

**A model's parameter count, long the headline figure, has quietly stopped meaning what it used to. In August 2026 the Chinese laboratory Z.ai released two models less than a fortnight apart. The larger, GLM-5.3, carries 753,329,940,480 parameters — a count that is not a marketing claim but an arithmetic fact, computed by Hugging Face from the published weight files. It is also, to the last parameter, the count of the model it succeeds, and the two share every architectural setting. Z.ai attributes every gain to the training that came after, and the price did not move. The second model, GLM-5.3-Flash, is less than half the size, scores three points lower on an independent index of ten evaluations, and cost about one-ninth as much to put through it. Those two facts frame the question: how a smaller or sparser system inherits capability, and what a unit of quality actually costs. Two mechanisms do much of the work. One compresses a large model into a small one; its recipe was set out in 2006. The other lets a very large model run only a fraction of itself on any given token; its ancestor was published in 1991. They are often conflated, they solve different problems, and only one of them saves memory. And in the comparisons that follow, a third quantity — how many tokens a model uses — decides the bill.**

---

## 1. A model that did not get bigger

---

Hugging Face computes a parameter total directly from the tensors in a repository's weight files, rather than from anything the publisher writes. For `zai-org/GLM-5.3` that total is 753,329,940,480. For `zai-org/GLM-5.2`, whose repository was created in June, it is 753,329,940,480.

GLM-5.3 arrived in stages. The US National Institute of Standards and Technology's Center for AI Standards and Innovation (CAISI), in an assessment published on 17 September, dates the model's release to 14 August 2026 and the public release of its weights to "two weeks later"; Z.ai's own release notes list the model on 18 August; the Hugging Face repository was created on 25 August, and its earliest surviving commit is dated 27 August.

The repositories' configuration files tell the same story from a second direction. GLM-5.2's `config.json` has 55 fields; GLM-5.3's has 56. Of the 55 they share, exactly one differs, and it records which version of the `transformers` library wrote the file. The extra field in GLM-5.3 is a quantisation block: its default download stores about 751.2 billion of the same number of parameters at eight-bit precision and about 2.1 billion at higher precision, in half as many weight shards. Every field that describes the architecture — 256 routed experts, 8 of them active per token, one shared expert, 78 layers of which the first three are dense, a hidden width of 6,144, a vocabulary of 154,880, a context window of 1,048,576 tokens — is identical across the two. The values of the weights are another matter: post-training changes them, which is the point of it.

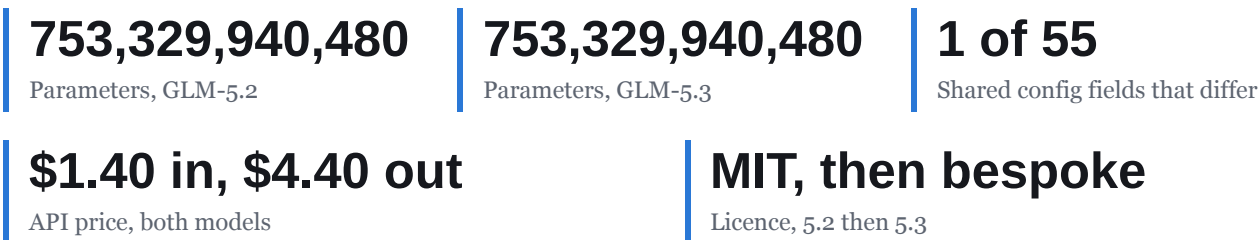
Z.ai's own account of this occupies half a sentence at the top of the model card:

GLM-5.3 uses the same base model as GLM-5.2 — every gain comes from post-training.

The developer documentation repeats it: "It uses the same base model as GLM-5.2, with all improvements driven by post-training." The claim is the vendor's; the files can test its architectural half, and they bear it out.

The price did not move either. Z.ai's published rate card, read on 22 and again on 23 September 2026, lists both GLM-5.2 and GLM-5.3 at \$1.40 per million input tokens, \$0.26 per million cached input tokens and \$4.40 per million output tokens. The laboratory's headline claim for the upgrade — "a 50% improvement over GLM-5.2 on our in-house Z.ai Code Bench" — rests on a benchmark it runs itself; neither the model card nor the developer documentation provides material to reproduce it. Its table of public benchmarks reports gains that include 7.2 points on Terminal Bench 2.1 (88.2 against 81.0), 4.7 points on Agents' Last Exam (28.5 against 23.8) and a rise from 4.6 to 28.3 on Terminal Bench 3.0 — developer-reported results on different scales, not one ordered range.

Figure 1. GLM-5.2 and GLM-5.3, as their own published files describe them



Sources: the Hugging Face weight index and config.json for zai-org/GLM-5.2 and zai-org/GLM-5.3, and Z.ai's published price list, read 22 September 2026 and unchanged on 23 September. The single differing shared field is transformers\_version; GLM-5.3 additionally carries an FP8 quantization\_config that GLM-5.2 does not. GLM-5.2 ships its parameters across 282 weight shards at BF16 precision; GLM-5.3 stores 751.2 billion of the same number of parameters at FP8 and the remaining 2.1 billion at higher precision, across 141 shards — roughly half the bytes. Z.ai also publishes a separate GLM-5.3-BF16 repository with the same parameter total.

▼ Table view

Figure 1. GLM-5.2 and GLM-5.3, as their own published files describe them

| Measure                          | Value                 |
|----------------------------------|-----------------------|
| Parameters, GLM-5.2              | 753,329,940,480       |
| Parameters, GLM-5.3              | 753,329,940,480       |
| Shared config fields that differ | 1 of 55               |
| API price, both models           | \$1.40 in, \$4.40 out |
| Licence, 5.2 then 5.3            | MIT, then bespoke     |

The 753 billion is not Z.ai's figure, and the distinction is worth keeping. The GLM-5.3 model card states no parameter count. The count comes from Hugging Face's index of the weight files, and Artificial Analysis republishes it. Z.ai's own published pair, in the GLM-5 technical report, is 744 billion total and 40 billion active — stated about GLM-5, the earlier model, not about GLM-5.3.

