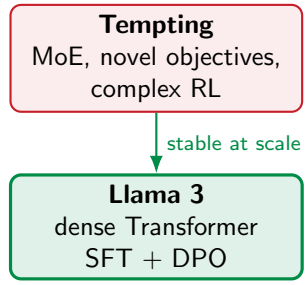
An open model in the GPT-4 league – built the boring way

In 2024 Meta released **Llama 3.1 405B**: an *openly available* model that performs on par with GPT-4 / GPT-4o / Claude 3.5 Sonnet.

You might expect an exotic architecture behind it. It is the opposite:

- A **standard dense Transformer** – deliberately *not* mixture-of-experts.
- Alignment by **SFT + DPO** – deliberately *not* PPO / complex RL.

The bet: at this scale, **managing complexity** matters more than cleverness. Pour the effort into *data* and *scale*, keep everything else simple and stable.

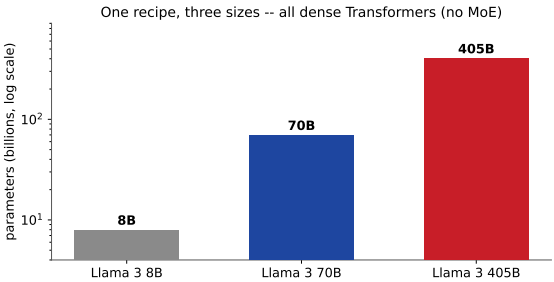


Outline

The herd and the philosophy

Three sizes, one recipe, and three levers that drive quality.

Three members, one recipe



- **8B** and **70B**: efficient, best-in-class for their size.
- **405B**: the flagship, $\approx$ compute-optimal for the training budget.
- All three are **dense Transformers** trained with the *same* pipeline.
- Smaller models are trained *well past* compute-optimal – worse per FLOP of training, but cheaper to *serve* at a given quality.

The flagship is also used to **improve the small models** during post-training (synthetic data, rejection sampling).

Three levers: data, scale, managing complexity

The paper's own framing of what makes a high-quality foundation model:

Data

More *and* cleaner. 15.6T curated tokens vs 1.8T for Llama 2 – heavy de-duplication and quality filtering.

Scale

3.8×10^{25} FLOPs, $\approx 50\times$ Llama 2. Scaling laws pick the size *and* forecast accuracy.

Complexity

Choose the *stable* option every time: dense over MoE, SFT+DPO over PPO. Fewer ways to fail at scale.

"Managing complexity" is a real design principle here: every choice is made to **maximize training stability** and keep the process scalable, even at the cost of a possibly-fancier alternative.

Data: the real workhorse

15.6T tokens – and most of the effort is in cleaning, not collecting.

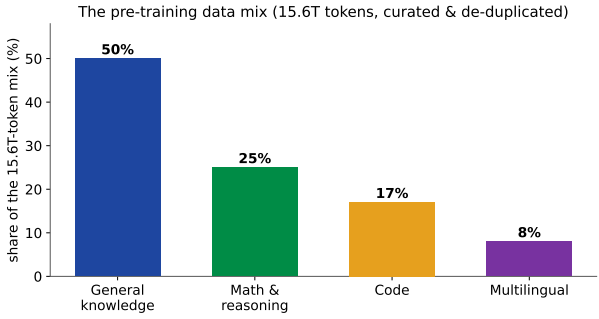
From raw web to high-quality tokens

Llama 3 is pre-trained on $\sim$**15.6T** tokens (knowledge through end of 2023). The pipeline is a stack of filters, not a single scrape:

- **De-duplication** at three levels: URL, document (global MinHash), and line (drop lines seen >6 times per 30M-doc bucket).
- **Heuristic filtering:** duplicated-n-gram removal, dirty-word lists, KL-divergence outlier-token filtering.
- **Model-based quality classifiers:** fastText and DistilRoBERTa scorers, trained on *Llama 2* judgments of document quality.
- **Domain pipelines** for code and math (custom HTML extraction, since their token distribution differs from prose).

A better tokenizer helps too: 128K vocab lifts English compression from **3.17 to 3.94** characters/token – more text "read" per unit of compute.

The data mix – and annealing at the finish



The mix is **chosen**, not inherited: a knowledge classifier *downsamples* over-represented web topics (arts, entertainment), and small-scale scaling-law runs compare candidate mixes.

