DeepSeek-V3 Pretraining

Deep Dive into DeepSeek — Chapter 6

Yin Yang

Foundation Books Project

The one rule of this chapter

The public V3 sources specify a training spine. They do not specify a training trace.

- Reported facts — 14.8T tokens, named schedules, FP8, a distributed topology — set the scale for every calculation in this chapter.
- The private corpus mixture, per-step token order, cluster scheduler, and incident history stay unknown, and nothing here fills them in with an invented default.
- A toy MoE training account — state, precision, parallelism, recovery — makes the mechanisms checkable by hand instead.

Goal: assemble the reported spine into one accounting structure, and keep every load-bearing number stamped by how it is known.

Roadmap

1. The public training spine

Roadmap

1. The public training spine
2. Token accounting and scaling laws

Roadmap

1. The public training spine
2. Token accounting and scaling laws
3. From architecture to a training-state ledger

Roadmap

1. The public training spine
2. Token accounting and scaling laws
3. From architecture to a training-state ledger
4. FP8: motivation and local definitions

Roadmap

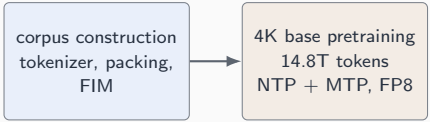
1. The public training spine
2. Token accounting and scaling laws
3. From architecture to a training-state ledger
4. FP8: motivation and local definitions
5. Parallelism, recovery, and the reported record

The public training spine

Four handoffs, one checkpoint chain

corpus construction
tokenizer, packing, FIM

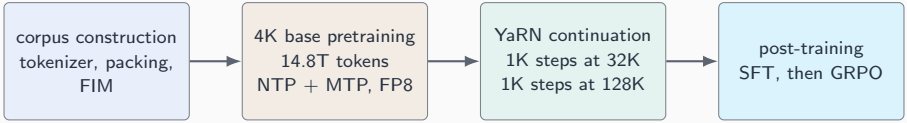
Four handoffs, one checkpoint chain



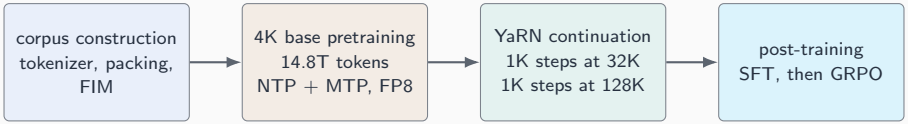
Four handoffs, one checkpoint chain



Four handoffs, one checkpoint chain



Four handoffs, one checkpoint chain



Every stage consumes the checkpoint the preceding stage produced.

[REPORTED: DeepSeek-V3 technical report]: this is the public spine. The private data pipeline, run trace, and incident history are not part of it.

The base run, by its reported numbers

Surface	Reported value
Training tokens	14.8T, next-token + one MTP depth
MTP weight	0.3 for first 10T tokens, then 0.1
Routing bias update	$\gamma = 0.001$ through 14.3T tokens, then 0
Batch size	3072 $\rightarrow$ 15360 over first 469B tokens
Topology	2048 H800 GPUs, 16-way PP, 64-way EP, ZeRO-1 DP
Precision	FP8 mixed precision, DualPipe scheduling
Cost	2.664M H800 GPU hours

[REPORTED: DeepSeek-V3 technical report]: exact corpus mixture, per-step order, and production rank trace remain private.

Token accounting and scaling laws

A trillion-token count is not self-authenticating

- A training token is a tokenizer output item the objective consumes — it can repeat, be resampled, or be filtered before training.

A trillion-token count is not self-authenticating

- A training token is a tokenizer output item the objective consumes — it can repeat, be resampled, or be filtered before training.
- Reporting "14.8T training tokens" answers one question: how much tokenized stream the objective processed.

A trillion-token count is not self-authenticating

- A training token is a tokenizer output item the objective consumes — it can repeat, be resampled, or be filtered before training.
- Reporting "14.8T training tokens" answers one question: how much tokenized stream the objective processed.
- It leaves the mixture, filtering, deduplication, curriculum weighting, and stream reproducibility open.

A trillion-token count is not self-authenticating

- A training token is a tokenizer output item the objective consumes — it can repeat, be resampled, or be filtered before training.
- Reporting "14.8T training tokens" answers one question: how much tokenized stream the objective processed.
- It leaves the mixture, filtering, deduplication, curriculum weighting, and stream reproducibility open.

The toy run fixes $N = BS$ tokens per step: $B = 2$, $S = 6$, $N = 12$ — small enough to check every later claim by hand.

A trillion-token count is not self-authenticating

- A training token is a tokenizer output item the objective consumes — it can repeat, be resampled, or be filtered before training.
- Reporting "14.8T training tokens" answers one question: how much tokenized stream the objective processed.
- It leaves the mixture, filtering, deduplication, curriculum weighting, and stream reproducibility open.

