

Ines-1: Training a 1.6B Mixture-of-Experts Model for Typed Decisions from Scratch

Endikavi
endika.vi@gmail.com

Ines-1 Technical Report, October 2026

Abstract

Typed-decision interfaces ask a model a question about a piece of state and return a probability distribution over a closed set of answers — a choice, an ordinal score or a yes/no — instead of free text. Inspired by such interfaces, we train Ines-1, a 1.6B-parameter mixture-of-experts causal encoder-decoder (405M parameters active per token), from scratch: its main pretraining run processed 9.96B tokens from an approximately 10B-token corpus on a single H200, and two decision fine-tuning stages then teach it to answer from the logits of option letters. On the typed-decisions benchmark, under a single-question specialist protocol after training on the benchmark’s training split, Ines-1 reaches 76.4% accuracy. This competence does not transfer to multi-question prompts: with a whole five-question case in one prompt, accuracy falls to 37–42%, below the dataset’s prior. On a private domain benchmark the model does not transfer reliably zero-shot (macro balanced accuracy 0.558); fine-tuning the released checkpoint with an existing recipe raises it to 0.790, at the cost of 6–9 points of general typed-decision accuracy. We report these results, the negative ones included, with the training and serving recipe and the failures found along the way; the work was carried out by one engineer directing coding agents.

1 Introduction

Many uses of language models in a product are not generation: they are *decisions*. Route this ticket to a department; is this call about a supplier; how urgent is this incident on a 0–4 scale; which tool should the agent call next. The typed-decision interface popularised by TypeSafe’s Jev [57] formalises this as `POST /v1/systemone` with a *state* (text or JSON) and a set of typed questions: **choice** (one of named criteria), **score** (an ordinal scale) and **noul** (a probability that a statement holds). The answer is a probability distribution, not a sentence.

Three families of models can serve such a protocol: large generative LMs prompted to answer (slow, and their probabilities are a by-product), encoders with a classification head trained for it (fast, but small and narrow), and — the subject of this paper — a small generative LM built and trained specifically to answer typed questions from the logits of option letters. Our questions were practical:

1. Can a model pretrained from scratch on a 10B-token public corpus, with its main pretraining run on a single GPU, learn to answer typed decisions with a short fine-tuning stage?
2. How far does that competence carry: to other request shapes, and to a narrow domain without adaptation?
3. How fast can it be served, with no inference framework?

Contributions. (i) An auditable end-to-end recipe for the public model — architecture, data, pretraining, decision fine-tuning, serving — with every number traceable to a log or a commit. (ii) A decision-reading method for causal LMs that needs no new head and serves through a truncated prefill in which the decoder runs only over a fixed trailing window of the prompt, so the decoder’s share of the cost does not grow with prompt length. (iii) A balanced decision-training mix assembled from public-source datasets, with leak checks against the evaluation sets. (iv) Negative and null results that we found as informative as the positive ones, chief among them that in our experiments, task competence did not imply interface generalisation (Section 9.1). (v) An honest record of how a non-specialist built this with coding agents, including the failures that passed every check we had written for them (Section 6).

2 Related work

Small models, mixtures of experts and efficient architectures. Sparsely-gated mixtures of experts [18, 50] decouple total from active parameters; DeepSeekMoE introduced fine-grained experts with shared experts [12], and DeepSeek-V3 bias-based load balancing [16]; OLMoE is a fully open small mixture-of-experts model [37], and SmolLM2 a data-centric small dense model [4]. Ines-1 is a reduced re-implementation of DeepSeek-V4.1-Flash [14], including its Engram memory [9], and uses grouped-query attention [1], rotary position embeddings [53] and sliding-window attention in the decoder [3]; we test context extension with position interpolation [8] and YaRN [44].

Typed-decision models. TypeSafe’s Jev is a hosted model behind a typed-decision API (POST /v1/systemone) [57]. The typed-decisions dataset [32] provides a public test with teacher distributions and keeps separate tables for zero-shot models and for models fitted on its training split; DecisionEval reports an independent run of Jev on it [13]. Open alternatives that we use or compare with include encoder-based decision models (Julia-1 [54], fine-tuned from mmBERT-small [34]; laya-multilingual [11], whose card names an mmBERT-base backbone) and a Jev-compatible server over a diffusion language model (OpenJev [46] over DiffusionGemma [21]). Our model differs in being a small generative model trained from scratch and read through option-letter logits; we do not claim to reproduce Jev’s architecture or protocol.

Specialisation, distillation and calibration. Adapting a pretrained model to a domain [23], training on a teacher’s output distribution [25], and the loss of earlier abilities after specialisation [29] are long-studied; we measure all three in one setting. We evaluate probabilities with the Brier score [7], KL divergence from the teacher distribution and expected calibration error [22, 38], with case-level bootstrap intervals [17] and exact McNemar tests [35] for paired comparisons.

3 How the work was done

This is not the work of an ML lab. The author is a software engineer, not a machine-learning researcher, and had not pretrained a language model before. The model was built in sixteen days (12–28 September 2026) on rented NVIDIA H200 GPUs: one for most of that time, occasionally two or three. The first prototype ran on a desktop RTX 5060 Ti with 16 GB.

Why this architecture. DeepSeek-V4.1-Flash was, as far as the author could judge, one of the most carefully optimised open architectures of its generation, above all in how it manages the KV cache: a decoder that attends to a short window plus one shared, compressed global memory instead

of a full cache per layer. It had also adopted Engram [9], a hashed n -gram memory. The question was what that design does at the other end of the scale: a model small enough to train on one GPU, with its scope deliberately narrowed to two languages (English and Spanish) plus code and structured text, to see how far a specialised, bounded model can go. The project began shortly after the public release of DeepSeek-V4.1-Flash (its technical report followed on 17 September [14]); nothing here should be read as a reproduction of its results.

Agents. The code, the corpus and most of the measurements were produced by coding agents (Claude Code, with Claude Opus 5 and, later in the project, Opus 5.5) working under the author’s direction, in three roles with separate sessions and separate responsibilities: a *model and training* agent (architecture, tokenizer, trainer, pretraining runs); a *data* agent (sources, cleaning, deduplication, contamination checks, packing); and a *capability lab* (an evaluation suite in its own repository, which the training side was not allowed to touch). They exchanged work through files and acceptance gates, never through shared code. The author’s role was to set the goals and the scope, decide between the options the agents laid out, and refuse to accept a result until it had been checked. A second assistant, GPT Astra, worked with the author outside the coding sessions: it helped write the specifications given to the agents and review the results of each experiment before the next one was decided. The long, formal briefs quoted below are the product of that collaboration; the short instructions in between are the author’s own.

Working rules. A handful of rules, set in the first brief and repeated whenever they were broken, shaped everything that follows:

- *Read the reference, do not infer it.* “Do not invent how the causal encoder–decoder works from this prompt alone; if my specification contradicts the official code, follow the code and tell me.” It did contradict it (Section 4).
- *Correctness before speed*, and “do not add GPUs to hide a code problem”.
- *Stop and report instead of patching silently:* e.g. if the tokenizer trainer cannot reach exactly 65,536 ids, stop; do not pad.
- *Every gate ships with a test that breaks it.* A check that cannot fail is not a check.
- *No claim without a control.* When the second run beat the first by 0.16 nats, the agent’s reading (“the experts are worth 0.16 nats”) was blocked until a factorial separated the two things that had changed (Section 5.2).
- *Retractions are commits.* When a claim did not survive measurement it was withdrawn in the repository, with the evidence, rather than quietly edited. The history has about 130 commits up to the end of V3; several of them are titled as corrections.

Timeline. Table 1. Every number in Sections 4–6 comes from a report, log or benchmark file in the repository, or from the run directories on the training machine; where a figure was recomputed after the fact we say so.

4 Model

The model was developed under the codename mini-v41 and is released as Ines-1. Ines-1 is a reduced re-implementation of the published architecture of DeepSeek-V4.1-Flash [14], written from its public

12 Sep	Brief. Reference code read; 482M prototype written and tested on the desktop GPU.
13 Sep	Engram ablation, CED scaling, 50M-token MoE stability run. Tokenizer V1 frozen. 1B-token corpus delivered, rejected by a gate, re-packed, accepted. Learning-rate pilot. V1 launched.
14 Sep	V1 finishes through a network outage (7.4 h). Recurrent-depth experiment. Discovery that V1 never had working experts. V2 launched.
15 Sep	V2 finishes. Factorial to attribute its gain; external benchmarks.
16–17 Sep	Supervised fine-tuning experiments on V2, which motivate a larger pretraining.
18–20 Sep	V3 corpus: 10B tokens, 16/16 preflight checks.
20–24 Sep	Learning-rate pilot with a written selection rule; V3 pretraining (71 h).
25–28 Sep	V3 export and evaluation; V2 vs V3 under identical fine-tuning; data-control run.

Table 1: From brief to the end of V3.

configuration and minimal inference code (no weights were used). The brief asked for a ladder — a ~ 500 M prototype, then ~ 1.6 B, perhaps more later — and explicitly not for an attempt at the 552B original. Both presets landed on their parameter targets without resizing: 482,583,360 parameters (196M active) for the prototype and 1,590,066,240 (405M active) for the model in this paper (Table 2).