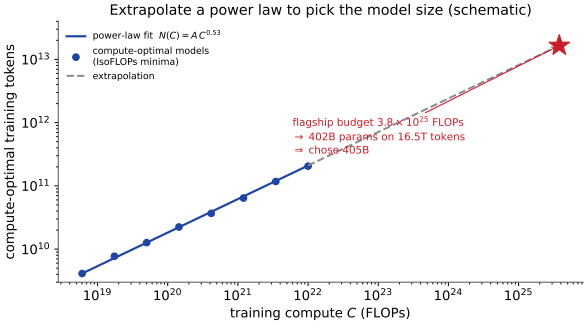
Annealing. On the *final* tokens, upsample very-high-quality math and code while decaying the learning rate to 0.

Annealing on quality data lifted 8B by **+24%** on GSM8K – but barely moved 405B. Scale can absorb what small models need spoon-fed.

Scale and scaling laws

How they chose 405B – and predicted its benchmark scores before training it.

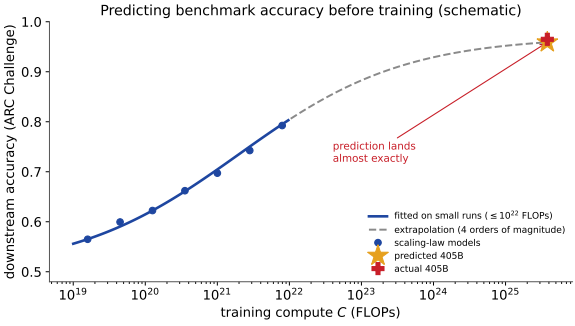
Choosing the size with a scaling law



1. Train many small models across compute budgets (6×10^{18} to 10^{22} FLOPs); read off the **compute-optimal** point at each budget.
2. Fit a power law $N(C) = A\,C^{\alpha}$, with $(\alpha, A) = (0.53, 0.29)$.
3. **Extrapolate** to the flagship budget 3.8×10^{25} FLOPs.

The law says ≈402B params on ≈16.5T tokens. Near the optimum the curve is *flat*, so **405B** was a safe, round choice.

The teachable trick: predict accuracy, not just loss



A raw scaling law predicts *loss*. Meta wanted *benchmark accuracy*, via two chained fits:

1. compute FLOPs $\rightarrow$ **negative log-likelihood** of the correct answer;
2. that log-likelihood $\rightarrow$ **task accuracy** (a sigmoid, anchored with Llama 2 runs).

Extrapolating **4 orders of magnitude**, the forecast for the 405B model landed almost exactly on the real score (it only slightly *un-*dershot).

Architecture: modest by design

Small tweaks to a familiar Transformer – stability over novelty.

What actually changed vs Llama 2

A **standard dense Transformer**. The gains come from data and scale, not the architecture. The handful of deliberate changes:

- **Grouped-Query Attention (GQA)**, 8 key-value heads – shrinks the KV cache and speeds up inference/decoding at scale.
- **128K-token vocabulary** (100K tiktoken + 28K for non-English) – better compression, better multilingual behavior.
- **RoPE base frequency** $\theta = 500{,}000$ – rotary positions that stay well-behaved out to very long contexts.
- **Context window 8K $\rightarrow$ 128K**, extended gradually in *six* stages ($\sim$800B tokens) at the end of pre-training.
- Cross-document attention masking; SwiGLU activation.

Notably *absent*: mixture-of-experts. A dense model is easier to train stably – exactly the "managing complexity" lever.

The three sizes, side by side

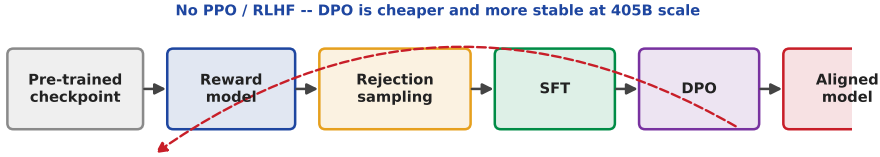
	8B	70B	405B
Layers	32	80	126
Model dimension	4,096	8,192	16,384
FFN dimension	14,336	28,672	53,248
Attention heads	32	64	128
Key/Value heads (GQA)	8	8	8
Peak learning rate	3×10^{-4}	1.5×10^{-4}	8×10^{-5}

Shared across all three: SwiGLU activation, 128K vocabulary, RoPE ($\theta = 500{,}000$), and up to 128K context. Same recipe, scaled up. (Table 3.)