The toy run fixes $N = BS$ tokens per step: $B = 2$, $S = 6$, $N = 12$ — small enough to check every later claim by hand.

Corpus size, training-token count, and reproducible stream are three different objects. A scale number answers only the middle one.

A scaling law is a fit, not a universal constant

$$\widehat{L}(M_{\text{work}}, N_{\text{tok}}) = L_\infty + A\, M_{\text{work}}^{-\alpha} + B\, N_{\text{tok}}^{-\beta}$$

- DeepSeek LLM fit its own IsoFLOP profiles and reported $M_{\text{work,opt}} \propto C_{\text{train}}^{0.524}$, $N_{\text{tok,opt}} \propto C_{\text{train}}^{0.476}$ — and a *different* exponent pair on a different in-house data version.
- V3 reports 671B total, 37B activated parameters, 14.8T tokens, and 2.664M GPU hours — one production point, not a fitted curve.

An exponent or tokens-per-parameter ratio is never a universal constant. No fitted V3 scaling law is public. [REPORTED: DeepSeek LLM scaling study; DeepSeek-V3 technical report]

From architecture to a training-state ledger

Active computation is per token; training state is not

- A routed token activates k experts of E; the routed-load vector obeys $\sum_e c_e = kN$ before drops or capacity policy.

Active computation is per token; training state is not

- A routed token activates k experts of E; the routed-load vector obeys $\sum_e c_e = kN$ before drops or capacity policy.
- MoE training state is broader: dense, shared-expert, routed-expert, and router parameters, plus gradients and optimizer state for *all* of them — because different tokens and steps activate different experts.

Active computation is per token; training state is not

- A routed token activates k experts of E; the routed-load vector obeys $\sum_e c_e = kN$ before drops or capacity policy.
- MoE training state is broader: dense, shared-expert, routed-expert, and router parameters, plus gradients and optimizer state for *all* of them — because different tokens and steps activate different experts.

Toy check: $N = 12$, $E = 4$, $k = 2$ gives the valid, imbalanced load $(6, 5, 7, 6)$, summing to $kN = 24$.

Active computation is per token; training state is not

- A routed token activates k experts of E; the routed-load vector obeys $\sum_e c_e = kN$ before drops or capacity policy.
- MoE training state is broader: dense, shared-expert, routed-expert, and router parameters, plus gradients and optimizer state for *all* of them — because different tokens and steps activate different experts.

Toy check: $N = 12$, $E = 4$, $k = 2$ gives the valid, imbalanced load $(6, 5, 7, 6)$, summing to $kN = 24$.

Counting only the two experts one token selected is the chapter's first named failure mode — it omits state other tokens and steps touch.

The peak-memory ledger, six named rows

$$M_{\text{peak}} = M_{\text{param}} + M_{\text{grad}} + M_{\text{opt}} + M_{\text{act}} + M_{\text{comm}} + M_{\text{temp}}$$

- Parameters, gradients, and optimizer moments are whole-run resident tensor state; activations, comm buffers, and workspace are step-lifetime state.

The peak-memory ledger, six named rows

$$M_{\text{peak}} = M_{\text{param}} + M_{\text{grad}} + M_{\text{opt}} + M_{\text{act}} + M_{\text{comm}} + M_{\text{temp}}$$

- Parameters, gradients, and optimizer moments are whole-run resident tensor state; activations, comm buffers, and workspace are step-lifetime state.
- The checkpoint payload C_t is a separate recovery object — not a seventh term in M_{peak}.

The peak-memory ledger, six named rows

$$M_{\text{peak}} = M_{\text{param}} + M_{\text{grad}} + M_{\text{opt}} + M_{\text{act}} + M_{\text{comm}} + M_{\text{temp}}$$

- Parameters, gradients, and optimizer moments are whole-run resident tensor state; activations, comm buffers, and workspace are step-lifetime state.
- The checkpoint payload C_t is a separate recovery object — not a seventh term in M_{peak}.

Every row needs a named owner, lifetime, precision, and sharding rule before a memory claim means anything.

FP8: motivation and local definitions

V3's FP8 recipe, by numerical role

$$\hat{x}_i = s\, q_i$$

Recipe surface	V3's reported choice
Payload format	E4M3
Scale granularity	activations 1×128; weights 128×128
Scale timing	online, per tile/block
Accumulation	FP32 promotion every $N_C = 128$ elements

Payload dtype, scale domain, and accumulation width are *independent* choices — a released weight file fixes only the first two. [REPORTED: DeepSeek-V3 technical report] [INSPECTED: DeepSeek-V3 released FP8 weights]

From concept to code: block-scaled FP8 kernels

The reference inference path specializes exactly the surfaces the recipe names: a per-block activation scale, a per-block weight scale, and an FP8 matrix product.