Parameters	1,590M total, 405M active per token
Layers	8 encoder + 8 decoder, one residual stack; hidden 1024
Attention	16 heads, 4 KV heads (GQA [1]), head dim 64, RoPE [53] $\theta = 10^4$, context 2,048
MoE	all 16 layers: 16 routed experts (FFN 1536), top-2, + 1 shared expert [12]
Router	sqrt-softplus scores [15], bias-corrected selection [16], fp32; aux 0.01, z 0.001
Engram	hashed 2/3/4-gram memory at encoder layers 1 and 5; 10^6 rows/layer, 128M params
Tokenizer	byte-level BPE [45, 48], 65,536 tokens, 3.62 bytes/token (Spanish 4.58)

Table 2: Ines-1, 1.6B preset.

Causal encoder–decoder (CED). Encoder layers are causal self-attention with their own K/V. Decoder layers attend, in one softmax, to their own sliding window ($W = 128$; cf. [3]) and to a *global* K/V projected once from the final encoder state. This is one residual stack, not two towers, and the global K/V is shared by all decoder layers rather than projected per layer: the brief said the opposite, the reference code (`kv_source_layer_ids` and the projection in `model.py`) settled it, and both variants were implemented with the reference’s as the default.

A decoder output at position t depends on decoder inputs only within the receptive field $R = W + (D - 1)(W - 1)$; the bound is tight (checked on a toy model: error 10^{-7} at span R , 4×10^{-6} at span $R - 1$). For prefill this means the encoder runs over all L positions but the decoder only over the last R (or, for n logits, $R + n - 1$): *for last-logit scoring, decoder-side prefill work is bounded by a fixed trailing receptive field rather than growing with prompt length*. The reference model leaves this asymmetry unused at prefill; we exploit it for scoring, where only the logit at the last position is needed (exactness verified to 2×10^{-7} in fp32).

The first 1.6B preset used $W = 512$, which gives $R = 4,089$ — longer than the 2,048-token training context, so the truncation could never engage. This was found by measuring, not by reading (Table 3); the preset was changed to $W = 128$ ($R = 1,017$), which also matches the reference configuration’s `sliding_window` [15].

Prompt length	$W = 512$ ($R = 4,089$)		$W = 128$ ($R = 1,017$)	
	decoder skipped	speed-up	decoder skipped	speed-up
1,024	0.0%	1.00×	0.7%	1.01×
2,048	0.0%	1.00×	50.3%	1.03×
4,096	0.2%	0.96×	75.2%	1.10×
8,192	50.1%	1.15×	87.6%	1.34×

Table 3: Truncated prefill on the 1.59B model (H200, bf16). Speed-ups are modest because a forward at these sizes is bound by kernel launches, not arithmetic ($16\times$ more tokens cost $2.1\times$ more time on the prototype); the saving grows with length, reaching $1.78\times$ at 16k tokens on the prototype against $1.83\times$ predicted.

Mixture of experts. Experts are stored as one grouped bank and run with a grouped matrix multiply (`torch.nn.functional.grouped_mm`); a per-expert loop is kept as a reference implementation. The grouped path was $2.0\text{--}2.8\times$ faster in training throughput (18,246 vs 6,539 tokens/s at 1.59B, sequence 2,048). It is also where the most expensive bug of the project lived (Section 5.1).

Engram. Each Engram layer hashes the last 2, 3 and 4 tokens (two hash heads each, distinct prime moduli per layer, order and head), retrieves embedding rows, and adds a gated value to the residual stream. Its gradients are two orders of magnitude smaller per element than those of the dense weights, so the tables train with a learning-rate multiplier and no weight decay. The multiplier was chosen by an ablation on the prototype (Table 4): $8\times$ gets 95% of the gain of $16\times$. The initialisation had to be seeded by parameter name first — otherwise switching Engram off reshuffled the initial values of 82 of 95 shared tensors and the ablation would have compared two different models.

Engram	validation loss	Δ vs off	mean gate
off	3.549	—	—
LR $\times 1$	3.530	-0.020	0.78
LR $\times 4$	3.519	-0.030	0.81
LR $\times 8$	3.508	-0.041	0.80
LR $\times 16$	3.506	-0.043	0.80

Table 4: Engram on the 482M prototype, 8.2M tokens per arm, one seed. The ordering holds at all 11 evaluation points of every arm. Cost: $+0.7$ GB of memory and about 4% of throughput.

Tokenizer. Byte-level BPE with exactly 65,536 ids (16 reserved), trained on a 3.5 GB sample of the pretraining mixture and frozen by hash before any corpus was packed, because Engram hashes token ids and any later change would invalidate its tables. The prototype’s 32k tokenizer, trained on English stories and Wikipedia, had left half of the 1.59B model’s embedding rows unreachable. The new one encodes 61% more bytes per token overall: $+63\%$ on Spanish, $+59$ to $+125\%$ on code, $+6\%$ on English prose. Special-token strings that appear literally in web text (`<eos>`, `<user>`, ...) are tokenised as control ids even with special tokens disabled, so 4,587 such documents were removed from the corpus.

What was left out. Hyper-connections (mHC; [58, 60]), compressed and sparse attention, FP8/FP4, the 1M-token context, and multi-token prediction [20]; and we trained with AdamW

rather than optimisers such as Muon [31]: either serving optimisations for very long contexts or out of scope for one GPU in bf16. Of these, hyper-connections are the largest architectural omission; the others are mostly about efficiency at a scale we do not reach. One consequence is worth stating, given why the architecture was chosen: of its KV-cache design we kept the structure (shared global K/V, windowed decoder) but not the compression, so our cache costs 5,880 bytes per token at 2,048 tokens, against 890 reported for the reference with FP4 and compressed sparse attention.

5 Pretraining: three runs

The three pretraining runs are Pretrain-V1, Pretrain-V2 and Pretrain-V3 (V1–V3 for short in this section). All three use the same model, AdamW [33] ($\beta = (0.9, 0.95)$), weight decay 0.1, clip 1.0), 65,536 tokens per step (micro-batch $2 \times 2,048$, accumulation 16), 500 warm-up steps and cosine decay to $0.1 \times$ the peak, on one H200 at about 40k tokens/s. Documents are packed with an end-of-document token and plain causal attention across them. Validation loss is reported per domain and as a mixture weighted like the corpus.

5.1 V1: a 1B-token corpus, and a model without experts

Corpus. 1,000,058,897 content tokens (plus one end-of-document token per document): general English 45%, Spanish 20%, code 15%, math 10%, reference 5%, structured 5%, from FineWeb-Edu [42], FineWeb-2 (Spanish) [43], OpenWebMath [41] and Common Pile [27] (Stack-Edu [4], StackExchange, Wikimedia, DOAB, peS2o [52]). Language was checked with GlotLID [28] (the FastText identifier [26] is non-commercial). Deduplication removed 1.36% (0.73% exact, 0.63% near-duplicate with MinHash at Jaccard 0.7); the contamination scan against the evaluation sets found one problem copied into a GitHub repository, which was removed.

Two lessons from the data side were invisible in every size metric. The public shards are not shuffled: the first Stack-Edu shard is 100% Markdown and the first StackExchange shard is 81% one academic forum, so “take the first N shards” produces a code domain without code and a math domain without math, with byte totals that come out perfect. Shards had to be mapped by reading their headers before choosing. And deduplication deletes by document id across all shards, so a document copied into a second domain must get a new id — otherwise “drop one copy” drops both, which is how a first build lost every native structured document.

The corpus went to the training side through nine acceptance gates (tokenizer hash, manifest hash, shard checksums, token-id range, packing length, validation isolation, a forward smoke test, domain interleaving and validation packing). The first delivery failed: the packer had not counted the end-of-document tokens, and 62.5% of the 2,048-token sequences overflowed. It was re-packed (99.6% packing efficiency) and passed all nine.

Run. A pilot on the real corpus picked a peak learning rate of 4×10^{-4} . V1 trained 15,337 steps in 7.4 hours — straight through a network outage that cut the machine off for more than an hour — and reached a validation loss of 2.859. Router metrics were healthy throughout: all 16 experts used, routing entropy 0.9997 of the maximum, a final load ratio of 1.75.

The experts were never there. A day later, a screening experiment split each expert into two narrower ones, which should change the loss. It changed by exactly 0.0000. Zeroing every routed expert in the trained checkpoint left the validation loss at 3.0942, unchanged in the fourth decimal: all 768 routed expert tensors were exactly zero, and their optimiser moments too. The

weight initialiser dispatched on `nn.Linear` and `nn.Embedding` and skipped the raw parameters of the grouped expert bank, which `torch.empty` had left at zero. Zero is an absorbing state for SwiGLU [49]: the gradient of the output projection is proportional to the activation, and that of the input projections to the output weights, so neither can ever move. V1 was in effect a 254M-active model with a dense shared expert, Engram and an idle router.

What makes this worth reporting is the list of checks that passed:

- parameter count and configuration fingerprint — they count shapes, not values;
- the loss curve — the dense path, the shared expert and Engram were learning;
- router metrics — computed upstream of the experts, so they measure the router’s choices, not whether the chosen experts do anything; V1’s load ratio (1.75) even looked *better* than that of the healthy model that followed (7–10);
- equivalence tests between the grouped and reference backends (relative error 0.0) — they copied weights from one backend into the other, so they never ran a freshly built grouped bank.