Post-training: the simple alignment loop

SFT + rejection sampling + DPO, repeated – and pointedly no PPO.

The alignment loop



- Train a **reward model** on human preference pairs.
- **Rejection sampling**: draw K=10–30 answers from the best current model, keep the RM's top pick.
- **SFT** on that curated + synthetic data, then **DPO**.
- Run the whole thing for **6 rounds**; each round's model generates the *next* round's data.
- **Model averaging** across RM / SFT / DPO runs.
- Most SFT targets are *model-generated*, then filtered.

DPO, not PPO – and why

For preference alignment Llama 3 uses **Direct Preference Optimization**, the same objective from our *[LLM-2] DPO* deck – optimize preferences directly, *no* reward-model-in-the-loop RL rollouts.

They *tried* PPO, but found DPO:

- needs **less compute** at 405B scale;
- is **more stable**, and better on instruction-following (IFEval).

Two small but important DPO tweaks:

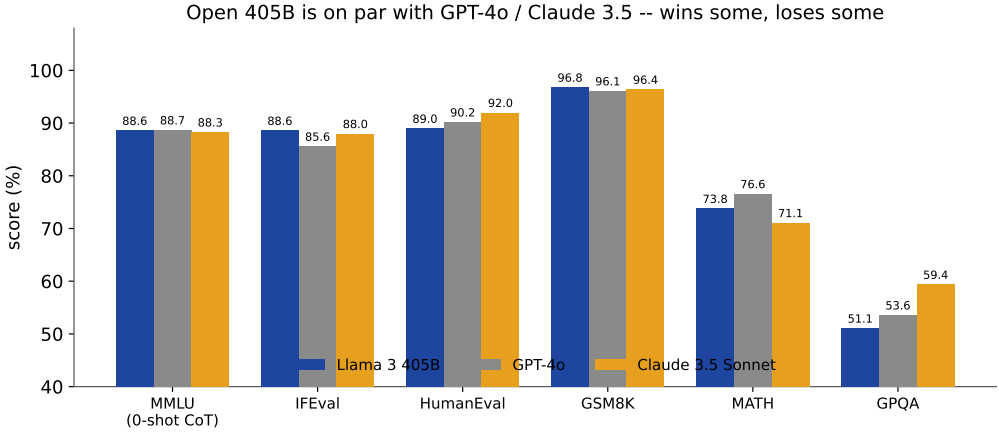
- **mask formatting tokens** (headers / end-of-turn) out of the loss – otherwise the contrastive objective destabilizes;
- add a small **NLL regularizer** (0.2) on the chosen response.

The whole alignment stack is intentionally the *simple* one – consistent with the "managing complexity" philosophy. (See *[LLM-2] DPO*.)

Does the boring recipe work?

An open 405B against the strongest closed frontier models.

On par with the frontier



Llama 3 405B trades wins with GPT-4o and Claude 3.5 Sonnet: ahead on GSM8K and IFEval, essentially tied on MMLU, a touch behind on HumanEval and the hardest reasoning (GPQA). The point is not that it wins – it is that an **open** model is *in the conversation*. (Table 2, exact numbers.)

One awareness note: multimodal is bolted on

The paper also reports *early* image, video, and speech experiments – built with a deliberately **compositional** approach:

- Keep the language model largely fixed.
- Attach separate **vision** and **speech** encoders/adapters via cross-attention.
- Train the adapters, not a single joint model from scratch.

These multimodal models were **not released** with Llama 3 and were still under development. Same philosophy though: add capability with a *modular, stable* extension, not a ground-up redesign.

Recap

- **Llama 3.1 405B**: an open, dense-Transformer model on par with GPT-4o / Claude 3.5 Sonnet – built with a deliberately *simple* recipe.
- **Three levers:** data, scale, and *managing complexity* (stability first: dense over MoE, SFT+DPO over PPO).
- **Data** is the workhorse: 15.6T tokens, aggressive de-duplication + quality filtering, a chosen mix ($\approx$50/25/17/8), quality annealing at the end.
- **Scaling laws** picked 405B *and* forecast downstream accuracy across four orders of magnitude of compute.
- **Architecture** changed little: GQA, 128K vocab, RoPE θ=500k, context to 128K.
- **Post-training** = SFT + rejection sampling + DPO, six iterative rounds.

The lesson: at frontier scale, a *well-executed simple recipe* – great data, honest scaling laws, stable training – beats architectural cleverness.