

Bongard: Training Machine Intuition

An Open Encoder–Decoder Model for Probabilistic Judgment

Li Ding * Haidi Jin Chen Ji

AgentBull Pte Ltd

Technical report · September 2026

Abstract

Human intelligence relies heavily on learned intuition: recognising patterns and judging situations without explicitly unfolding every intermediate step. We introduce **Bongard**, an open-weight System One model that treats machine intuition as an independent capability to design and train. A T5Gemma 2 4B-4B encoder–decoder separates reading the evidence from making judgments. The encoder reads the state bidirectionally together with the question instructions, and separate decoder branches share this encoding, so many judgments about the same situation require only one reading of the state. A trained head returns probabilities over the supplied candidates without generating text. Training proceeds in three stages, from supervised judgments to semantic relationships to action outcomes, and each stage updates all 7.09 billion trainable parameters on one Blackwell GPU. Joint-embedding post-training raises accuracy on held-out rephrasings from 75.7% to 85.9%. A sandbox stage then learns outcome distributions from action rollouts and exact oracles, raising accuracy on a frozen sandbox panel from 50.6% to 64.8%. On DecisionBench, the final model reaches 78.05% accuracy over 23,900 decisions and ranks fourth of 61 systems in the public comparison. On one RTX PRO 6000, its median latency is 36 ms for short requests, and 32 questions about one state take 221 ms. Bongard demonstrates that machine intuition can be systematically trained via representation learning and outcome feedback, providing an open, efficient alternative for high-throughput decision workloads.

Weights: huggingface.co/AgentBull/bongard-mini

1 Introduction

Human expertise often manifests as intuitive judgment: rapidly identifying patterns and evaluating situations without explicitly verbalising intermediate reasoning steps (Kahneman, 2011). Expert intuition develops when experience exposes useful regularities and provides feedback (Kahneman and Klein, 2009). Its value extends beyond speed, because an intuitive judgment can draw on a whole pattern whose relevant features are difficult to state one by one. Explicit analysis, by contrast, can change which features a person attends to: in preference studies, asking people to analyse their reasons reduced agreement with expert judgments (Wilson and Schooler, 1991).

In machine learning, related phenomena emerge when models internalise structured tasks. Transformers trained on chess positions can play at grandmaster level without search (Ruoss et al., 2024), and models trained on Othello move sequences can learn internal board representations (Li et al., 2023). In both cases, training places structure in the model, which then uses it directly at inference time.

We study *machine intuition* as the ability to learn relationships within situations and use them to form direct judgments. Bongard exposes this capability through a programmable interface. A request supplies evidence, questions and their possible outcomes, and the model returns a probability distribution for each question without generating an intermediate explanation. The interface covers judgments of meaning and relevance as well as the consequences of actions, so the same model can assess a document, detect conflicting facts or judge an agent’s next step.

TypeSafe AI named models for this workload *System One models*, after Kahneman’s System 1, and released Jev as the first of them (TypeSafe AI, 2026b). A System One model takes a state and typed

*Corresponding author: Li Ding, boris@agentbull.com.

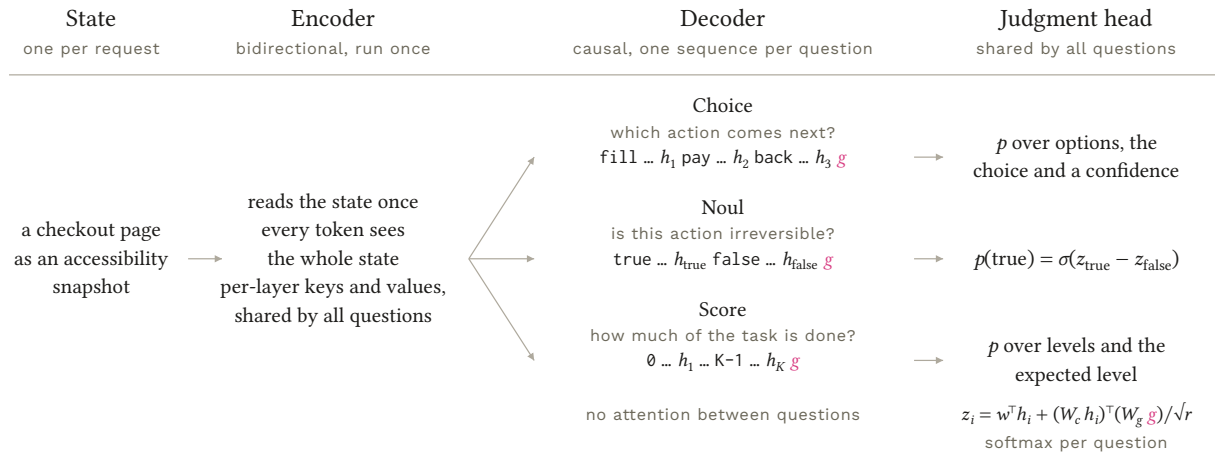


Figure 1: Architecture of Bongard: several typed questions share a single encoding of the state, and no tokens are generated. In this example, three questions of different types concern one checkout page. The encoder reads the state once, and each question is a separate decoder sequence. The decoder processes all sequences in one batched pass, with merged self- and cross-attention over the shared state keys and values. There is no attention between questions. Hidden states at the candidate-end markers (h_i) and the decision-end marker (g) feed a shared bilinear judgment head.

questions and returns a separate distribution over the outcomes of each question. Although the category is recent, it already serves production traffic, for example row by row within SQL queries (MotherDuck, 2026) and as the step executor of browser and mobile agents (Browser Use, 2026; Zhang, 2026). Thousands of public projects use such models for attribute judgment, scoring, action selection, content filtering and model or tool selection (Ling et al., 2026). Replacing large language model (LLM) calls with them can reduce both latency and cost: in one edge service, Jev cut median decision latency by 15.9–26.5% and API fees per correct completion by 69.0–70.6% (Li et al., 2026).