The weight index also sorts the family into two pairs. GLM-5 and GLM-5.1 both count 753,864,139,008 parameters; GLM-5.2 and GLM-5.3 both count 753,329,940,480 — although the expert, layer and prediction-layer fields are the same in all four configuration files, and what accounts for the difference of about half a billion is not stated. The matching configurations establish a shared published design for GLM-5.2 and GLM-5.3; neither they nor the identical totals prove a shared training history, since the same architecture trained again from scratch would count the same.

The second number on the page matters more than the first. Of those 753 billion parameters, roughly 40 billion perform any arithmetic on any particular token; the rest are resident and idle for that token. Artificial Analysis, an evaluation firm that neither built nor sells the model, publishes both counts side by side, under the headings "Total parameters" and "Active parameters", and the 40 billion can be rebuilt from the published configuration to within a few hundred million.

## 2. Two readings the files do not support

---

The first is the one the headline invites: that 753 billion parameters make this about six times the size of a model with 128 billion. Mistral Medium 3.5, at 128 billion parameters, is shown on Artificial Analysis's parameter chart as dense — 128 billion active, none passive — so every one of them runs on every token. GLM-5.3 runs about 40 billion. On active parameters, a rough guide to the arithmetic per token, the smaller-sounding model is roughly three times the larger-sounding one; on total parameters, a rough guide to what must be stored, the relationship reverses — neither ratio is a ratio of what a query costs. GLM-5.3 does score far higher on the same independent index, 44.8 against 14.2, and nothing in either count explains why.

The second reading is the opposite error: that sparsity makes a model cheap. Artificial Analysis files GLM-5.3, in the standard summary line its model pages carry, as

amongst the leading models in intelligence, but particularly expensive when comparing to other open weight models of similar size

— a clause that appears word for word on its GLM-5.2 and Mistral Medium 3.5 pages as well, and that ranks price against open-weights peers of similar size rather than delivering a bespoke verdict. The same page adds that the model is "very verbose", generating 210 million output tokens over the index against a median of 140 million. Sparse activation avoids the arithmetic of the unselected experts. It does not by itself determine how many tokens a model processes, and it does not set the price per token, which is a commercial decision. Cheap to compute and cheap to buy are separate properties, and the distance between them is where much of the confusion about model economics lives.

### 3. The idea: two routes to cheaper capability

---

Two older ideas explain how a large model's ability ends up in something that costs less to run — there are others, such as storing the weights at lower precision, which GLM-5.3's own default download does — and they are not variants of each other. One compresses. The other routes. They have different origins, different mechanisms, and different things they fail to do.

#### 3.1 Compression, and what actually transfers

The paper usually cited for it is not where the recipe was first set out. In August 2006, at the ACM's knowledge-discovery conference in Philadelphia, Cristian Buciluă, Rich Caruana and Alexandru Niculescu-Mizil published *Model Compression*, whose second section states the idea in one sentence:

The main idea behind model compression is to use a fast and compact model to approximate the function learned by a slower, larger, but better performing model.

It sets out the ensemble-to-small-model recipe that the 2015 distillation paper explicitly acknowledges. The procedure is worth stating plainly because it explains the shape of everything since. The expensive model — in their case an ensemble of hundreds or thousands of classifiers — is used to label a very large pool of unlabelled data. A small neural network is then trained on those labels rather than on the original training set. The result, in their words, is "a neural net that makes predictions similar to the ensemble, and which performs much better than a neural net trained on the original training set". Where unlabelled data was unavailable they synthesised it, with a method they called MUNGE, and reported networks "a thousand times smaller and faster than ensemble selection ensembles, but which have nearly the same performance".

The name arrived nine years later. Geoffrey Hinton, Oriol Vinyals and Jeff Dean posted *Distilling the Knowledge in a Neural Network* on 9 March 2015, and were explicit about the credit:

A version of this strategy has already been pioneered by Rich Caruana and his collaborators

Their paper also records that Caruana's group had trained the small model on the large one's logits — its graded scores — rather than on hard labels. What the 2015 paper added was a generalisation and an explanation. The generalisation is to "raise the temperature of the final softmax until the cumbersome model produces a suitably soft set of targets", of which, the authors show, "matching the logits of the cumbersome model is actually a special case". The explanation corrects the intuition most readers bring. On their account the teacher supplies more than an answer — a probability for every alternative, the wrong ones included — and that extra structure is what a hard label lacks:

The relative probabilities of incorrect answers tell us a lot about how the cumbersome model tends to generalize

with this example:

An image of a BMW, for example, may only have a very small chance of being mistaken for a garbage truck, but that mistake is still many times more probable than mistaking it for a carrot.

A hard label reading "car" tells the student one thing. The teacher's full distribution — overwhelmingly car, faintly truck, negligibly vegetable — tells it what the teacher believes the space of possibilities looks like. On the 2015 account, that structure is what makes soft targets more informative than hard labels; the method does not guarantee that the student reproduces the teacher's predictions, and it can be combined with the ordinary labels.

How faithfully the function moves is less settled than the technique's ubiquity suggests. Samuel Stanton and colleagues put the question in their title — *Does Knowledge Distillation Really Work?* — at NeurIPS in 2021, and answered it in two halves:

We show that while knowledge distillation can improve student generalization, it does not typically work as it is commonly understood: there often remains a surprisingly large discrepancy between the predictive distributions of the teacher and the student, even in cases when the student has the capacity to perfectly match the teacher.