The fix is two lines; the lesson is a startup assertion on values, not shapes. V2 refuses to start if any 2-D weight is all zero, checks before step 1 that zeroing the experts changes the logits, and aborts if an expert stops receiving gradient.

The bug invalidated V1 as a mixture-of-experts baseline, the learning-rate pilot, an experiment with recurrent depth (looping three encoder layers twice; it “bought nothing”, on a model without experts) and the screening that found it. The author’s instruction was to treat all of it as “experimental history, not a baseline”; the weights were archived and deleted.

5.2 V2: the same corpus, with experts

V2 repeats V1 with the experts initialised, deterministic name-seeded initialisation, a fix for a sequence the sampler silently dropped at the end of each epoch, and a new learning-rate sweep on the real model (75M tokens per arm on the real schedule): 2×10^{-4} , 4×10^{-4} , 6×10^{-4} and 8×10^{-4} reached 4.516, 4.163, 4.064 and 4.008. The best value was again at the edge of the range; 8×10^{-4} was taken knowing the optimum was not bracketed. V2 took the same 7.4 hours and reached 2.700, better than V1 on every domain (Table 5).

	code	structured	math	reference	English	Spanish	mixture
V1 (experts dead), LR 4×10^{-4}	1.692	2.254	2.361	2.948	3.354	3.602	2.859
V2, LR 8×10^{-4}	1.564	2.109	2.233	2.780	3.162	3.404	2.700

Table 5: Validation loss after 1B tokens, on the V1 validation set. These numbers are not comparable with those of Table 9, which use the V3 validation set.

Attribution. V1 and V2 differ in two things at once — working experts and a doubled learning rate — so the 0.16-nat gap says nothing about either. A 2×2 factorial at 100M tokens (experts dead or alive, 4×10^{-4} or 8×10^{-4} ; Table 6) separated them: doubling the learning rate was worth 0.10–0.13 nats, the experts 0.05–0.07, and the two interact (the experts are worth less at the higher rate). At this scale the learning rate mattered more than the experts.

	LR 4×10^{-4}	LR 8×10^{-4}	LR effect
experts dead	3.886	3.759	-0.127
experts alive	3.813	3.712	-0.101
expert effect	-0.073	-0.047	interaction +0.027

Table 6: Validation loss at 100M tokens, one run per cell.

What the experts and Engram do. Routing did not specialise by domain: the routing entropy per domain is 1.789 out of a maximum of 1.792, and every expert takes 5.5–8.0% of every domain against 6.25% for uniform routing. Engram’s contribution, measured by switching it off at evaluation time, settles at about 0.14 nats and stops growing after 500M tokens (Figure 1). In V1, without experts, the same probe kept rising, and at the time we read that as Engram becoming more valuable while its gate closed; with the experts back, a plausible reading is that part of what Engram did in V1 was standing in for them.

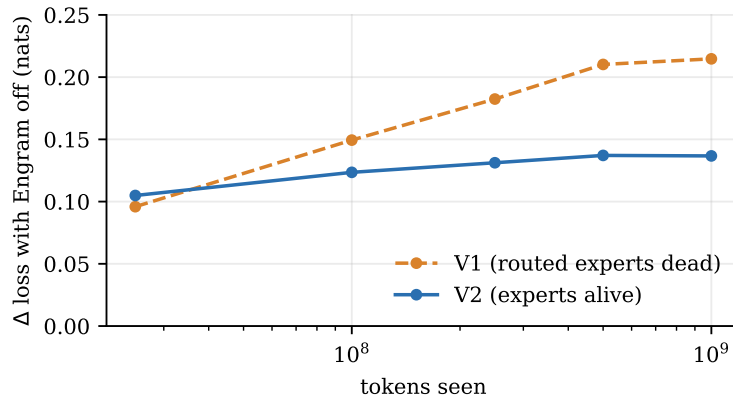


Figure 1: Loss increase when Engram is switched off at evaluation time, through pretraining.

Outside benchmarks. On zero-shot HellaSwag, PIQA, ARC-Easy and ARC-Challenge, V2 is better than V1 on all four, but only HellaSwag significantly ($0.264 \rightarrow 0.282$, $p = 0.005$). Both models are close to chance on ARC; at 1B tokens the main thing a 1.6B model learns is the surface of language.

5.3 V3: a new 10B-token corpus

Fine-tuning experiments on V2 (not reported here) kept running into the limits of its pretraining: what fine-tuning taught on in-distribution data did not transfer, and the more it taught, the more general loss it destroyed. V3 asked the simplest question first — “isolate the effect of giving the same model about ten times more pretraining” — with no architectural change. As built, however, V2 and V3 differ in both training horizon and data mixture (and therefore schedule), so their comparison measures the combined effect of the revised pretraining regime rather than an isolated token-scaling intervention.

Corpus. 10,000,031,033 content tokens, assembled through two gates: one on what was really available locally (it retracted seven figures from a first estimate, among them that 8.98B tokens of

code were available, not 0.85B), and one on what would remain after deduplication. The mixture (Table 7) was the author’s call: code was kept at 8.5% by choice, not by availability, to favour broad natural language; structured text was cut to 1% rather than re-mining documents the quality filter had rejected (“do not re-mine rejected documents just to hit a quota”). That 1% is three quarters real JSON/YAML/XML mined from Stack-Edu and one quarter synthetic JSON-schema documents, generated with their format mix and size matched to the mined ones. Deduplication removed 1.09% of documents (the largest overlap between two sources: math StackExchange and OpenWebMath, 3% of the latter); 407 validation documents that shared a source group with training documents were removed from validation. A 16-point preflight checked the result before any GPU time was spent.

Domain	Sources	Tokens	Share
General English	FineWeb-Edu, FineWeb	4,650M	46.5%
Spanish	FineWeb-2 (spa_Latn)	2,000M	20.0%
Reference	Common Pile: Wikimedia, DOAB, peS2o, PEPs	1,500M	15.0%
Math	OpenWebMath, math StackExchange	900M	9.0%
Code	Common Pile Stack-Edu	850M	8.5%
Structured	Stack-Edu (mined) 75M + synthetic 25M	100M	1.0%
Total		10,000M	

Table 7: V3 corpus as trained. Every domain lands within 0.01% of its planned budget.

Choosing the learning rate, with the rule written first. The pilot ran each candidate for 400M tokens on the real 152,588-step schedule. The selection rule (disqualify unstable arms, then lowest loss, and on a tie under 0.01 nats take the lower rate) was written down before any per-domain or router figure was read — though after two of the three global losses were visible, which the rule’s document says. 6×10^{-4} won on the global loss, all six domains and router balance. A fourth, lower arm was then run to bracket the optimum, and on the 80-sequence evaluation used during training it beat 6×10^{-4} by 0.0055 nats — a tie, which by the rule meant switching. Re-evaluated on the whole validation set the order reversed (Table 8; 6×10^{-4} also wins four of the six domains), 4×10^{-4} also had the worse router balance, and the run, already under way, was not restarted. We report it as a declared departure from the rule.

Peak LR	in-training eval (1% of val.)	full val., uniform	full val., mixture	router load ratio
4×10^{-4}	2.958	3.018	3.440	7.60
6×10^{-4}	2.963	3.016	3.439	7.46
8×10^{-4}	2.975	3.027	3.449	9.04
1.2×10^{-3}	3.007	3.066	3.490	10.70

Table 8: V3 learning-rate pilot, 400M tokens per arm. The in-training evaluation reads 1% of the validation set and swings by up to 0.13 nats between reads.

Run. 152,588 steps, 9.96B tokens, 71.2 hours on one H200 at a median of 40,232 tokens/s (about 20% model-FLOPs utilisation by the trainer’s own count, against the dense bf16 peak) and 41 GB of memory. Two defects were found during the run and documented rather than fixed by restarting: the number of steps was computed from content tokens only, so the last 49M tokens (0.49%) were

never seen; and a milestone constant inherited from V1 meant the 1B, 2B, 5B and 10B checkpoints were never exported — neighbouring checkpoints were saved instead, and the 1B point of the V3 curve is lost.

Result. Validation loss 2.655 against 3.041 for V2 on the same V3 validation set, better on every domain (Table 9, Figure 2). At equal tokens V2 is slightly ahead, as expected: it is further along a schedule ten times shorter.

	English	Spanish	math	reference	code	structured*	mixture
V2, 1B tokens	3.346	3.407	2.537	2.835	1.477	2.461	3.041
V3, 5B tokens (mid-run)	3.137	3.239	2.409	2.784	1.506	1.925	2.888
V3, 10B tokens	2.898	2.982	2.208	2.551	1.316	1.810	2.655

Table 9: Validation loss on the V3 validation set. *V3’s structured domain contains synthetic documents that V2 never saw, so that column is not a fair comparison; without it the gap is 0.383. Perplexity falls from 20.9 to 14.2.