The same interface can expose very different models. Some open implementations wrap pretrained models and read probabilities from answer tokens (OpenJev contributors, 2026; TheoLeeCJ, 2026). Others train a decision model: Kev adds LoRA adapters and a candidate-scoring head to Qwen (Palmer and Kev contributors, 2026), whereas JevK5 and Plumb-4B train adapters but retain letter-logit readouts (allebee, 2026; crh225, 2026). These choices determine what is trained, how the evidence is represented and how candidates are scored, and they imply different costs of adaptation.

Bongard combines a T5 encoder–decoder judgment architecture with training on semantic relationships and action outcomes (Figure 1). The encoder forms a bidirectional representation of the evidence with the question instructions in view. Separate decoder branches judge the supplied candidates from this shared representation, and a dedicated head returns their probabilities directly. This structure suits situations that require several judgments from related evidence, such as a contract, an incident log or a page snapshot. Machine intuition thus becomes a capability that can be trained across tasks and called from within an application.

T5Gemma 2 supplies the encoder–decoder backbone (Zhang et al., 2025c). It inherits Gemma 3 pretraining (Gemma Team, 2025), supports a 128K-token context and more than 140 languages, and includes a vision encoder. Bongard adds a judgment head of 1.3 million parameters that reads its hidden states. The model is trained in three stages, each on one GPU (Figure 2). After supervised training on judgments, a joint-embedding predictive architecture (JEPA) objective aligns the representations of semantically related inputs, and a final sandbox stage learns from the outcomes of actions in executable environments.

Our contributions are the following.

- **An encoder–decoder architecture for machine intuition** (Sections 2 and 3). A bidirectional encoder represents the shared evidence once, and separate decoder branches make probabilistic

Table 1: The three question primitives. The same head answers all of them in the same forward pass.

Primitive	Question	Criteria	Answer
Choice	Which named alternative applies?	1–255 options, each with an optional description	A probability per option, the selected option and a confidence
Noul	Is this statement true?	Optional descriptions of the true and false outcomes	The probability that the statement is true
Score	Where does the input fall on an ordered scale?	2–10 ordered level descriptions	A probability per level, the expected level and a confidence

judgments from it, so N questions cost one encoder pass and about $\lceil N/8 \rceil$ decoder calls. This T5 structure distinguishes Bongard from the decoder-only and diffusion systems in Table 4. Probes with a retention rule fixed in advance favoured the unmodified backbone over every change we tried.

- **Representation alignment and outcome learning** (Sections 4.2 and 4.3). Stage 2 exposes a gap between accurate decisions and the representation behind them, and it shows why cosine alignment saturates without a next-token anchor. A pack-centred contrastive objective makes weak content correspondences recoverable from the decision representation, and the complete stage improves consistency across re-expressions of the same questions. Stage 3 grounds judgments in environment transitions: rollouts and exact oracles label every candidate action with an outcome distribution, and a direct proper-scoring loss trains the model on these distributions.
- **Full-model post-training without a GPU cluster** (Section 5). We post-train all 7.09 billion trainable parameters on one Blackwell GPU, with NVFP4 and FP8 matrix multiplications, FlashAttention-4 over unpadded packs and 8-bit AdamW. The three stages took about 51, 23 and 11 hours, so our training pipeline requires no cluster-scale infrastructure.

The final model is open at huggingface.co/AgentBull/bongard-mini under the Gemma terms of use. The runtime and the server are open at github.com/AgentBull/bongard under the Apache 2.0 licence. The name refers to Bongard problems (Bongard, 1970), visual puzzles in which a hidden rule separates two sets of figures. Solving them requires recognising invariant discriminative rules directly from examples—a core objective of System One models.

2 Task and interface

A request contains a *state*, optional images and a dictionary of typed *questions* (Table 1). The state is a string, a JSON object or an array. The *candidates* of a question are its possible outcomes: Choice options, the two Noul outcomes or Score levels. Questions may ask for a label, such as the queue for a ticket, or for an outcome, such as whether an action is irreversible. A request may mix all three question types.

Requests and responses follow TypeSafe’s System One API (TypeSafe AI, 2026a), so existing clients work unchanged. Images are a Bongard extension. For each question, the response gives the distribution over its candidates and derived fields (Appendix A). These fields are the selected option and a confidence for Choice, the truth probability for Noul, and the expected level and a confidence for Score.