The same abstract records that "more closely matching the teacher paradoxically does not always lead to better student generalization". The technique earns its place empirically; the tidy account of *why* it works — the student acquiring the teacher's function — is often not what those measurements show happening.

A September 2026 model card describes a different teacher signal from the 2015 one. Xiaomi's MiMo team, whose repository for it was created on 21 September 2026, describes a small model as "a 9B agentic model developed by Xiaomi MiMo through supervised fine-tuning of Qwen3.5-9B on MiMo-generated data": the student trains on text the teacher wrote, not on the teacher's probabilities. The scores reported for it are Xiaomi's own.

### 3.2 Conditional computation, and what it does not save

The second idea starts earlier. Robert Jacobs, Michael Jordan, Steven Nowlan and Geoffrey Hinton published *Adaptive mixtures of local experts* in *Neural Computation* volume 3, issue 1, pages 79–87, in 1991: several small networks, and a gating network that decides which of them handles a given example. DeepSeek's own summary of the lineage, in the DeepSeekMoE paper of January 2024, is accurate and brief:

The Mixture of Experts (MoE) technique is first proposed by Jacobs et al. (1991); Jordan and Jacobs (1994) to deal with different samples with independent expert modules.

A mixture of experts is not by itself a saving. Noam Shazeer and colleagues, writing in 2017, describe the classic softmax gate of that line as "a simple choice of non-sparse gating function": every expert receives some weight, so every expert has to be computed. The saving arrives when the gate is made sparse and the unchosen experts are skipped. Yoshua Bengio discussed it as a research direction in *Deep Learning of Representations: Looking Forward*, posted on 2 May 2013, with a precedent: "decision trees exploit

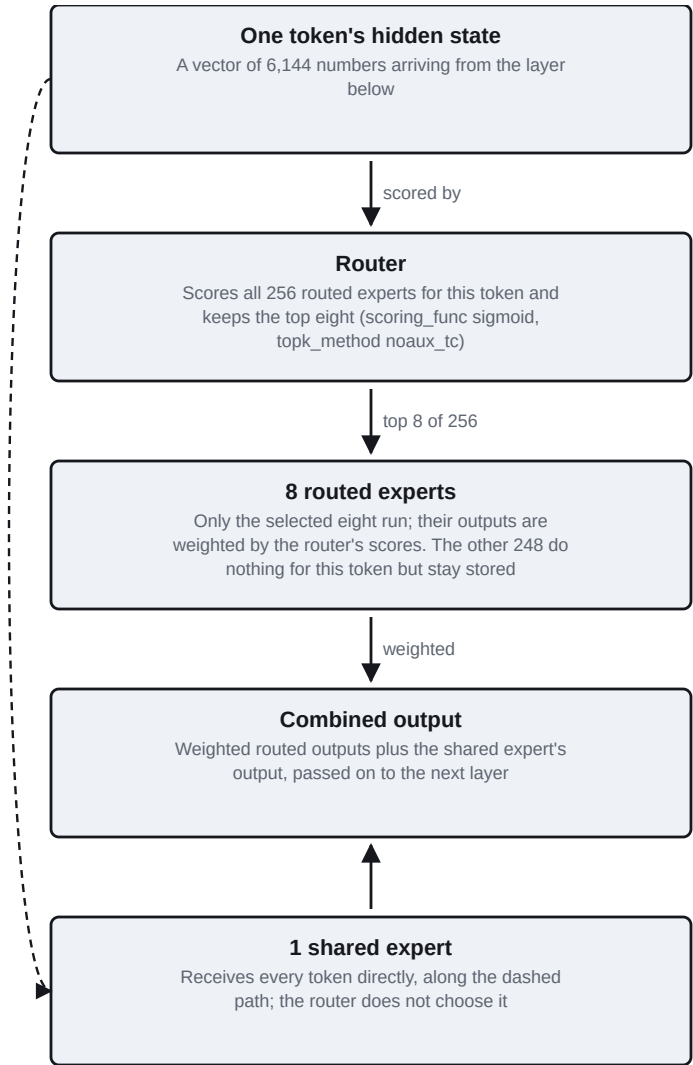
conditional computation: for a given example, as additional computations are performed, one can discard a gradually larger set of parameters (and avoid performing the associated computation)". Its application at language-model scale arrived on 23 January 2017, when Shazeer's team at Google posted *Outrageously Large Neural Networks*, whose abstract states the principle, and whose introduction credits proposals from 2013 onwards:

Conditional computation, where parts of the network are active on a per-example basis, has been proposed in theory as a way of dramatically increasing model capacity without a proportional increase in computation.

Two quantities are named there, and the whole subject is the relationship between them. *Capacity* is how much a model can hold: its total parameters. *Computation* is how much work it does per token. In a dense network the two are welded together, because adding a parameter means running it. Conditional computation unwelds them. Shazeer's team reported the result as "greater than 1000x improvements in model capacity with only minor losses in computational efficiency", in models whose mixture-of-experts layer held up to 137 billion parameters.

The mechanism inside a current model is not complicated. GLM-5.3's configuration file describes 75 sparse layers, each holding 256 expert sub-networks. For each token, a small learned router scores all 256, the eight highest-scoring ones run, and a single shared expert runs for every token regardless. Nine experts do that layer's expert work; 248 do nothing for that token and may be selected for the next one.