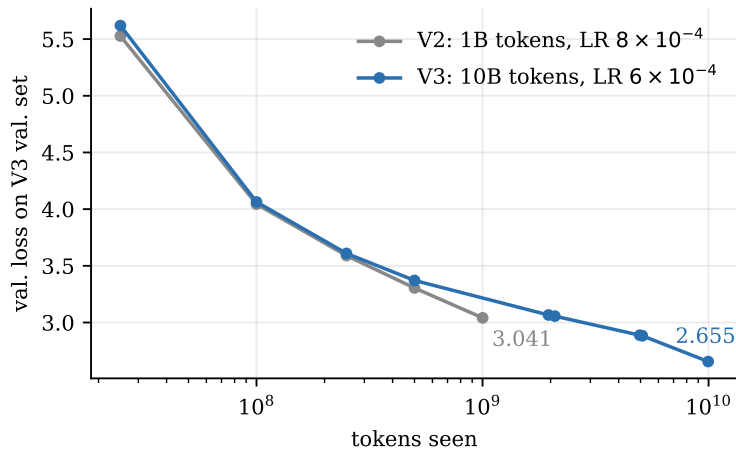


Figure 2: Validation loss on the V3 validation set against tokens seen (log scale).

Capabilities. Table 10. The V3 regime (new corpus, ten times the tokens, longer schedule) moved commonsense benchmarks (HellaSwag [59] $0.28 \rightarrow 0.34$, PIQA [6] $0.58 \rightarrow 0.64$, both $p < 0.01$) and made the base model sensitive to a JSON schema given in the prompt ($0.05 \rightarrow 0.75$). It did nothing for ARC [10] or tool use: the base model hallucinates a tool call in every case of the tool-use probe, at 1B tokens and at 10B. The schema result has a confound we have not yet removed: V3 is also the first corpus with synthetic schema documents. A control run — V2’s exact recipe on a 1B-token subset of the V3 corpus — was trained to separate better data from more data, but has not been evaluated, so we make no causal claim about which of the two produced the gains: Pretrain-V2 and Pretrain-V3 differ in both training horizon and data mixture, and their comparison measures the combined effect of the revised pretraining regime.

Does the better backbone survive fine-tuning? We fine-tuned V2 and V3 with exactly the same generic recipe (JSON and tool-use tasks, 2k and 10k examples; the parent model is the only

	V1	V2	V3	V2 vs V3
HellaSwag (acc_norm)	0.264	0.282	0.345	$p < 10^{-20}$
PIQA	0.560	0.582	0.635	$p = 0.001$
ARC-Easy	0.311	0.319	0.315	n.s.
ARC-Challenge	0.245	0.267	0.288	n.s.
JSON discrimination (chance 0.389)	–	0.444	0.628	$p < 10^{-5}$
JSON schema sensitivity	–	0.05	0.75	$p < 10^{-4}$, $n = 20$
hallucinated tool-call rate	1.00	1.00	1.00	–

Table 10: Base-model capabilities (zero-shot; lm-evaluation-harness 0.4.13 [5, 19] for the first four rows). V1 is the model without working experts.

variable). V3 with 2k examples beats V2 with 10k on 10 of 15 metrics, and the gap is largest out of distribution (Table 11). The price is forgetting: fine-tuning raises V3’s general validation loss about twice as much as V2’s, and V3 after 10k examples lands where V2 started. Mixing 10% pretraining data back into the fine-tuning loss (“replay”) cut V3’s forgetting by 54%, in every domain, with no loss out of distribution, though greedy generation became more repetitive. Nothing taught depth-3 nesting to either model. This is the point where the work turned to decision fine-tuning.

Fine-tuning examples	2k		10k	
	V2	V3	V2	V3
JSON valid, in-distribution	0.221	0.629	0.576	0.886
JSON valid, unseen vocabulary	0.093	0.486	0.087	0.365
correct tool, unseen tools	0.317	0.638	0.287	0.455
copies an unseen key exactly	0.221	0.720	0.142	0.573
depth-3 nesting	0	0	0	0
forgetting (+ general val. loss)	+0.088	+0.158	+0.226	+0.404

Table 11: The same fine-tuning recipe on the two parents.

6 Silent failures

The most useful thing this project produced may be its list of failures that did not look like failures. None of them crashed, and most passed a check that had been written for them. Table 12 collects them. The common thread: each check verified that something *existed* or had the right *shape*, not that it had the right *value*; and several reported a reassuring summary (“PASS”, “deviations: []”, “isolation verified”) while doing nothing. What caught them was almost always a number that was too clean — a delta of exactly zero, a byte total exactly on target — or a person refusing to accept a result without a control.

7 Typed decisions

A case is a state s and questions $q_1, \dots, q_m$. A **choice** question has named criteria $\{k_i : d_i\}$; a **score** question has an ordered list of level descriptions; a **noul** question optionally has descriptions of the true and false sides. The response gives, per question, the probabilities over its keys (for **noul**, $P(\text{true})$), the argmax and, for **score**, the expected level.

Failure	What it looked like	What caught it
Routed experts initialised to zero (V1)	falling loss, healthy router metrics	an expert split that changed the loss by exactly 0.0000
Gradient accumulation averaged per-micro-batch means	nothing	a test with unequal micro-batches: up to 22.6% gradient error
Packer ignored end-of-document tokens	exact token totals	gate 5: 62.5% of sequences too long
Validation-isolation gate (V1 corpus)	PASS	reading its counters: it had scanned zero documents
Unshuffled public shards	perfect byte totals per domain	mapping shard contents before choosing
Deduplication by shared document id	structured domain at 27M tokens instead of 91M	manual inspection of the domain's documents
Hardware utilisation (MFU)	plausible percentages	recounting: once $3.6\times$ too high (experts counted as dense), once $2\times$ too low (sparse peak used)
Speed-up of truncated prefill	$2.36\times$	re-timing: one path had been measured warm, the other cold
Final save overwrote the 1B milestone (V1)	a 1B milestone file	its contents were those of the 500M point
V3 preflight identity check	"ALL HASHES MATCH"	it had verified the V1 corpus; now it must print this corpus's own token count
V3 validation split	"isolation verified"	checking globally, not per domain: 407 documents leaked
V3 selector	"deviations: []"	comparing against the plan: it had read V1's constants
Learning-rate comparison on 1% of validation	a winner by 0.0055 nats	full re-evaluation reversed it
Milestone constant inherited from V1	a healthy run	noticing that no milestone after 500M had been exported

Table 12: Failures that passed silently, in the order they were found.

Evaluation data. *typed-decisions* [32] (LocalLLaMA, Apache-2.0, pinned at revision c76749ec) has 1,200 train and 400 test cases (6,000 / 2,000 questions) over four workflows (agent trace observability, customer service, invoice processing, security incidents); gold labels come with the distribution of a teacher sampled three times. We machine-translated train and test into Spanish (only text values; keys and structure verified identical, otherwise retried or dropped: 1,985 test questions kept), and also use an independent Spanish translation (*telepatia-ai/typed-decisions-pt-es* [56]) as a second Spanish test. *jev-decisions-v1* [47] (CC-BY-4.0) provides agent tool-selection cases with 2–26 options.

8 Reading decisions from a causal LM

Each question becomes one prompt, a user turn in the model’s chat template:

```
Case:
<state, as text or JSON>

Question: <instructions>
Options:
A) <criterion or level description>
B) ...

Answer only with the letter of the correct option.
```

(and the same in Spanish). The answer is the softmax, restricted to the option letters, of the logits at the first assistant position. The token id of each letter is obtained by rendering an assistant reply that contains it, so it is the token the model would emit; the code asserts the template prefix is stable and the letters are distinct. `noul` is presented as yes/no, `score` keeps its order, `choice` options are shuffled at training time (position bias). Long states are trimmed in the middle and each option capped at 400 tokens (shrunk if needed). At inference the logits come from `prefill(num_logits=1)`: 33 ms instead of 44 ms for a full forward at ~ 850 tokens.

Supported interface. One model prompt per typed question. Several questions of a case can be batched by the API (one prompt each, scored together), but that is batching, not multi-question modelling: the model was never trained to condition jointly on several typed questions in one prompt (Section 9.1 measures what happens when it is asked to). Canonical inference, used for every number in this paper, is bf16 weights under bf16 autocast.

9 Decision fine-tuning

Recipe. Full fine-tuning, bf16 autocast over fp32 weights, AdamW at 10^{-5} , accumulation 16, linear warm-up over 5% of the steps and linear decay to zero. The loss is the cross-entropy of the letter distribution against the teacher distribution when the data has one [25], one-hot otherwise, plus $0.01 \times$ the MoE auxiliary loss. Validation is one file per data family, all weighted equally; the best epoch is kept and the test sets are run once at the end.

Lineage and names. Pretrain-V3@9.96B tokens $\rightarrow$ Decision-FT Stage 1 $\rightarrow$ Decision-Mix Stage 2 $\rightarrow$ the release, *Ines-1*. (Run names in the repository: `1p6b-pretrain-v3`, `decisions-v2-e8`, `decisions-v3-mix-from-v2`; each stage’s parent-weights hash matches the previous stage’s output.)

