in-deep-learning`
Code	`github.com/foundation-books/mathematics-in-deep-learning-code`
Paperback	`amazon.com/dp/B0H1ZZHDT2`
Hardcover	`amazon.com/dp/B0H1ZL11MJ`
Kindle	`amazon.com/dp/B0GX32M8SP`
License	CC BY-NC-ND 4.0

These free overview slides may be shared non-commercially with attribution and without modifications. Full explanations, exercises, and companion-code context are in the book and linked repository.