Two properties of this interface shape the design. First, the request fixes the outcome space of each question, so an answer cannot leave its type. Second, questions are isolated: for fixed weights θ ,

$$\text{answer}_i = F_\theta(\text{state}, I, \text{question}_i), \quad (1)$$

where I is the list of distinct question instructions that the encoder reads with the state (Section 3.3). Other questions can reach an answer only through I , never through their names, candidates or criteria. For the stage-1 and stage-2 checkpoints, I is empty. Applications can therefore send every question they have about a state in one request.

3 Architecture

Bongard reads a state once and answers many questions about it (Figure 1). It has three parts: a bidirectional encoder, a causal decoder with one sequence per question, and a compact judgment head.

3.1 Architecture follows the decision workload

Bongard uses T5Gemma 2 as a judgment model. The encoder represents the situation once, and for each question a decoder branch reads this representation before the judgment head scores the candidates directly. This division separates learning to read the evidence from learning to apply it in a judgment.

Adaptation from decoder-only pretraining. T5Gemma 2 is adapted from the decoder-only Gemma 3 with a UL2-style objective (Tay et al., 2023; Zhang et al., 2025c). Both halves therefore inherit the pretraining, the 262K-entry vocabulary and the vision tower of Gemma 3. With the same knowledge, it matches or exceeds Gemma 3 after pretraining, clearly outperforms it after post-training and is stronger on long contexts (Zhang et al., 2025c). The first T5Gemma models showed a better quality–efficiency trade-off than their decoder-only originals (Zhang et al., 2025b). RedLLM finds that instruction-tuned encoder–decoders match or exceed decoder-only models up to 8B parameters, with substantially more efficient inference (Zhang et al., 2025a).

Bidirectional evidence encoding. An encoder represents every state token in the context of the whole state, whereas a decoder-only model reads its prefix forward, so early tokens never attend to what follows them. Whole-state context suits tables, contracts, logs and page snapshots, whose cells and lines depend on headers and neighbours. In the controlled comparison of Raffel et al. (2020), an encoder–decoder with a denoising objective transferred better than decoder-only and prefix language models of similar cost. Su (2023) argues for the opposite design, on the grounds that bidirectional attention matrices tend towards low rank (see also Dong et al., 2021). In our probe, however, forcing the frozen T5Gemma 2 1B-1B encoder to read forward only cut accuracy from 69.8% to 64.6% and raised NLL more than fourfold (Table 3a).

Single-pass state encoding with shared cross-attention. The encoder reads the state, typically hundreds to tens of thousands of tokens, only once. Each decoder layer then projects the encoded state into keys and values once and shares them with every question, as in Fusion-in-Decoder (Izacard and Grave, 2021). A decoder-only model can recover this reuse only at serving time, by prefilling the state as a shared prefix and batching the per-question suffixes (Juravsky et al., 2024; TheoLeeCJ, 2026; kikoncuo, 2026; ekzhang, 2026), and its shared representation remains causal. The split into two stacks costs memory rather than compute: each token passes through only one 34-layer stack shaped like Gemma 3 4B.

Asymmetric capacity allocation. In a decoder-only model, every question token passes through the full stack. In an encoder–decoder, the state passes once through the encoder, and question tokens pass only through the decoder. Because adaptation allows the two stacks to differ in size (Zhang et al., 2025b), capacity can shift to the encoder, which runs once per state, while the decoder, which runs for every question, remains small. A decoder-only model offers no such trade-off, since every token incurs the cost of the full stack. The current model uses the balanced 4B-4B configuration and leaves unbalanced configurations to future work.

Trained judgment head instead of vocabulary readout. Next-token logits over letters, yes/no tokens or masked slots give a distribution over vocabulary items. This distribution is entangled with the tokeniser, label-word priors and option position (Zhao et al., 2021b; Zheng et al., 2024), which post-hoc temperature scaling addresses only in part (Guo et al., 2017). Readouts over fixed option codes avoid the vocabulary but still score code slots rather than the candidates themselves (Garg, 2026a). Bongard instead scores candidates given in full, by name and description, with a head trained under proper scoring rules. Encoder–decoder rerankers established the pattern of scoring from decoder outputs (Nogueira et al., 2020; Zhuang et al., 2023), and Bongard extends it to typed judgments with many questions per request.

Table 2: Bongard 4B-4B configuration. Training updates all components except the vision tower, so 7.09B of the 7.51B parameters are trainable.

Component	Configuration	Parameters
Text encoder	34 layers, $d = 2560$, FFN 10,240, 8 query / 4 KV heads of dimension 256, window 1,024 (5 local : 1 global), bidirectional	3.21B
Text decoder	Same shape, causal, merged self- and cross-attention	3.21B
Tied embeddings	262,144 tokens $\times$ 2,560	0.67B
Vision tower	SigLIP (Zhai et al., 2023), 896 px, 256 tokens per image, frozen	0.42B
Projector, head	Trained, with $w \in \mathbb{R}^{2560}$ and $W_c, W_g \in \mathbb{R}^{256 \times 2560}$	4.26M
Total	Bongard limit: 32,768 tokens per state plus question	7.51B

3.2 Backbone and judgment head

Bongard keeps the architecture of google/t5gemma-2-4b-4b unchanged (Table 2) and adds new parameters only for the head and the multimodal projector. The state is serialised as compact JSON and encoded together with any image tokens. Each question is compiled into a separate decoder sequence that spells out its type, instructions and candidates (Appendix A). Two unused entries of the existing vocabulary serve as markers: a candidate-end marker follows each candidate, and a decision-end marker ends the question. The vocabulary is therefore never resized. A guarded tokeniser encodes any literal marker text in user input as byte tokens, so user input cannot forge a marker.