Figure 2. The expert part of one sparse layer of GLM-5.3, for one token



Schematic of the feed-forward (expert) block only; attention and residual paths are omitted. Built from the published `config.json` of `zai-org/GLM-5.3` (`n_routed_experts` 256, `num_experts_per_tok` 8, `n_shared_experts` 1, `hidden_size` 6,144, `first_k_dense_replace` 3, `num_hidden_layers` 78), read 22 September 2026 and byte-identical on 23 September. Three of the 78 layers are dense and have no router. The shared expert bypasses the selection, following the shared-expert design DeepSeek set out in January 2024. What any individual expert specialises in, and why a router sends a given token where it does, is not published by the vendor and cannot be read off a configuration file.

▼ Table view

Figure 2. The expert part of one sparse layer of GLM-5.3, for one token — stages

| # | Stage                    | Note                                                                                                                                         |
|---|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| 1 | One token's hidden state | A vector of 6,144 numbers arriving from the layer below                                                                                      |
| 2 | Router                   | Scores all 256 routed experts for this token and keeps the top eight ( <code>scoring_func</code> sigmoid, <code>topk_method</code> noaux_tc) |
| 3 | 8 routed experts         | Only the selected eight run; their outputs are weighted by the router's scores. The other 248 do nothing for this token but stay stored      |
| 4 | Combined output          | Weighted routed outputs plus the shared expert's output, passed on to the next layer                                                         |

| # | Stage           | Note                                                                                |
|---|-----------------|-------------------------------------------------------------------------------------|
| 5 | 1 shared expert | Receives every token directly, along the dashed path; the router does not choose it |

Figure 2. The expert part of one sparse layer of GLM-5.3, for one token — connections

| From                     | To               | Label        |
|--------------------------|------------------|--------------|
| One token's hidden state | Router           | scored by    |
| Router                   | 8 routed experts | top 8 of 256 |
| 8 routed experts         | Combined output  | weighted     |
| One token's hidden state | 1 shared expert  |              |
| 1 shared expert          | Combined output  |              |

The shared expert is not decoration. GLM-5's architecture table gives it one, as GLM-4.5 had; the design was set out by DeepSeek in January 2024, which proposed segmenting experts finely and then "isolating *Ks* experts as shared ones, aiming at capturing common knowledge and mitigating redundancy in routed experts". The GLM-5 technical report does not cite that paper, so the resemblance is a fact about two architectures rather than an attribution, and GLM's similar structure does not show what its individual experts actually learned.

Sparsity is also not monotonically good, and the clearest statement of that comes from a study arguing for it. Samira Abnar and colleagues, in a January 2025 paper on optimal sparsity for mixture-of-experts models, find "an optimal level of sparsity that improves both training efficiency and model performance" under the constraints they study — an optimum, not a direction — and, as a separate finding, that with "the same perplexity on the pretraining data distribution, sparser models, i.e., models with fewer number of active parameters, perform worse on specific types of downstream tasks that presumably require more "reasoning"".

And here is the limitation that the arithmetic hides. Sparse routing saves computation; it saves no storage. For the model to run at speed, all of its roughly 750 billion parameters must be resident on the serving hardware, because the router may ask for any of them on the next token; experts can be moved to slower memory only at a cost in speed. A sparse model is cheap in the way a large library is cheap to read from and expensive to house. That asymmetry helps make sparse models attractive to operators running many requests in parallel and awkward for anyone serving one request at a time on constrained hardware; with communication, batching and the rest of the computation, it is one reason the arithmetic saving does not translate into a proportional saving on the bill.

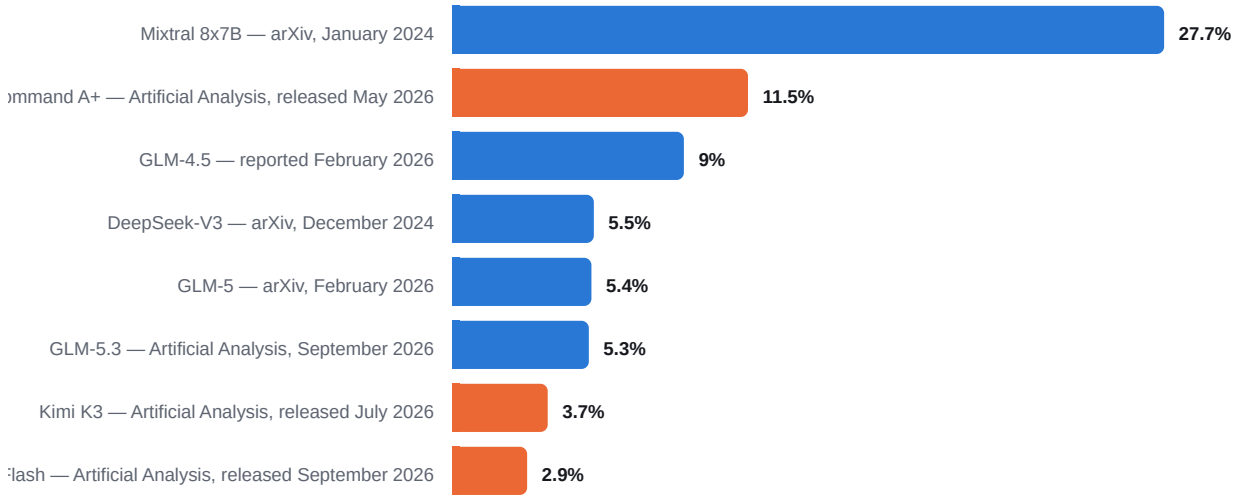
## 4. How much of a model runs on each token

The number to watch in the second idea is one fraction: the share of a model's parameters that run on any given token. Research models reached very small fractions early: Shazeer's team trained models with up to 131,072 experts in which "each example is processed by exactly 4 experts".