from `DeepSeek-V3/inference/kernel.py`

```
@triton.jit
def act_quant_kernel(x_ptr, y_ptr, s_ptr,
                     BLOCK_SIZE: tl.constexpr, scale_fmt: tl.constexpr):
    ...
@triton.jit
def weight_dequant_kernel(x_ptr, s_ptr, y_ptr, M, N,
                          BLOCK_SIZE: tl.constexpr):
    ...
```

From concept to code: block-scaled FP8 kernels

The reference inference path specializes exactly the surfaces the recipe names: a per-block activation scale, a per-block weight scale, and an FP8 matrix product.

from **DeepSeek-V3/inference/kernel.py**

```
@triton.jit
def act_quant_kernel(x_ptr, y_ptr, s_ptr,
                     BLOCK_SIZE: tl.constexpr, scale_fmt: tl.constexpr):
    ...
@triton.jit
def weight_dequant_kernel(x_ptr, s_ptr, y_ptr, M, N,
                          BLOCK_SIZE: tl.constexpr):
    ...
```

`act_quant` and `weight_dequant` are the payload and scale-metadata operations the recipe table names; `fp8_gemm` performs the multiply and accumulates in FP32 — three separate kernels for three separate numerical decisions. [INSPECTED: DeepSeek-V3 `inference/kernel.py`]

Parallelism, recovery, and the reported record

Five ownership axes, one dependency chain

$$(D_{\mathrm{dp}}, D_{\mathrm{tp}}, D_{\mathrm{pp}}, D_{\mathrm{ep}}, D_{\mathrm{sp}})$$

- Tensor owners finish local attention and the router; expert owners receive and compute their dispatched rows; combine restores token order; only then may the pipeline stage publish.

Five ownership axes, one dependency chain

$$(D_{\mathrm{dp}}, D_{\mathrm{tp}}, D_{\mathrm{pp}}, D_{\mathrm{ep}}, D_{\mathrm{sp}})$$

- Tensor owners finish local attention and the router; expert owners receive and compute their dispatched rows; combine restores token order; only then may the pipeline stage publish.
- Dispatch payload: $B_{\mathrm{dispatch},e} = A_e(d\, b_a + b_m)$ — a byte count, not a duration.

Five ownership axes, one dependency chain

$$(D_{\mathrm{dp}}, D_{\mathrm{tp}}, D_{\mathrm{pp}}, D_{\mathrm{ep}}, D_{\mathrm{sp}})$$

- Tensor owners finish local attention and the router; expert owners receive and compute their dispatched rows; combine restores token order; only then may the pipeline stage publish.
- Dispatch payload: $B_{\mathrm{dispatch},e} = A_e(d\, b_a + b_m)$ — a byte count, not a duration.

DualPipe, DeepEP, and EPLB specialize scheduling, dispatch/combine, and placement — without revealing one exact V3 production trace. [REPORTED: DeepSeek-V3 technical report; DualPipe, DeepEP, EPLB READMEs]

The checkpoint tuple and the recovery obligation

$$C_t = (\theta_t,\ \mathrm{opt}_t,\ \mathrm{sched}_t,\ \mathrm{rng}_t,\ \mathrm{data}_t,\ \mathrm{meta}_t)$$

- Weights alone serve inference; a training resume additionally needs optimizer, schedule, random-state, and data-position fields.
- Omitting any field that determines the next update changes the resumed trajectory, even when θ_t matches exactly.

DeepSeek reports V3 completed without irrecoverable loss spikes or rollbacks — a self-reported outcome; the private cadence, incident log, and storage path stay undisclosed. [REPORTED: DeepSeek-V3 repository]

The reported record: compute by stage

Stage	H800 GPU hours
Pretraining	2.664M
Context extension	119K
Post-training	5K
Total	2.788M

Matched ablations credit MTP, auxiliary-loss-free balancing, and the sequence-wise balance signal within the reported setup — comparisons that hold data, tokens, and schedule fixed.

An ablation delta supports a mechanism claim only when everything else in the comparison is held fixed. A cost total is a GPU-hour sum, not a wall-clock or engineering-labor account.

[REPORTED: DeepSeek-V3 technical report]

Concept-to-code map for the chapter

- **model dimensions and MoE/precision fields** → `DeepSeek-V3/inference/configs/config_671B.json`
- **dense prefix, then routed MoE layers** → `DeepSeek-V3/inference/model.py`
- **FP8 GEMM dispatch and operand scales** → `DeepSeek-V3/inference/model.py`
- **released FP8 weight and scale format** → `DeepSeek-V3/README_WEIGHTS.md`
- **pipeline schedule, dispatch, placement** → `DualPipe, DeepEP, EPLB README`s

Reported spine, derived ledger, undisclosed trace — keep the three apart.

Next: the V4 architecture chapter compares against this training baseline; Volume II's parallelism taxonomy and 3FS chapters turn these ownership tuples into schedules, transports, and durable storage.