Let h_i be the last-layer decoder hidden state at the i -th candidate-end marker, and let g be the hidden state at the decision-end marker. We call h_i a *candidate readout* and g the *decision readout*. The head scores candidate i as

$$z_i = w^\top h_i + \frac{(W_c h_i)^\top (W_g g)}{\sqrt{r}}, \quad r = 256, \quad (2)$$

and a softmax over the scores $z_1, \dots, z_K$ of the K candidates gives the distribution of the question.

For fixed g , the head is a linear scorer whose weights depend on the whole question. This dependence lets a causal decoder score early candidates with information from later candidates. The interaction must be multiplicative, because an additive term in g would cancel in the softmax. Noul returns $\sigma(z_{\text{true}} - z_{\text{false}})$ and Score returns the level distribution and its expectation, so no primitive requires token generation.

3.3 Parallel questions and question-aware encoding

T5Gemma 2 normalises decoder self-attention and cross-attention in one softmax. Bongard computes this softmax in two partitions: the causal prefix of each question and the shared state. Following Hydragen (Juravsky et al., 2024), it merges the two partitions with their log-sum-exp weights, $\log Z_q$ for the question prefix and $\log Z_s$ for the state. The state partition batches the queries of all questions against a single copy of the state keys and values. For a given encoder input, a batched request therefore computes exactly the same function as separate single-question requests. The decoder processes questions in length-sorted groups of up to eight, so N questions cost one encoder pass and about $\lceil N/8 \rceil$ decoder calls.

Question-aware encoding. From stage 3 on, the encoder reads the state followed by the distinct text instructions of the request’s questions. The encoder can therefore focus its reading on what the questions ask, while candidates and criteria stay in the decoder. A request that would exceed the token budget with the instructions is encoded without them. This rule is deterministic, so training and serving agree.

Tests confirm isolation directly. Without question-aware encoding, answers stay the same when other questions in the request are reordered, renamed, deleted or injected. With it, the same holds for every change that keeps I , because the encoder input is then unchanged. Gradients from one question also never reach the decoder sequence of another question.

Table 3: Architecture probes on T5Gemma 2 1B-1B. (a) Encoder masks with a frozen backbone and a trained head, on synthetic judgments of thresholds, negation and role binding, averaged over two seeds. Fact direction is the share of fact changes that move the truth probability in the correct direction. (b) Decoder modifications after short full-parameter training, as changes relative to the unmodified model for seeds 23 and 37. The value-residual row comes from a longer run and reports cross-task NLL as transfer.

<i>(a) Encoder attention mask</i>					
	Accuracy $\uparrow$	NLL $\downarrow$	Brier $\downarrow$	Fact direction $\uparrow$	Effective rank
Bidirectional (native)	69.8%	0.677	0.219	100.0%	49.7
Forward only	64.6%	2.944	0.320	70.8%	37.9
Forward and backward heads	57.3%	1.596	0.363	62.5%	31.7
<i>(b) Decoder modification</i>					
	Held-out NLL $\Delta \downarrow$		Transfer NLL $\Delta \downarrow$		Latency Δ
Bidirectional global layers	−25.4% / −1.0%		−3.3% / +1.7%		$\approx 0\%$
Tail readout, zero-initialised residual	−4.3% / +0.2%		+10.7% / +12.1%		+13 to +25%
Zero-initialised fusion bias	+26.4% / +27.5%		−6.9% / +45.2%		$\approx +1\%$
Value residual	+3.2% / +9.3%		−0.03% / −1.2%		−

An optional cache of encoder outputs serves repeated requests about a recent state. A cache hit needs the same encoder input, so under question-aware encoding it also needs the same I . On a 1B-1B model, this cache cut the latency of a repeated request with a 1,924-token state tenfold, from 1,165 to 115 ms, with bit-identical logits.

3.4 Architecture probes

We probed the design on the 1B-1B model of the same family (Table 3). The native bidirectional encoder outperformed the forward-only mask and the mask with separate forward and backward heads on every measure. Its final hidden states also kept the highest effective rank. We therefore found no sign of the rank collapse that Su (2023) predicts.

We retained none of the four decoder modifications. Bidirectional global layers within a question improved held-out negative log-likelihood (NLL) on both seeds, but by widely varying amounts. They also worsened transfer on one seed and so failed the retention rule that we fixed before training. A tail readout added 13–25% to latency and transferred worse. The native fusion of self- and cross-attention is already the gate $\sigma(\log Z_s - \log Z_q)$, and a learned bias on this gate hurt held-out NLL. The value residual of ResFormer (Zhou et al., 2024), which RWKV-7 also uses (Peng et al., 2025), raised validation NLL on both seeds.

3.5 Jev, open implementations and design tradeoffs

Table 4 locates Bongard within the System One ecosystem. An inference wrapper changes how an existing model is called and how its answer is read, whereas training changes the judgment function itself. Bongard trains the text backbone, the multimodal projector and the judgment head on judgments, semantic correspondences and action outcomes, while the vision tower stays frozen.