Switch Transformer, posted by William Fedus, Barret Zoph and Noam Shazeer on 11 January 2021, took the simplification as far as it goes. Where earlier designs routed each token to two or more experts, Switch "route[s] to only a single expert", and the paper's claim for that choice is modest and empirical: "We show this simplification preserves model quality, reduces routing computation and performs better." Its largest version, Switch-C, had "1.6T parameters and 2048 experts".

What arrived later was publication of both counts by open-weights releases. Mixtral, on 8 January 2024, stated that "each token has access to 47B parameters, but only uses 13B active parameters during inference" — an active share of 28%. DeepSeek-V3, in December 2024, opened with "671B total parameters with 37B activated for each token", or 5.5%. The GLM-5 technical report of February 2026 records 744 billion and 40 billion for GLM-5, 5.4%, and 355 billion and 32 billion for its predecessor GLM-4.5.

Figure 3. Share of a model's parameters that run on any one token



Computed from published pairs of counts — the developers' own for Mixtral, DeepSeek-V3, GLM-4.5 and GLM-5, the evaluator's listings for the rest: Mixtral 47B/13B (arXiv 2401.04088 v1, 8 January 2024); DeepSeek-V3 671B/37B (arXiv 2412.19437 v2, 18 February 2025); GLM-4.5 355B/32B and GLM-5 744B/40B (arXiv 2602.15763 v2, 24 February 2026, Table 10); GLM-5.3 753B/40B, Command A+ 218B/25B, Kimi K3 2,800B/104B and DeepSeek V4.1 Flash 552B/16B (Artificial Analysis model records, read 23 September 2026; listed by the evaluator, not independently measured). Highlighted rows are other 2026 open-weights releases, shown for their spread. The GLM-4.5 row is dated to the report that states its counts rather than to the model's release. Models chosen for having published both numbers illustrate a range; they are not a survey of the field.

▼ Table view

Figure 3. Share of a model's parameters that run on any one token

| Model, and where its counts were published                         | Active share of total parameters |
|--------------------------------------------------------------------|----------------------------------|
| Mixtral 8x7B — arXiv, January 2024                                 | 27.7%                            |
| Command A+ — Artificial Analysis, released May 2026                | 11.5%                            |
| GLM-4.5 — reported February 2026                                   | 9%                               |
| DeepSeek-V3 — arXiv, December 2024                                 | 5.5%                             |
| GLM-5 — arXiv, February 2026                                       | 5.4%                             |
| GLM-5.3 — Artificial Analysis, September 2026                      | 5.3%                             |
| Kimi K3 — Artificial Analysis, released July 2026                  | 3.7%                             |
| DeepSeek V4.1 Flash — Artificial Analysis, released September 2026 | 2.9%                             |

Among these published pairs the share went from roughly a quarter to roughly a twentieth within a year, and the GLM-5 report published the same twentieth in February 2026. Among the 2026 sparse open models for which the evaluator's records carry both counts, the calculated share runs from 2.9% to 11.5% — the range on offer, not a historical minimum. For models built this way the headline parameter count stopped predicting what a query costs to compute; the active count is the better guide to the arithmetic.

The two totals for GLM-5 do not agree, and both are published. The technical report says 744 billion, under a convention its table caption states — multi-token-prediction layers included, word embeddings and the output layer excluded. The weight index says 753,864,139,008. One multi-token-prediction layer, the block

these models carry to predict the token after next, holds about 9.9 billion parameters when rebuilt from the published configuration, which is close to the difference; but subtracting it would contradict the report's own stated convention, and the published sources do not reconcile the two totals. The weight index describes the tensors shipped in the repository; a deployment may omit optional components or store weights at a different precision.

The same report complicates the popular account of distillation. Its final training stage is described as one in which

the final checkpoints from the preceding training stages serve as teacher models

with the stated purpose of mitigating "the cumulative degradation of previously acquired capabilities" that, in the report's words, sequentially optimising for distinct objectives "can lead to". The teacher and the student are the same size, and the procedure is on-policy: the student generates, and the log-ratio of the teacher's probabilities to the student's replaces "the advantage term" in the reinforcement-learning objective. The teacher-and-student idea is Buciluă's; the procedure is not, and the application is not compression at all.

## 5. One laboratory, two answers

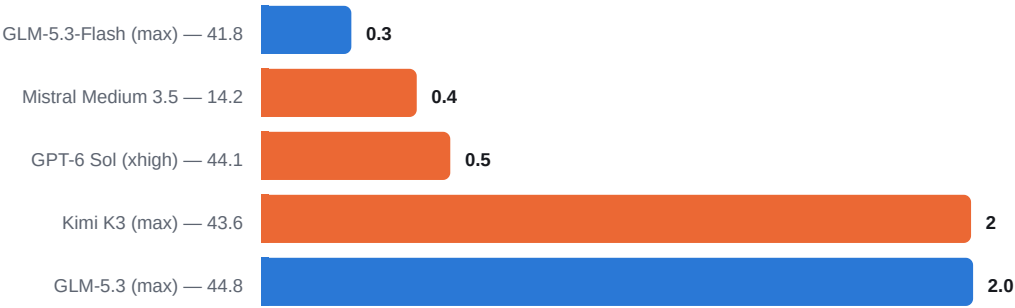
Z.ai's release notes list GLM-5.3-Flash on 26 August, eight days after GLM-5.3. It carries 320 billion parameters with 18 billion active, and its model card states its provenance directly:

GLM-5.3-Flash starts from a newly trained base model, with its architecture and training recipe redesigned around capability and efficiency.

The card does not say how the new base was initialised or name any teacher, so whether GLM-5.3 contributed to its training is not stated. The pair is not a controlled comparison — the smaller model has its own base and its own architecture — but it shows one laboratory answering the same question in two ways within the same month. The card claims that "with 320B total parameters and just 18B active parameters, it outperforms GLM-5.2 across benchmarks and real-world workloads at one-tenth the price" — a vendor claim, though the price element is checkable against the same rate card, which lists Flash at \$0.15 and \$0.50 against GLM-5.3's \$1.40 and \$4.40.