- **Decision-FT Stage 1:** typed-decisions train (English) + its Spanish translation + 9.5k jev-decisions cases (20k questions). 8 epochs scheduled, **7 effective:** the epoch chosen on validation is the checkpoint kept. (An earlier 3-epoch run is reported as “Stage 1, 3 epochs”).
- **Decision-Mix Stage 2:** a balanced mix of 86,436 questions (Table 13) in which no family exceeds 29%. tasksource rows [51, 55] are kept only when the dataset’s own `license_use` metadata says `commercial` (an annotation of the dataset, not a legal guarantee), at most 80 questions per source, and no source derived from typed-decisions or jev-decisions is used. Exact duplicate (state, question) pairs are dropped, test pairs reserved first. The independent Spanish translation turned out to contain 117 cases of our *validation* split (detected by a fingerprint of the non-text leaves of the state, which survives translation) and they were removed. Two runs: from Stage 1 (2 epochs: the release) and, as an ablation, from Pretrain-V3 (3 epochs). Provenance, as it happened: Stage 2 ran from a commit with the mix builder still uncommitted; committed afterwards, it rebuilds all 16 data files byte for byte.

Family	Questions	Source (licence)
typed-decisions EN	5,420	LocalLLaMA/typed-decisions (Apache-2.0)
typed-decisions ES (ours)	5,335	machine translation, structure-checked
typed-decisions ES (independent)	5,415	telepatia-ai/typed-decisions-pt-es (Apache-2.0)
tool selection	17,898	samatv256/jev-decisions-v1, clean-50k (CC-BY-4.0)
tasksource	24,959	tasksource/tasksource-jev-typed-decisions, 366 sources (per-row)
synthetic, teacher soft labels	18,001	n4ze3m/typed-decisions-synth (MIT) [39]
synthetic	9,408	helmo/synthetic-typed-decisions (MIT) [24]
total	86,436	choice 61%, noul 21%, score 18%

Table 13: Decision-Mix Stage 2 data.

Results: single-question specialist protocol. Table 14. One prompt per question; the model is a specialist, trained on the benchmark’s `train` split (same four workflows, teacher soft labels). No test case, state, normalised state text or state structure appears in any training file.

	typed EN	typed ES	typed ES (indep.)	tasksource
Pretrain-V3, untouched [†]	29.4	33.6	—	—
Stage 1, 3 epochs [†]	65.1	64.1	—	—
Decision-FT Stage 1 (7 effective epochs)	75.5	74.7	74.6	38.5
Engram removed at inference	74.0	71.9	71.7	37.0
Decision-Mix Stage 2 from Pretrain-V3 (ablation)	72.6	70.4	71.2	50.7
Ines-1 (Stage 2 from Stage 1)	76.4	75.4	76.3	49.5
Engram removed at inference	74.7	72.8	73.5	47.0

Table 14: Accuracy (%), single-question specialist protocol, canonical environment (bf16 weights + bf16 autocast). typed: 400 cases, 2,000 questions (1,985 in our Spanish translation); the independent Spanish test is the same 400 cases translated by another system; tasksource: 2,206 held-out questions from sources that are also in the Stage 2 mix (new examples of trained tasks). Release, English: 1,528/2,000, 95% CI 74.2–78.5 (bootstrap over cases [17]); choice 71.5, score 74.5, noul 83.8; Brier 0.065, KL(gold||p) 0.121, log loss 0.635. [†]Measured during training (fp32 master weights under autocast), not re-run.

Brier [7] and KL are computed against the teacher’s gold distribution with the definitions of the dataset card (they reproduce its *Uniform* reference row exactly: KL 0.444, Brier 0.238). Our ECE [22, 38] (10 equal-width bins of top-1 confidence; 0.132) is reported as *our-ECE*: no public code reproduces the card’s reference ECE, so it is not compared with published values. By the dataset’s teacher-agreement flag, the release scores 86.4% where the teacher’s three samples agree ($n = 1,188$) and 61.7% where they do not ($n = 812$).

Training longer mattered more than we expected: with 8 epochs scheduled (7 effective) the slower linear decay kept the model learning (validation $0.525 \rightarrow 0.692$; tool-selection validation $35.4 \rightarrow 56.9\%$).

What Stage 2 bought, and why the order of the stages matters. Continuing Stage 1 on the balanced mix for two epochs did *not* significantly change typed-decisions accuracy in English (+0.9 points, paired exact McNemar [35] $p = 0.24$) or in our Spanish translation (+0.7, $p = 0.32$), and moved the independent translation by +1.7 ($p = 0.023$). It gained 11.0 points on held-out tasksource questions ($p = 10^{-14}$). Those are new examples of tasks present in the mix — all 350 test sources are among its 366 training sources — so this is generalisation to new examples, not to new tasks; we have no unseen-task evaluation. Training on the same mix straight from Pretrain-V3 reached the same tasksource accuracy but stayed about 4 points lower on typed-decisions, where the target workflows are 19% of the mix: *target task first, then diversify*. The two Spanish translations agree within 1 point for every model.

9.1 Full-context stress tests

The dataset’s public runs send the whole case — the state and its five questions — in one request, and the dataset card notes that request shape can change results. Ines-1 has no readout that answers several questions independently in one forward (that would need a tree attention mask), so we ran two stress tests (Table 15): *context*, where every prompt holds the state and all five questions and ends with “Answer question k ”, with one forward per question; and *joint*, one prompt whose answers are read in sequence as “ k : letter”, each conditioned on the model’s own previous answers. Neither reproduces the public protocol or a single forward over five independent decisions, and neither is a DecisionEval score.

model	readout	accuracy	95% CI	noul	Brier
Ines-1	single question	76.4	74.2–78.5	83.8	0.065
	context (one forward per question)	41.5	39.5–43.6	49.2	0.266
	joint (one prompt, sequential)	37.1	34.9–39.3	46.0	0.246
Decision-FT Stage 1	single question	75.5	73.4–77.6	83.8	0.070
	context	42.8	40.7–45.0	49.7	0.269
<i>dataset reference: Prior (train label frequencies)</i>		<i>47.0</i>			<i>0.189</i>

Table 15: typed-decisions English test, 2,000 questions. Spanish behaves the same (release: 75.4 $\rightarrow$ 43.3 *context*, 38.4 *joint*). Paired single vs context: $p < 10^{-100}$ for both models.

Both readouts fall below the dataset’s prior, and Stage 1 falls as far as the release, so in this experiment the drop is not caused by Stage 2. The prompts are well formed (the five questions and the final instruction are intact, at most 942 tokens), and the failure is not mechanical (only 21 of 400 cases get the same letter for all five questions); the model is disoriented by several questions — in *context* mode it answers “no” to 544 of 600 yes/no questions. In this experiment, **task competence**

did not imply interface generalisation: the model learned isolated decisions well, but not the structural operation of answering one of several questions about the same case. We do not claim this holds for other models or training recipes.

External figures, kept apart. We do not rank our model against published systems: modes, request shapes and training exposure differ. For reference only: among *general, zero-shot* systems, DecisionEval’s independent run [13] of TypeSafe Jev 1.13 on this test (frozen 2026-09-20; data files identical to ours) reports 74.0% [72.1–75.9], Brier 0.148, with the whole case per request. Among *specialists trained on the benchmark’s train split* — this model’s category — the dataset card’s separate table lists mostly self-reported results from 0.483 to 0.801, and notes that scores above its teacher self-agreement (0.735) indicate learning the teacher’s quirks. Under our own harness we also measured other models in earlier experiments (Julia-1 [54], 144M, through its own server: 72.6%; DiffusionGemma [21] 26B through OpenJev [46]: 67.1%, 70.0% with thinking); these are context, not a controlled comparison.

10 Adapting to a domain

The use case that motivated this work is a narrow one: decisions over the records of a single business (routing, classification, triage), in Spanish, with few hand labels: training labels come mostly from a teacher model, and a small human-labelled held-out set is used only for testing. This section is a guide to the recipe that worked; it does not describe any particular organisation’s data.

1. Write the questions first. Each decision becomes a typed question with criteria written the way a person would explain them. Criteria text is part of the input: vague criteria cap accuracy whatever the model.

2. Get labels from a teacher, as distributions. Take unlabeled real records (a staging copy of the operational database is enough), and have a large model answer the typed questions through the same protocol, keeping its *probabilities*. Labelling thousands of real, unlabeled records this way was the largest gain in our domain experiments, and training on the distributions transfers the teacher’s uncertainty on ambiguous cases instead of forcing a coin flip into the labels. We did not run an ablation of soft against hard targets, so we do not claim that the distributions improve accuracy or calibration; their contribution was not isolated. A teacher that reasons before answering was a better judge (higher accuracy on the test) but not a better teacher.

3. Split validation by organisation, not by record. Records of the same customer or account share names and wording. A random split leaked them and inflated accuracy by more than 20 points in our case; grouping by the owning organisation gives honest numbers.

4. Choose the epoch on the mean per task. With several decisions of very different sizes, a pooled validation score lets the largest one hide the collapse of a small one. Average per task.

5. Synthetic cases, validated. Extra cases generated by an LLM for rare classes are kept only when a second model labels them consistently with the intended class. Their independent effect on the domain test was not measured: they were always part of the training data, with no run without them, so we attribute no improvement to them.