Kev makes the architectural contrast concrete (Palmer and Kev contributors, 2026). It trains rank-16 LoRA adapters and a pointer head on Qwen, and its server reuses a causal state cache across question branches. Bongard instead forms a bidirectional state representation in a T5 encoder, which its decoder branches share. Full-model updates with additional representation and outcome objectives require more training and data construction than adapter training, but they allow Bongard to shape the evidence representation and the judgment function together.

In practice, bidirectional encoding allows earlier facts to be represented in the context of later ones, which matters for relationships across tables, documents and page snapshots. Separate encoder and

Table 4: Architecture and learning choices in documented System One systems. Sources: Jev ([TypeSafe AI, 2026b](#); [Hume, 2026](#)), OpenJev ([OpenJev contributors, 2026](#)), Kev ([Palmer and Kev contributors, 2026](#)), JevK5 and Plumb-4B ([allebee, 2026](#); [crh225, 2026](#)), AutoJev ([denis-pplx, 2026](#)), MoJev ([MoLeMo Lab, 2026](#)), CLM ([Contrastive-LM, 2026](#)), imajev ([Garg, 2026a](#)), and Laya and Verdict ([Convai Innovations, 2026a](#); [OpenJev contributors, 2026](#)). Jev’s architectural details are inferred from API behaviour.

System	Backbone and training	How the state is read	How the answer is read
<i>Encoder–decoder</i>			
Bongard	T5Gemma 2 4B-4B; full text-model training on judgments, semantic pairs and action outcomes	Bidirectional, question-aware encoder, once per request, shared by separate decoder branches	Trained bilinear head on candidate and decision readouts, no sampling
<i>Closed</i>			
Jev	Undisclosed backbone; RLCD described by TypeSafe	Isolated question branches (observed), state processed once (suggested by latency)	Direct probabilities, options interact (observed)
<i>Decoder-only or diffusion language models</i>			
OpenJev default	Pretrained DiffusionGemma 26B-A4B; inference wrapper	State and all questions in one canvas	Label-token probabilities at masked slots, noisy re-reads when uncertain
Kev	Qwen3.5/3.8; rank-16 LoRA and pointer head, supervised decision loss	Causal state cache reused across separate question rows in the server	Trained pointer head scores each option against the final decision state
JevK5, Plumb-4B	Qwen3.5-4B decoder with LoRA adapters	Causal prefix, one prompt per question	Letter logits over options with a fitted temperature
AutoJev-27B	Qwen3.8-27B decoder, full fine-tuning	Causal prefix, one pass per question	Probabilities over supplied choices, scalar temperature
MoJev	Qwen3.5-0.8B decoder	Tree-packed attention over state and questions	Shared rank-512 judgment head
CLM	Frozen Qwen3-8B with small projection heads	State embedded once, as one vector	Softmax over scaled cosine of state and action projections
imajev	Qwen3.5 2B, 4B or 9B decoder with LoRA adapters (rank 16)	Causal prefill once per request (up to 4,096 tokens), one pass per question	Readout over 255 option codes plus an unknown option, four option rotations when served
<i>Text encoders</i>			
Laya, Verdict	ModernBERT encoders	Bidirectional, short inputs, one pass per question	Classification or GLiClass head

decoder stacks also allow future designs to allocate more capacity to the single reading of a state than to each repeated question. Semantic-correspondence training targets invariance to changes in wording and modality (Section 4.2), whereas outcome training ties judgments to the effects of actions (Section 4.3).

Jev is the closed reference system. TypeSafe describes a specialised architecture and a training method, Reinforcement Learning for Calibrated Decisions (RLCD) ([TypeSafe AI, 2026b](#)), and API experiments suggest shared state processing, isolated question branches and interaction between candidates ([Hume, 2026](#)). Its internal representation and full training recipe, however, are not public. Bongard offers an open encoder–decoder route to the same class of callable judgments, and the task-level comparisons in Table 8 and Figure 4 show where the strengths of the two systems differ.

4 Training

The three training stages (Figure 2) progressively establish task competence (stage 1: supervised training), semantic alignment (stage 2: JEPA) and outcome-driven calibration (stage 3: sandbox RL). All stages use

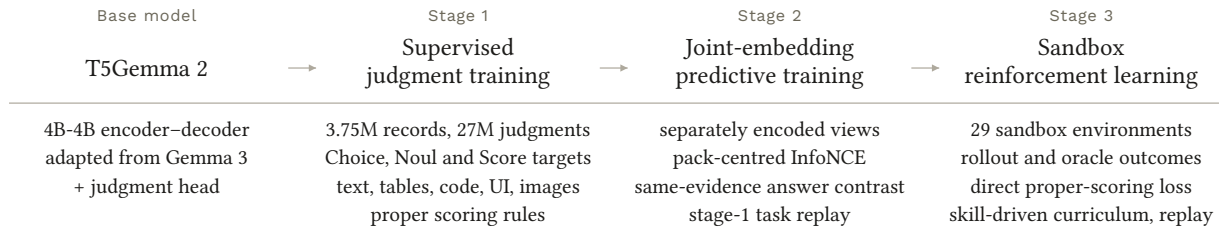


Figure 2: Training pipeline. A pretrained T5Gemma 2 encoder-decoder receives a judgment head and is then trained in three stages: supervised judgment training, joint-embedding predictive training and sandbox reinforcement learning. Each stage updates all trainable parameters on one GPU.

the same request format and judgment head, and each trains on one GPU (Section 5). From stage 3 on, the encoder also reads the question instructions (Section 3.3).