An independent measurement puts numbers on the trade. Artificial Analysis ran both models across Intelligence Index v4.3.2, a fixed battery of ten evaluations, and published the results with their costs. The larger model scored 44.8 and cost \$2.01 per task; the smaller scored 41.8 and cost \$0.25. Putting each model through the whole battery cost \$2,503.48 and \$280.28 respectively — about a ninth on that measure, and about an eighth on the evaluator's weighted cost per task. Hugging Face's "Downloads last month" counters, read on 23 September 2026, stand at 3,547,021 for Flash and 1,039,477 for GLM-5.3 — a ratio of more than three to one, in a counter that tallies requests for specified files, not users, deployments or the reasons for choosing a model.

Figure 4. Cost of one Intelligence Index task, and the score it bought



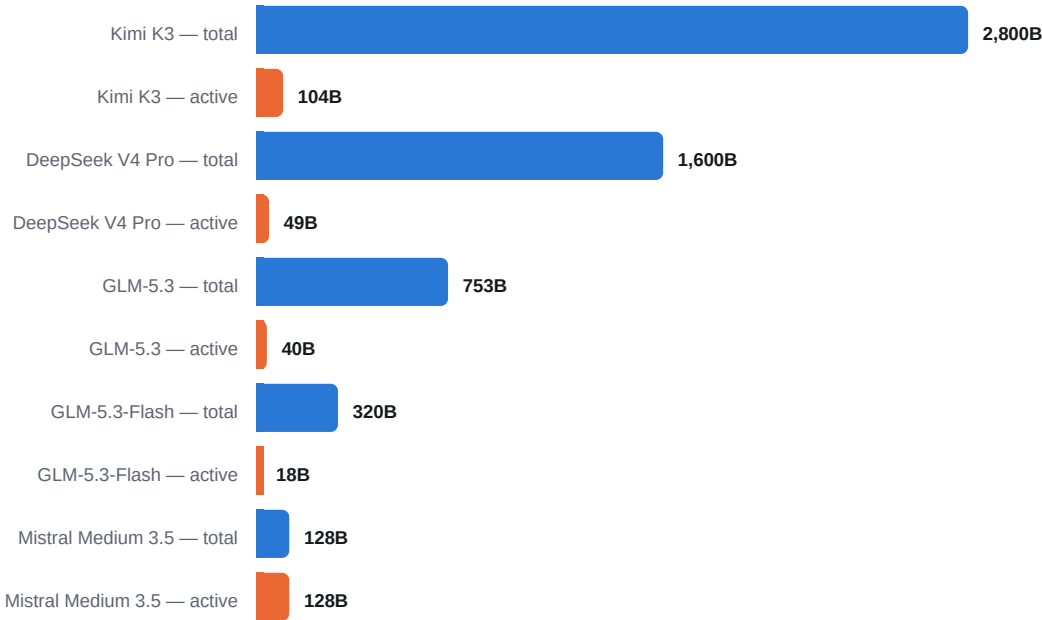
Source: Artificial Analysis, Intelligence Index v4.3.2 (ten evaluations: AA-Briefcase v1.1, GDPval-AA v2.1, AutomationBench-AA, Terminal-Bench 4.0, SciCode, Humanity's Last Exam, GDP.pdf, CritPt, AA-Omniscience, AA-LCR v1.1), model records read 23 September 2026. Settings are those the evaluator lists. The two highlighted rows are the pair released by one laboratory in August 2026. Cost per task is the evaluator's own weighted per-task figure. Artificial Analysis estimates the 95% confidence interval of an index score at less than  $\pm 1\%$ .

▼ Table view

Figure 4. Cost of one Intelligence Index task, and the score it bought

| Model and setting, with its Intelligence Index score | US\$ per Index task |
|------------------------------------------------------|---------------------|
| GLM-5.3-Flash (max) — 41.8                           | 0.3                 |
| Mistral Medium 3.5 — 14.2                            | 0.4                 |
| GPT-6 Sol (xhigh) — 44.1                             | 0.5                 |
| Kimi K3 (max) — 43.6                                 | 2                   |
| GLM-5.3 (max) — 44.8                                 | 2.0                 |

Figure 5. Total parameters against parameters active per token, September 2026



Source: Artificial Analysis model pages, fields "Total parameters" and "Active parameters", read 22 September 2026 and unchanged on 23 September. Mistral Medium 3.5 is dense, so its two counts coincide — it runs more parameters per token than any sparse model in this group and scores 14.2 on the index in Figure 4, below every sparse model shown there. Kimi K3's 104 billion active parameters are more than two and a half times GLM-5.3's 40 billion; on the index in Figure 4 it scores 43.6 against 44.8, at \$2.00 a task against \$2.01.

▼ Table view

Figure 5. Total parameters against parameters active per token, September 2026

| Model, and which count      | Parameters, billions |
|-----------------------------|----------------------|
| Kimi K3 — total             | 2,800B               |
| Kimi K3 — active            | 104B                 |
| DeepSeek V4 Pro — total     | 1,600B               |
| DeepSeek V4 Pro — active    | 49B                  |
| GLM-5.3 — total             | 753B                 |
| GLM-5.3 — active            | 40B                  |
| GLM-5.3-Flash — total       | 320B                 |
| GLM-5.3-Flash — active      | 18B                  |
| Mistral Medium 3.5 — total  | 128B                 |
| Mistral Medium 3.5 — active | 128B                 |

The Kimi row cuts against the simplest story. A model with two and a half times the active parameters and nearly four times the total scores a point lower for the same money. Neither count orders the table on its own, because what a user pays is set by a price list and by how many tokens the model uses, and what a model scores depends on the model and the evaluation setup; the architecture constrains these without determining them.