6. Keep a few real hand-labelled cases as the test , used once.

Results: zero-shot versus adapted. On an internal held-out domain benchmark (nine Spanish subtasks, 1,026 questions; not released, aggregate figures only), the generic decision model does not transfer uniformly zero-shot. Performance is highly task-dependent (Table 16); we report macro balanced accuracy over subtasks because several are imbalanced. On one imbalanced yes/no subtask (21 positives, 122 negatives), no untuned checkpoint separates the classes: AUROC is 0.48–0.55 for Pretrain-V3, Stage 1, Stage 2 from pretraining and the release, and the mean predicted probability is the same for positives and negatives. Stage 1 answers “no” to 133 of 143, and the Stage 2 mixture shifts the predicted class prior to “yes” (137 of 143): Stage 2 changes the prior, it does not destroy a discrimination that did not exist. Stage 1 + domain fine-tuning reaches AUROC 0.883 on that subtask. This motivates explicit domain adaptation: **generic typed-decision training teaches a reusable decision interface but does not replace domain adaptation.** The historical domain-adapted checkpoints in Table 16 start from Pretrain-V3 or Stage 1; the released checkpoint itself is adapted below (Ines-1 + domain fine-tuning). No test state appears, exactly or after normalisation, in their training data.

model	macro balanced accuracy	micro accuracy
Pretrain-V3	0.280	0.247
Decision-FT Stage 1	0.447	0.439
Decision-Mix Stage 2 from Pretrain-V3	0.543	0.446
Ines-1 (release), zero-shot	0.558	0.457
untuned 322M multilingual encoder	0.553	0.551
untuned 144M encoder	0.485	0.534
Stage 1 + domain fine-tuning	0.784	0.752
Ines-1 + domain fine-tuning (Section 10)	0.790	0.788
<i>test-set majority oracle (diagnostic only)</i>	<i>0.247</i>	<i>0.419</i>

Table 16: Internal domain benchmark, nine subtasks, 1,026 questions. Micro accuracy is dominated by the imbalanced subtasks; the oracle uses the test labels and is not a deployable baseline.

Specializing the released checkpoint. The domain-adapted models above start from internal checkpoints. To test the artefact users would download, we fine-tuned Ines-1 itself — its released bf16 weights, upcast exactly to fp32 — with the recipe that produced “Stage 1 + domain fine-tuning”: same data, validation split, template, loss, four scheduled epochs, learning rate, schedule, accumulation, optimizer and seed; only the parent changed, and the recipe was frozen before the run. Validation chose epoch 3; the test was read once. Ines-1 + domain fine-tuning reaches macro balanced accuracy 0.790, micro accuracy 0.788 and macro-F1 0.780 (Stage 1 + domain: 0.784, 0.752, 0.766; on the same questions, 72 answers are right only for the new model and 35 only for the earlier one, $p = 4 \times 10^{-4}$, most of them on one subtask). On the imbalanced yes/no subtask AUROC rises from 0.541 zero-shot to 0.889 (Stage 1 + domain: 0.883), although its MCC (0.52) stays below the earlier model’s (0.63). Tickets from organisations never seen in training score 0.860 accuracy (0.678 balanced) against 0.841 (0.665) for seen ones, so the gain is not memorisation of known organisations. Adaptation took 21,423 domain cases (17,954 of them labelled by a teacher), 23,608 questions per epoch, 28.5M prompt tokens processed and 2.25 H200 GPU-hours including validation. It is not better everywhere — three subtasks are slightly lower than with the earlier domain model — and specialization has a measurable cost on general ability (Table 17). **Generic typed-decision**

training does not replace domain adaptation, but the released checkpoint remains an effective initialization for specialization.

public test	Ines-1	Ines-1 + domain fine-tuning	Δ (points)
typed-decisions, English	76.4	69.8	-6.6
typed-decisions, Spanish (ours)	75.4	68.9	-6.4
typed-decisions, Spanish (independent)	76.3	67.5	-8.8
tasksource held-out examples	49.5	48.0	-1.6

Table 17: Specialization forgetting, single-question protocol, same questions before and after domain fine-tuning (typed-decisions: paired $p < 10^{-11}$; tasksource: $p = 0.1$). Brier on typed-decisions roughly doubles ($0.065 \rightarrow 0.129$ in English).

11 Serving

The server implements the Jev protocol over HTTP (aiohttp). One thread owns the GPU; whatever arrives while it works goes into the next forward, so a lone request is not delayed. Each turn it groups the waiting prompts by length (the oldest plus its length neighbours while padding stays $\leq 30\%$) and runs one batched prefill in which every row’s decoder covers only that row’s last R positions; small batches replay a CUDA graph captured per (batch, length) bucket, larger ones run eagerly (a forward is launch-bound below a few thousand tokens and GPU-bound above). Only the letter logits are read back.

The GIL lesson. Building prompts (templating and tokenisation) in request threads took the interpreter lock from the thread launching $\sim 2,000$ kernels per forward: the GPU appeared 99% busy while delivering a third of its throughput (76k vs 230k prompt tokens/s in isolation). Moving prompt building to worker processes doubled throughput on realistic requests.

Workload (one process, one H200)	alone	peak questions/s	p50 at peak
2 questions, ~ 60 tokens	9 ms	1,800	66 ms
typed-decisions case, 5 questions, ~ 300 tokens	34 ms	535	290 ms
long cases, ~ 850 tokens	14 ms	200	140 ms

Table 18: Closed-loop load test through HTTP, measured with the Stage 1 checkpoint (same architecture and cost as the release). Through the release server (batched prefill, CUDA graphs, bf16 autocast) the release scores 76.2% (EN) and 75.3% (ES) against 76.4 / 75.4 in the canonical single-prompt path: the batched path is not bit-identical (8 of 400 answers differ on a probe), the accuracy difference is within bf16 noise.

Against encoders, same harness. Table 19 runs the 144M and 322M encoders (Julia-1 and laya-multilingual architectures, in our production server: micro-batching and four worker processes, each with a copy of the model) and Ines-1 (one process) one after another on the same H200, with the same requests and the same load generator. On short requests the 1.6B model is as fast as both encoders — slightly lower latency alone and equal throughput — because at that size every forward is bound by kernel launches, not by arithmetic, and CUDA graphs remove most of them. On typed-decisions cases (~ 300 tokens per question) the work becomes arithmetic and the encoders

are $1.4\text{--}2.1\times$ faster (over two runs): Ines-1 runs at $\sim 180\text{k}$ prompt tokens/s, close to its measured ceiling, with 405M active parameters against 144M and 322M.

Model (processes)	2 questions, ~ 60 tokens		typed-decisions case, 5 questions	
	p50 alone	peak questions/s	p50 alone	peak questions/s
Julia-1 architecture, 144M (4)	13 ms	1,780	17 ms	1,019
laya-multilingual architecture, 322M (4)	18 ms	1,666	17 ms	835
Ines-1, 1.6B / 405M active (1)	10 ms	1,883	30 ms	539

Table 19: Same GPU (one H200), same requests, closed loop with 1–256 clients from 16 generator processes (first of two runs; the second is within $\pm 10\%$ on throughput; full protocol in the release’s `eval/speed.md`). Not a general speed ranking: the serving stacks differ. The encoders’ p90 alone was 25–85 ms against 10–37 ms for Ines-1: their four processes compete for the GPU. The encoder checkpoints used were domain fine-tunes; their speed is that of the architecture.

For generation, the same model reaches 664 tokens/s for a single stream and 38k tokens/s over 128 rows with CUDA graphs and `torch.compile` [2], documented separately.

12 Context length

The model is trained to 2,048 positions with plain RoPE. Rebuilding the rotary tables at load time, without training, we compared position interpolation [8], NTK scaling and YaRN [44] at $4\times$ (Table 20). YaRN holds the loss flat to 8k; NTK breaks past 6k; interpolation without training breaks everything. On the only domain task with long inputs (long call transcripts), however, trimming the middle, NTK, YaRN and chunk-and-combine all scored the same (19/23): the cases it misses are missed with the whole text too.

Mean next-token loss at positions	0–2k	2–4k	4–6k	6–8k
plain RoPE, extrapolated	2.40	4.56	6.40	6.51
position interpolation $\times 4$	5.21	5.20	5.25	5.29
NTK $\times 4$	2.47	1.72	1.89	3.60
YaRN $\times 4$	2.50	1.74	1.81	1.72

Table 20: 24 public documents of 8,192 tokens. They repeat template text, so part of the drop past 2k is copying from context: the table shows what breaks, not long-range understanding.

13 Limitations and negative results

- **Interface.** One question per prompt. With several questions in the prompt accuracy falls to 37–42%, below the dataset’s prior (Section 9.1); API batching does not change that the model sees one question at a time.
- **Evaluation scope.** The typed-decisions results are in-distribution for a specialist trained on its train split; the tasksource test measures new examples of trained tasks; there is no unseen-task evaluation.
- **Uneven zero-shot domain transfer.** On the private benchmark the release is at the level of an untuned 322M encoder (macro balanced accuracy 0.558 vs 0.553) and far below its own

domain fine-tuning (0.790); some subtasks work zero-shot, others do not, and yes/no questions over long texts show a class-prior shift after Stage 2.