4.1 Stage 1 (supervised training): learning to judge

Stage 1 draws supervised judgments from text, structured data and images (task families in Appendix B). Each record is a state with one or more questions and a target for each question. Noul targets are hard or soft Bernoulli targets, and Choice and Score targets are hard classes, full distributions or *allowed sets* of tied correct candidates. All losses apply proper scoring rules to the output distributions of the head (Gneiting and Raftery, 2007). Four construction methods matter most.

Program-exact synthesis. Generators create constrained worlds and executable problems, and exact solvers or executions label the questions. Where independent checks are available, a label is kept only if they agree. The generators also create paired examples that either change the answer through a relevant fact or preserve it under an irrelevant change.

Derived judgments. A labelled example can support related judgments, including per-option truth, numeric thresholds, joint events and next-step prediction. Each question sees only the evidence available before its target outcome.

Retrieval judgments. Retrieval examples become listwise, pairwise and single-passage questions. A teacher reranker filters ambiguous examples, while the supervised targets remain hard. Candidate order varies, and some questions have no relevant passage. The retrieval intent stays in the state because relevance depends on it.

Structured data and teachers. Structured records support lookup and prediction questions. Rule-based checks validate exact answers, and consistency checks filter teacher labels. Where exact labels are unavailable, filtered teacher distributions provide soft targets. Questions with known random mechanisms carry exact probabilities.

Across the corpus, instructions are rewritten in multiple languages and candidate order is varied. Related records stay in the same split because split groups are assigned before rewriting or pairing. Stage 1 makes one pass over the corpus.

4.2 Stage 2 (JEPA): learning the structure behind judgments

4.2.1 Motivation

Supervised training constrains only the final output distribution, leaving the model prone to relying on template fingerprints, option heuristics or surface shortcuts. To ensure robust generalisation, the underlying representation should reflect semantic invariants: equivalent facts phrased differently should map to similar embeddings, while the prediction of an outcome should align with the representation of the observed outcome.

JEPAs learn such invariants: they predict the representation of one view from another view in

Table 5: Geometry of the decision readout g after stage 1, from forward passes over 256 training origins per family. Raw cosine between unrelated readouts is already 0.93–0.996. Centring each side on its mean exposes the true alignment. Logical complements and inherited views are already aligned. Separately encoded content is not, except for paraphrase pairs, whose two sentences share most of their words.

Family	Raw cosine paired / unpaired	Centred cosine paired / unpaired	Centred retrieval top-1 ($N = 256$)
<i>Shared encoded state</i>			
Answer state	0.995 / 0.993	0.46 / -0.01	28.9%
Logical complement	1.000 / 0.996	0.94 / 0.05	99.6%
Inherited view	1.000 / 0.979	0.98 / 0.00	71.9%
<i>Separately encoded</i>			
Paraphrase	1.000 / 0.996	0.86 / 0.02	93.0%
Question $\rightarrow$ answer content	0.936 / 0.929	0.30 / 0.05	0.8%
Image $\rightarrow$ content	0.962 / 0.961	0.20 / 0.00	3.5%
Query $\rightarrow$ passage	0.950 / 0.947	0.23 / -0.01	11.3%
World prediction	0.980 / 0.979	0.07 / 0.00	1.2%
Causal prediction	0.988 / 0.988	0.19 / 0.00	1.6%
Assignment (action values)	0.993 / 0.992	0.11 / 0.00	1.2%
Matching (stable allocation)	0.995 / 0.995	0.00 / 0.00	0.0%

embedding space (LeCun, 2022; Dawid and LeCun, 2023; Assran et al., 2023; Bardes et al., 2024; Assran et al., 2025). LLM-JEPA adds an embedding-space term to the standard loss of a language model (Huang et al., 2025), and BERT-JEPA applies the idea to encoder sentence embeddings (Gillin et al., 2026). In LLM-JEPA, however, the hidden state must also predict the next token over a vocabulary of about 260,000 entries. This next-token loss anchors the representation, so a plain cosine term suffices. A judgment model lacks such an anchor: its head reads g only through a 256-dimensional projection, and pure alignment without negatives has nothing to keep representations apart (Wang and Isola, 2020).

4.2.2 Views

A JEPA pair consists of two *views*, each a state with a question. The views must agree: a declared family of questions has the same answer distribution under both views. Each pair comes from one stage-1 record, its *origin*, which receives one relation (Table 11). Two kinds of relation carry the alignment terms.

Content and world pairs encode their two views separately, from different inputs. For example, a query pairs with a passage’s content, and an image pairs with its annotated content. A world and an action pair with the observed next state, and a program pairs with its re-executed result. Two independent solvers or a re-execution verify the targets of these pairs. *Answer states* pair a multiple-choice question with two event questions under the same evidence. One asks whether the correct option is the answer, and the other asks the same of a wrong option.