## Price per token is not cost per task

The Z.ai pair shows a smaller model, with its own base and design, scoring lower at a fraction of the cost. A second comparison, on the same index, shows what decides the bill.

OpenAI released GPT-6 Sol on 22 September 2026. At its second-highest effort setting Artificial Analysis scores it 44.10, against GLM-5.3's 44.78 at its highest — a gap of 0.68 points in the open model's favour. The evaluator estimates the 95% confidence interval of an index score at under  $\pm 1\%$  and publishes none for the difference between two models. Sol's headline prices are higher: \$2.00 per million input tokens and \$10.00 per million output tokens, against \$1.40 and \$4.40. Its price for cached input is slightly lower, \$0.20 against \$0.26, and cached input is where more than half of GLM-5.3's spending on this index goes; weighted by each model's actual mix of tokens, Sol's price list still comes out about 40% dearer.

The evaluator's cost per task nonetheless runs the other way: \$0.53 for Sol against \$2.01 for GLM-5.3, about a quarter, and \$865 against \$2,503 for the whole index, about a third. The two ratios differ because Artificial Analysis's cost per task is a benchmark-weighted average across its ten evaluations, not the whole-run bill divided by a pooled task count. Priced at Sol's rates, GLM-5.3's own usage would have cost about \$3,520; priced at Z.ai's rates, Sol's usage would have cost about \$620. The difference is volume. GLM-5.3 wrote 209 million output tokens over the index against Sol's 40 million, and read 5.41 billion input tokens against 1.31 billion — five times as many written and four times as many read.

Figure 6. Two models a fraction of a point apart, and a fourfold difference in the bill



GLM-5.3 (max) against GPT-6 Sol (xhigh), Artificial Analysis Intelligence Index v4.3.2, model records read 23 September 2026. GPT-6 Sol's listed release date is 22 September 2026; these figures are from the evaluator's records as read on 23 September. Cached-input prices are \$0.26 for GLM-5.3 and \$0.20 for GPT-6 Sol. The two rows are different effort settings of different models, chosen because their scores nearly coincide; GPT-6 Sol at its maximum setting scores 47.53 at \$1.06 a task.

▼ Table view

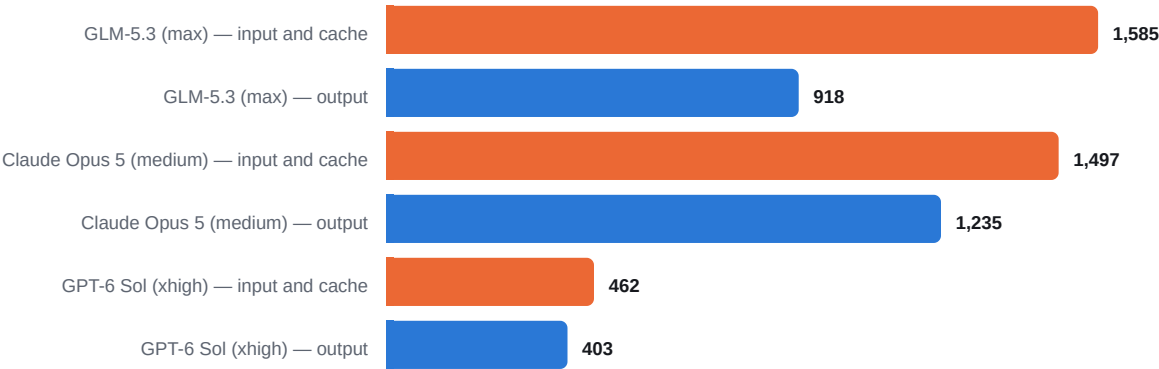
Figure 6. Two models a fraction of a point apart, and a fourfold difference in the bill

| Measure                                    | Value                               |
|--------------------------------------------|-------------------------------------|
| Intelligence Index                         | 44.78 vs 44.10                      |
| List price per 1M tokens, input and output | \$1.40 / \$4.40 vs \$2.00 / \$10.00 |
| Output tokens over the index               | 209M vs 40M                         |
| Input tokens over the index                | 5.41B vs 1.31B                      |
| Cost per Index task                        | \$2.01 vs \$0.53                    |

A similar pattern appears against a closed model priced above both, measured on 22 September, the day Anthropic replaced it with a successor: Claude Opus 5 at medium effort scored 44.83, 0.05 points from GLM-5.3, at uncached-input and output prices 3.6 and 5.7 times GLM-5.3's and a cached-input price 1.9 times, and its index run cost \$2,731.91 against GLM-5.3's \$2,503.48, about 9% more. Artificial Analysis now marks that page deprecated; the comparison stands as a measurement of that date.

Most of GLM-5.3's bill is reading rather than writing. Artificial Analysis's own cost breakdown puts GLM-5.3's output at \$918 of its \$2,503, and its input at \$1,585, of which cache reads alone are \$1,365 — more than half the bill. Against Opus 5, GLM-5.3 spent less on output (\$918 against \$1,235) and more on input including caching (\$1,585 against \$1,497), despite lower list prices in every input category.

Figure 7. Where the money went: one full Intelligence Index run



Source: Artificial Analysis's own *intelligenceIndexCost* breakdown in its model records, read 23 September 2026; totals \$2,503.48, \$2,731.91 and \$865.48. Input includes fresh input, cache reads and cache writes; for GLM-5.3, cache reads alone are \$1,364.52. Output includes reasoning tokens. Scores: 44.78, 44.83 and 44.10.