- **Specialization forgetting.** Fine-tuning Ines-1 on the domain costs 6–9 points of typed-decisions accuracy and doubles its Brier score (Table 17); a domain model is a specialist, not a better general model [29].
- **Adaptation narrows the gap between starting points.** Under the fixed domain recipe, four substantially different starting checkpoints (Pretrain-V3; Stage 1 after 3 epochs; Stage 1; the release) end in a narrow range of macro balanced accuracy (0.766–0.790), each from a single run; we do not know whether a different adaptation recipe would move that range.
- **No retrieval.** The decision model fails a passkey test [36] at 1k tokens, inside its window. It decides; it does not look things up.
- **The trained model relies on Engram.** Removing it at inference costs 1.7–2.9 points on typed-decisions and up to 17 on a domain task whose classes are short names. No model was trained without it, so this measures reliance, not Engram’s contribution to training. Its pretraining value was smaller (0.14 nats at 1B tokens) and saturated early.
- **We found no domain specialisation in routing** (routing entropy 1.789 of a maximum 1.792; this measures how tokens are routed, not what each expert learns internally), and the learning rate mattered more than adding experts in a 2×2 factorial.
- **Pretraining scale did not create tool use:** identical behaviour at 1B and 10B tokens.
- **Modest hardware efficiency:** about 20% MFU during pretraining (after correcting two accounting errors in opposite directions); generation needed CUDA graphs and compilation to go from 11 to 664 tokens/s.
- **Speculative decoding** [30] with an n -gram drafter was exact but slower.
- The Spanish evaluation is machine-translated; the second, independent translation is there to measure that effect (they agree within 1 point).
- *our-ECE* is not comparable with published ECE values; Brier and KL are.
- **Unmeasured contributions.** Three ingredients of the recipe were never ablated: the larger pretraining corpus as opposed to the longer schedule (Pretrain-V2 vs V3 differ in both), soft teacher distributions as opposed to hard labels, and the validated synthetic examples. We report what the full recipe produced, not the share of each.

14 Future work

Hypotheses, not results: branched or tree attention, or a shared-state prefill, so that several questions about one case are answered independently in a single forward; training explicitly on multi-question cases; and measuring whether either restores full-case behaviour without sacrificing single-question accuracy.

15 Conclusion

Ines-1 is a 1.6B mixture-of-experts model trained from scratch whose main pretraining run processed 9.96B tokens, from a corpus assembled predominantly from public sources, on a single H200. After

decision fine-tuning it reaches 76.4% on a single-question specialist protocol and serves up to about 1.9k questions/s on short requests in our serving setup. That competence does not generalise to multi-question prompts and transfers unevenly to an unseen private domain. However, fine-tuning the exact released checkpoint with a fixed domain recipe raises macro balanced accuracy from 0.558 to 0.790, showing that the public checkpoint remains an effective starting point for specialisation, at a measurable cost to general typed-decision ability. For last-logit scoring, the causal encoder–decoder architecture bounds decoder-side prefill work to a fixed trailing receptive field. We report the adaptation recipe as a whole; the individual effects of soft teacher distributions, synthetic examples and validation design were not isolated.

Reproducibility. Code, configurations, logs and the exact data revisions are in the repository (code: MIT; the weights’ licence is pending). All canonical figures were measured in one fixed environment (one H200, PyTorch 2.14.0+cu130, bf16 weights under bf16 autocast; the release’s `ENVIRONMENT.md`): with the same code PyTorch 2.13 and 2.14 give bit-identical outputs, while autocast on vs off and the attention backend each change 8 of 400 answers. The domain data is private and is not released; only aggregate figures are reported.

References

- [1] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4895–4901, 2023. doi: 10.18653/v1/2023.emnlp-main.298. URL <https://aclanthology.org/2023.emnlp-main.298>.
- [2] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, C. K. Luk, Bert Maher, Yunjie Pan, Christian Puhersch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Shunting Zhang, Michael Suo, Phil Tillet, Xu Zhao, Eikan Wang, Keren Zhou, Richard Zou, Xiaodong Wang, Ajit Mathews, William Wen, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster machine learning through dynamic Python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS ’24)*, pages 929–947. ACM, 2024. doi: 10.1145/3620665.3640366.
- [3] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020. URL <https://arxiv.org/abs/2004.05150>.
- [4] Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM2: When smol goes big – data-centric training of a small language model. In *Conference on Language Modeling (COLM)*, 2025. URL <https://arxiv.org/abs/2502.02737>.

- [5] Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, Anthony DiPofi, Julen Etxaniz, Benjamin Fattori, Jessica Zosa Forde, Charles Foster, Jeffrey Hsu, Mimansa Jaiswal, Wilson Y. Lee, Haonan Li, Charles Lovering, Niklas Muennighoff, Ellie Pavlick, Jason Phang, Aviya Skowron, Samson Tan, Xiangru Tang, Kevin A. Wang, Genta Indra Winata, François Yvon, and Andy Zou. Lessons from the trenches on reproducible evaluation of language models. *arXiv preprint arXiv:2405.14782*, 2024. URL <https://arxiv.org/abs/2405.14782>.
- [6] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7432–7439, 2020. doi: 10.1609/aaai.v34i05.6239.
- [7] Glenn W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1–3, 1950. doi: 10.1175/1520-0493(1950)078<0001:VOFEIT>2.0.CO;2.
- [8] Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023. URL <https://arxiv.org/abs/2306.15595>.
- [9] Xin Cheng, Rui Tian, Wangding Zeng, Damai Dai, Qinyu Chen, Bingxuan Wang, Zhenda Xie, Kezhao Huang, Xingkai Yu, Chengqi Deng, Shangyan Zhou, Chenggang Zhao, Zhewen Hao, Yukun Li, Han Zhang, Zhengyan Zhang, Yixu Wei, M. Y. Xu, Huishuai Zhang, Dongyan Zhao, and Wenfeng Liang. Conditional memory via scalable lookup: A new axis of sparsity for large language models. *arXiv preprint arXiv:2601.07372*, 2026. URL <https://arxiv.org/abs/2601.07372>.
- [10] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. URL <https://arxiv.org/abs/1803.05457>.
- [11] Convai Innovations. laya-multilingual. <https://huggingface.co/convaiinnovations/laya-multilingual>, 2026. Hugging Face model card; mmBERT-base backbone with a decision head; Apache-2.0. Accessed 2026-10-02.
- [12] Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024. URL <https://arxiv.org/abs/2401.06066>.
- [13] DecisionEval. Jev by TypeSafe. <https://decisioneval.dev/models/typesafe-jev/>, 2026. Benchmark page; evaluated on LocalLLaMA/typed-decisions on 2026-09-20. Accessed 2026-10-02.
- [14] DeepSeek-AI. DeepSeek-V4.1-Flash: Pushing the limits of KV cache compression, 2026. URL <https://arxiv.org/abs/2609.19969>.
- [15] DeepSeek-AI. DeepSeek-V4.1-Flash: model card and configuration (config.json). <https://huggingface.co/deepseek-ai/DeepSeek-V4.1-Flash>, 2026. Accessed 2026-10-02.
- [16] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, et al. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024. URL <https://arxiv.org/abs/2412.19437>.

- [17] Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979. doi: 10.1214/aos/1176344552.
- [18] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. URL <https://arxiv.org/abs/2101.03961>.
- [19] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, release v0.4.13. <https://github.com/EleutherAI/lm-evaluation-harness/releases/tag/v0.4.13>, 2026. Release v0.4.13, commit ddd6722. Accessed 2026-10-02.
- [20] Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. In *International Conference on Machine Learning (ICML)*, 2024. URL <https://arxiv.org/abs/2404.19737>.
- [21] Google DeepMind. DiffusionGemma 26B A4B (diffusiongemma-26B-A4B-it). <https://huggingface.co/google/diffusiongemma-26B-A4B-it>, 2026. Model card, Apache-2.0; overview: <https://ai.google.dev/gemma/docs/diffusiongemma>. Accessed 2026-10-02.
- [22] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017. URL <https://arxiv.org/abs/1706.04599>.
- [23] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don’t stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 8342–8360, 2020. doi: 10.18653/v1/2020.acl-main.740. URL <https://aclanthology.org/2020.acl-main.740>.
- [24] helmo. Decida synthetic typed decisions. <https://huggingface.co/datasets/helmo/synthetic-typed-decisions>, 2026. Hugging Face dataset, MIT; generated with DeepSeek-V4-Flash-0731, labels not checked by hand. Accessed 2026-10-02.
- [25] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. URL <https://arxiv.org/abs/1503.02531>. NIPS 2014 Deep Learning Workshop.
- [26] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification, 2016. URL <https://arxiv.org/abs/1607.01759>.
- [27] Nikhil Kandpal, Brian Lester, Colin Raffel, Sebastian Majstorovic, Stella Biderman, Baber Abbasi, Luca Soldaini, Enrico Shippole, A. Feder Cooper, Aviya Skowron, John Kirchenbauer, Shayne Longpre, Lintang Sutawika, Alon Albalak, Zhenlin Xu, Guilherme Penedo, Loubna Ben Allal, Elie Bakouch, John David Pressman, Honglu Fan, Dashiell Stander, Guangyu Song, Aaron Gokaslan, Tom Goldstein, Brian R. Bartoldson, Bhavya Kailkhura, and Tyler Murray. The Common Pile v0.1: An 8TB dataset of public domain and openly licensed text. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025. URL <https://arxiv.org/abs/2506.05209>.