The alignment terms always act on the decision readout g of each view. As in LLM-JEPA, the decoder itself serves as the predictor, with tied weights. There is no separate predictor, momentum teacher or negative queue.

4.2.3 Diagnosis: cosine alignment has almost no signal

We probed the stage-1 model with forward passes only, over 256 origins per family, and the results shaped the objective (Table 5).

Raw cosine is saturated. All readouts share one dominant direction, the anisotropy that is familiar from contextual representations (Ethayarajh, 2019). For answer states, $1 - \cos$ is 0.0051 for true pairs and 0.0071 for unrelated pairs. A cosine objective therefore spends nearly all its gradient on making all

readouts more collinear. The head’s subspace captures only 11.9% of the item-specific variance of g , which is about what a random 256-dimensional subspace would capture.

Shared-state views are already aligned, but separately encoded content is not. After centring, logical complements and inherited views, which share an encoded state, reach retrieval of 72–99.6%. Further alignment of these views would reward copying the fingerprint of the state. Separately encoded content reaches only 0–11%, and LLM-JEPA operates in this regime. Paraphrase pairs are the exception: they are encoded separately but reach 93.0%, because their two sentences share most of their words.

The decision readout does not encode the answer. Although the model answered these multiple-choice questions with 99.6% accuracy, g was closer to the wrong option’s “no” event (centred cosine 0.54) than to the correct option’s “yes” event (0.46). The candidate readouts and the bilinear head decide the answer, and a positive-only objective cannot separate events that share their state and template.

4.2.4 Objective

Stage 2 minimises

$$\mathcal{L} = \mathcal{L}_{\text{SFT}} + \mathcal{L}_{\text{view}} + \lambda (\mathcal{L}_{\text{content}} + \mathcal{L}_{\text{answer}}), \quad (3)$$

where $\mathcal{L}_{\text{SFT}}$ is the stage-1 judgment loss and $\mathcal{L}_{\text{view}}$ supervises the alternative views. The two alignment terms act on $g \in \mathbb{R}^{2560}$ itself, after *pack centring*. A *pack* is one physical training batch. For the set S of readouts of one view type (such as source or target) in a pack, we subtract the detached mean and normalise:

$$c(x)_i = \frac{x_i - \text{sg}[\mu_S]}{\|x_i - \text{sg}[\mu_S]\|}, \quad \mu_S = \frac{1}{|S|} \sum_{j \in S} x_j. \quad (4)$$

The shared direction of Table 5 therefore cannot contribute to any loss.

Content pairs (Figure 3a) use a symmetric InfoNCE loss (van den Oord et al., 2018) over the pack, with $s = c(g_{\text{src}})$, $t = c(g_{\text{tgt}})$ and temperature $\tau = 0.1$:

$$\mathcal{L}_{\text{content}} = \frac{1}{|C|} \sum_{i \in C} \frac{1}{2} [\text{CE}(\ell_i, i) + \text{CE}(\ell_i, i)], \quad \ell_{ij} = s_i \cdot t_j / \tau, \quad (5)$$

where C is the set of content pairs in the pack. A split group holds the records derived from one source item, and pairs from the same split group are excluded from each other’s negatives. Both views keep their gradients. The other origins in the pack provide the uniformity that pure alignment lacks (Wang and Isola, 2020; Gao et al., 2021), so the objective rewards correspondence rather than collinearity.

Answer states (Figure 3b) contrast the anchor $u = c(g_{\text{MCQ}})$, the centred decision readout of the multiple-choice question, with two stop-gradient targets under the same evidence. The target p is the readout of the correct option’s “yes” event, and n is that of a random wrong option’s “no” event. The loss over the set $\mathcal{A}$ of answer-state origins in the pack is

$$\mathcal{L}_{\text{answer}} = \frac{1}{|\mathcal{A}|} \sum_{i \in \mathcal{A}} \text{softplus} \left(-\frac{u_i \cdot p_i - u_i \cdot n_i}{\tau} \right). \quad (6)$$

The event questions carry privileged information, the pointer to the answer, so they serve as targets rather than trainable views. Their readouts are centred separately to remove the global offset between yes and no. Because p and n share evidence and template, their common fingerprint cancels in $u_i \cdot (p_i - n_i)$, and only the direction that separates the correct option from a wrong one remains. Wrong options are only ever contrasted and never used as positives. View supervision gives yes and no events equal weight, so the alternative template carries no label prior.

Weighting. We measured the ratio of gradient norms between the alignment terms and the judgment loss on real packs. At $\lambda = 1$ its median was 128.6, so we set $\lambda = 0.004$, which makes the alignment gradient about half the judgment gradient.

Stage 2 assigns one relation to each origin and replays stage-1 tasks to limit drift. Before the full run, we trained an independent short control, which Section 4.2.5 reports with the results.