▼ Table view

Figure 7. Where the money went: one full Intelligence Index run

| Model, and part of the bill              | US\$  |
|------------------------------------------|-------|
| GLM-5.3 (max) — input and cache          | 1,585 |
| GLM-5.3 (max) — output                   | 918   |
| Claude Opus 5 (medium) — input and cache | 1,497 |
| Claude Opus 5 (medium) — output          | 1,235 |
| GPT-6 Sol (xhigh) — input and cache      | 462   |
| GPT-6 Sol (xhigh) — output               | 403   |

That is the cost of a given benchmark score, as opposed to the cost of a token. GLM-5.3's lower price list gives it an advantage per token; the number of tokens it uses, above all the number it reads, takes that advantage back — most of it against Opus 5, and more than all of it against Sol. A buyer comparing rate cards would predict a large saving over Opus 5 and find 8%, and would predict a saving over Sol and find a fourfold premium. Holding the score nearly fixed avoids treating an index point as a unit of quality, which it is not; the comparison remains specific to this evaluator's mix of benchmarks and to the settings tested.

The same arithmetic runs the other way for the smaller Z.ai model, which is cheaper per token and less verbose than its larger sibling — 180 million output tokens against 209 million — and whose weighted cost per task is about an eighth of GLM-5.3's on this index; how much of that saving carries to other workloads is not measured.

The two Z.ai releases also differ in a way that has nothing to do with computation. GLM-5.2 and GLM-5.3-Flash both carry the MIT licence. GLM-5.3 carries a bespoke licence of the same name as the model, whose second clause reads:

If the Licensee or any of its affiliates operates a Model as a Service business, and the aggregate revenue of the Licensee and its affiliates exceeds 10 billion US dollars (or the equivalent in other currencies) in total over any consecutive 12 months, the Licensee must pass Z.AI's security review before using the Software or its derivative works for any commercial purpose.

The test is on a whole corporate group's revenue, not on the size of its model-serving business, and the review precedes any commercial use. The same laboratory released its cheaper model under the MIT licence and attached that condition to its more expensive one.

## 6. Three things the record leaves open

---

The clearest outside measurement bearing on the upgrade answers one question and leaves another. Artificial Analysis scores GLM-5.2 at 33.71 and GLM-5.3 at 44.78, both at maximum effort on Intelligence Index v4.3.2 over the same ten evaluations — eleven points apart on identical weight counts, identical architecture and an identical rate card, measured by a party that sells neither. That is consistent with Z.ai's post-training claim; a difference in scores cannot show which training changes produced it. The GLM-5.2 page also carries a deprecation banner — "This model is deprecated. We only continue performance benchmarking for the default 10k input token workload. Results for other workloads are historical and no longer updated." — which concerns the performance workloads still being updated rather than the index result.

The GLM-5 technical report states that the model "reduces its layer count to 80". Every shipped `config.json` in the family — GLM-5, GLM-5.1, GLM-5.2 and GLM-5.3 alike — records 78 hidden layers plus one multi-token-prediction layer, which is 79. The report's own architecture table lists three dense layers, 75 mixture-of-experts layers and one multi-token-prediction layer — the same categories as the shipped files — so the discrepancy sits inside the report itself, which does not explain it.

Three things only Z.ai can see. The 50% coding improvement is measured on an in-house benchmark, and the card provides no material to reproduce it. Z.ai's claim that nothing but post-training changed is corroborated by the files only as far as *architecture* goes. And what the post-training consisted of is described in the vendor's own vocabulary, which on the GLM-5.3 page includes an acronym it never expands.

## 7. What to watch

---

Three things, each checkable on a page anyone can open.

**The parameter total on the next Z.ai model page.** Hugging Face derives it from the weight files rather than from a press release, which is what makes it a usable signal, and it counts the same whether the download is stored at eight-bit or sixteen-bit precision. If the next release reports 753,329,940,480 again

and its configuration matches, that fits an unchanged base model for a third release running, though matching files cannot prove it. If the count moves, the stored tensors changed, and the vendor's account of why is the next thing to read.

**The "Active parameters" field on Artificial Analysis's model pages.** It is a published proxy for part of the arithmetic a token requires. The large open-weights models on those pages fill it in, as do a few closed models; for GPT-6 Sol and Claude Opus 5.5, both released on 22 September 2026, it is blank. Whether OpenAI or Anthropic ever publishes the figure is the question it answers.

**Cost per task beside token counts on the evaluator's page for Z.ai's next model.** GLM-5.3's position is set by 209 million output tokens and 5.41 billion input tokens over the index at \$2.01 a task. Whether the next model closes the gap to GPT-6 Sol's \$0.53 by using fewer tokens or only by charging less for them is the distinction between an efficiency gain and a price cut.

## 8. The idea to keep

---

The concept is conditional computation: a system in which not every part runs on every input. Its ancestor, the mixture of experts, was published in 1991; the sparse form that saves computation was proposed from 2013 and demonstrated at language-model scale in 2017; and it is now the design behind the largest open-weights models on the evaluators' pages.

The habit that goes with it is asking for two numbers where the headline offers one. Total parameters states how much a model holds and, with the precision it is stored at, what it costs to house. Active parameters is a rough guide to the arithmetic it does for each token. For most of the field's history those two moved together, which is why one number was allowed to stand for both. They no longer move together, and a figure that reports only the larger has described a warehouse while saying nothing about a delivery.

Neither number is the price of an answer, and neither, on its own, predicts quality — Figures 4 and 5 show a model with two and a half times the active parameters of another scoring below it at the same price, and Figure 6 shows a model with the more expensive price list costing a quarter as much per task. The third number is how many tokens the work took. The useful question to put to any claim about a very large model is the narrow one: how many of these parameters ran on this request, how many tokens did the request take, and what did the answer cost?