- [28] Amir Hossein Kargaran, Ayyoob Imani, François Yvon, and Hinrich Schütze. GlotLID: Language identification for low-resource languages. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023. URL <https://arxiv.org/abs/2310.16248>.
- [29] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114.
- [30] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning (ICML)*, 2023. URL <https://arxiv.org/abs/2211.17192>.
- [31] Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for LLM training. *arXiv preprint arXiv:2502.16982*, 2025. URL <https://arxiv.org/abs/2502.16982>.
- [32] LocalLLaMA. Typed decisions. <https://huggingface.co/datasets/LocalLLaMA/typed-decisions>, 2026. Hugging Face dataset, Apache-2.0. Accessed 2026-10-02.
- [33] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. URL <https://arxiv.org/abs/1711.05101>.
- [34] Marc Marone, Orion Weller, William Fleshman, Eugene Yang, Dawn Lawrie, and Benjamin Van Durme. mmBERT: A modern multilingual encoder with annealed language learning. In *International Conference on Machine Learning (ICML)*, 2026. URL <https://arxiv.org/abs/2509.06888>.
- [35] Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947. doi: 10.1007/BF02295996.
- [36] Amirkeivan Mohtashami and Martin Jaggi. Landmark attention: Random-access infinite context length for transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL <https://arxiv.org/abs/2305.16300>.
- [37] Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, et al. OLMoE: Open mixture-of-experts language models. In *International Conference on Learning Representations (ICLR)*, 2025. URL <https://arxiv.org/abs/2409.02060>.
- [38] Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using Bayesian binning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, pages 2901–2907, 2015. doi: 10.1609/aaai.v29i1.9602. URL <https://ojs.aaai.org/index.php/AAAI/article/view/9602>.
- [39] Muhammed Nazeem. Typed decisions synth. <https://huggingface.co/datasets/n4ze3m/typed-decisions-synth>, 2026. Hugging Face dataset (user n4ze3m), MIT; LLM-generated (DeepSeek V4.1 Flash), unchecked labels. Accessed 2026-10-02.

- [40] NVIDIA. Nemotron agentic datasets: Nemotron-SFT-Agentic-v2, Nemotron-RL-Agentic-Conversational-Tool-Use-Pivot-v1, Nemotron-RL-Agentic-Function-Calling-Pivot-v1, and Open-SWE-Traces. <https://huggingface.co/datasets/nvidia/Nemotron-SFT-Agentic-v2>, 2026. Also <https://huggingface.co/datasets/nvidia/Nemotron-RL-Agentic-Conversational-Tool-Use-Pivot-v1>, <https://huggingface.co/datasets/nvidia/Nemotron-RL-Agentic-Function-Calling-Pivot-v1>, <https://huggingface.co/datasets/nvidia/Open-SWE-Traces>. Accessed 2026-10-02.
- [41] Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. OpenWebMath: An open dataset of high-quality mathematical web text. In *International Conference on Learning Representations (ICLR)*, 2024. URL <https://arxiv.org/abs/2310.06786>.
- [42] Guilherme Penedo, Hynek Kydliček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024. URL <https://arxiv.org/abs/2406.17557>.
- [43] Guilherme Penedo, Hynek Kydliček, Vinko Sabolčec, Bettina Messmer, Negar Foroutan, Amir Hossein Kargaran, Colin Raffel, Martin Jaggi, Leandro Von Werra, and Thomas Wolf. FineWeb2: One pipeline to scale them all – adapting pre-training data processing to every language. In *Conference on Language Modeling (COLM)*, 2025. URL <https://arxiv.org/abs/2506.20920>.
- [44] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *International Conference on Learning Representations (ICLR)*, 2024. URL <https://arxiv.org/abs/2309.00071>.
- [45] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.
- [46] razorback16. OpenJev: Open, Jev-compatible system one decision server on DiffusionGemma. <https://github.com/razorback16/openjeve>, 2026. GitHub repository, Apache-2.0; independent project, not affiliated with TypeSafe AI. Accessed 2026-10-02.
- [47] samatv256. Jev decisions v1. <https://huggingface.co/datasets/samatv256/jeve-decisions-v1>, 2026. Hugging Face dataset, CC-BY-4.0; derived from NVIDIA Nemotron agentic datasets and nvidia/Open-SWE-Traces. Accessed 2026-10-02.
- [48] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1715–1725, 2016. doi: 10.18653/v1/P16-1162. URL <https://aclanthology.org/P16-1162>.
- [49] Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. URL <https://arxiv.org/abs/2002.05202>.
- [50] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarsz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts

- layer. In *International Conference on Learning Representations (ICLR)*, 2017. URL <https://arxiv.org/abs/1701.06538>.
- [51] Damien Sileo. tasksource: A large collection of NLP tasks with a structured dataset preprocessing framework. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 15655–15684, 2024. URL <https://aclanthology.org/2024.lrec-main.1361>.
 - [52] Luca Soldaini and Kyle Lo. peS2o (Pretraining Efficiently on S2ORC) Dataset. Technical report, Allen Institute for AI, 2023. ODC-By, <https://github.com/allenai/pes2o>. Accessed 2026-10-02.
 - [53] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063. URL <https://arxiv.org/abs/2104.09864>.
 - [54] Supersonic Labs. Julia-1. <https://huggingface.co/SupersonicLabs/Julia-1>, 2026. Hugging Face model card; fine-tuned from jhu-clsp/mmBERT-small; Apache-2.0. Accessed 2026-10-02.
 - [55] tasksource. tasksource-jev-typed-decisions. <https://huggingface.co/datasets/tasksource/tasksource-jev-typed-decisions>, 2026. Hugging Face dataset; license "other" (per-source licenses). Accessed 2026-10-02.
 - [56] telepatia-ai. Typed decisions PT/ES. <https://huggingface.co/datasets/telepatia-ai/typed-decisions-pt-es>, 2026. Hugging Face dataset, Apache-2.0; unofficial PT-BR/ES translation of LocalLLaMA/typed-decisions. Accessed 2026-10-02.
 - [57] TypeSafe AI. Jev: TypeSafe’s first system one model (System One concepts and API reference). <https://docs.typesafe.ai/concepts/system-one>, 2026. API: <https://docs.typesafe.ai/api>; launch post: <https://typesafe.ai/blog/introducing-system-one-models-and-jev>. Accessed 2026-10-02.
 - [58] Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Jiang Chang, Kuai Yu, Liang Zhao, Shangyan Zhou, Zhean Xu, Zhengyan Zhang, Wangding Zeng, Shengding Hu, Yuqing Wang, Jingyang Yuan, Lean Wang, and Wenfeng Liang. mHC: Manifold-constrained hyper-connections. In *International Conference on Machine Learning (ICML)*, 2026. URL <https://arxiv.org/abs/2512.24880>.
 - [59] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4791–4800, 2019. doi: 10.18653/v1/P19-1472. URL <https://aclanthology.org/P19-1472>.
 - [60] Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. In *International Conference on Learning Representations (ICLR)*, 2025. URL <https://arxiv.org/abs/2409.19606>.

A Data licences

typed-decisions (Apache-2.0); telepatia-ai/typed-decisions-pt-es (Apache-2.0, Spanish files only); jev-decisions-v1 (CC-BY-4.0, upstream NVIDIA Nemotron agentic data [40]); tasksource-jev-typed-decisions (per-row licences; only rows whose own `license_use` metadata says `commercial`, which is the dataset’s annotation and not a legal guarantee, mostly MIT, CC-BY-4.0, Apache-2.0, CC-BY-SA-4.0, BSD, with smaller shares of GPL, AGPL, MPL and ODbL sources); typed-decisions-synth and synthetic-typed-decisions (MIT). jev-decisions-v1 is by samatv256 (CC-BY-4.0, attribution required; [47]). Pretraining: FineWeb and FineWeb-Edu (ODC-By), FineWeb-2, Common Pile, OpenWebMath. No training data is redistributed with the model; the licence of the weights is pending review.

B Case study: the deployment decision

For the operational question of whether Ines-1-Domain-v1 (the private adaptation of Section 10) should replace the encoder currently serving the domain — a 322M laya-multilingual model [11] distilled from the same teacher — we measured both with the same harness: the same 1,026 questions sent over HTTP as whole cases, the predicted option taken as the most probable, no tuned thresholds, on the same GPU. Only what was measured is reported here; it says nothing about the two architectures in general.

	correct	micro acc.	macro bal. acc.	macro-F1
incumbent encoder (4 server processes)	841	0.820	0.825	0.819
Ines-1-Domain-v1 (1 server process)	804	0.784	0.788	0.778

Table 21: Private domain benchmark through the production server path. Paired: 82 questions right only for the incumbent, 45 only for Ines-1-Domain-v1 (exact McNemar $p = 0.0013$).

The incumbent is better on overall quality and on the imbalanced yes/no subtask (AUROC 0.927 vs 0.892, MCC 0.76 vs 0.52); Ines-1-Domain-v1 is ahead, without statistical significance, on two subtasks. In serving on the measured traffic pattern, Ines-1-Domain-v1 had lower single-request latency (p95 26 ms vs 92 ms on long single-question requests) and higher throughput on those requests (183 vs 116 questions/s), with slightly more GPU memory (12.4 vs 11.0 GB). The incumbent remains the production choice.