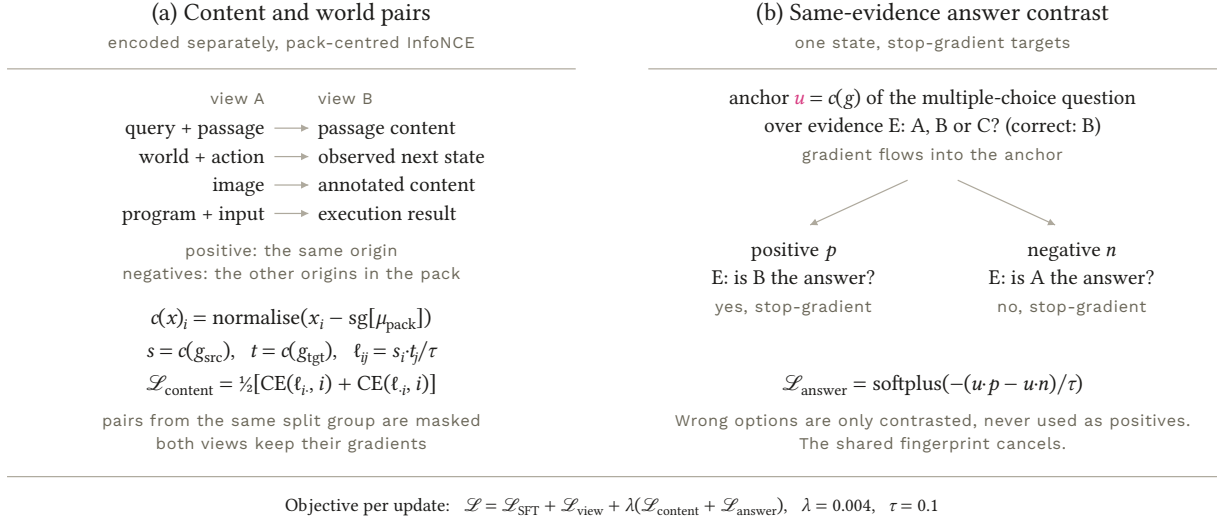


Figure 3: JEPA without a next-token anchor: the stage-2 objective. (a) Separately encoded content and world pairs are centred per pack and trained with a symmetric InfoNCE loss against the other origins in the pack. (b) The anchor is the decision readout of a multiple-choice question. It is contrasted with stop-gradient readouts of a correct “yes” event and a wrong “no” event under the same evidence, so their shared fingerprint cancels.

Table 6: Effect of stage 2. The differences combine the alignment terms, the alternative views and continued training. Retrieval is measured within one source, so matching on source fingerprints alone cannot raise the score.

Measure	Stage 1	Stage 2
<i>Within-source retrieval top-1 on never-paired origins ↑</i>		
Query → passage	15%	98%
Question → answer content	4%	86%
Image → content	7%	67%
World prediction / causal prediction	4% / 1%	35% / 31%
Answer state: correct event nearer than wrong event	23%	77%
<i>Frozen panel: 14,606 held-out re-expressions of 4,096 questions</i>		
Accuracy ↑	75.7%	85.9%
NLL ↓ / Brier ↓	0.613 / 0.342	0.437 / 0.217
Agreement with the original question ↑	78.9%	89.0%
Logical complements answered correctly ↑	44.8%	71.1%
<i>Evidence panels: both variants correct after a fact change ↑</i>		
Unseen rule worlds / image evidence	97.7% / 98.8%	98.3% / 100%

4.2.5 Results

We evaluate stage 2 in three ways (Table 6). A representation probe uses held-out origins that never appeared in a stage-2 pair. A frozen panel holds 4,096 origins and 14,606 held-out re-expressions of their questions, such as restatements, logical complements, conditional events, ordinal thresholds and reordered options. Evidence panels test whether judgments change when facts change.

Representation learning. For question–answer, query–passage and image content, retrieval of never-paired content rose from single digits or low teens to 67–98%. The families for world prediction, causal prediction, and assignment and matching also improved. The raw cosine between unrelated answer-state readouts fell from 0.993 to 0.48, so the anisotropy that motivated the objective is largely removed.

Judgment consistency. On held-out re-expressions, accuracy rose by ten points and NLL fell by 29%. The model improved markedly on logical complements and on both accepting correct options and rejecting wrong ones (Appendix C). Evidence sensitivity improved slightly. External calibration, however,

moved towards overconfidence: expected calibration error rose from 0.12 to 0.15, and temperature fitting targets this shift (Section 6).

Attribution. Stage 2 combines alternative views, continued training and the alignment terms. An independent control isolates the alignment terms: two 400-update runs started from the stage-1 checkpoint, saw identical batches and differed only in λ (0 or 0.004). Alignment raised within-source retrieval of held-out question–answer content from 2.3% to 56.6% and of image content from 1.6% to 39.1%.

4.3 Stage 3 (sandbox RL): learning from consequences

Calibrated decision-making requires feedback grounded in environment transitions rather than static reward signals. Stage 3 therefore trains the model on the outcome distributions of candidate actions, estimated from rollouts or computed by exact oracles. TypeSafe states that RLCD, the training method of Jev, optimises calibrated probabilities (TypeSafe AI, 2026b), and our stage aims at the same effect with its own recipe. In sandbox environments, the model answers outcome questions about candidate actions, takes the action with the highest expected utility and then learns from the observed outcomes.

Environments. The sandbox covers games, grid worlds, verifiable judgments, business simulators, computer use, classification and executable generators (Appendix D). Classification tasks include structured-record prediction and assistant-intervention decisions.

The model judges *roots*, the sandbox states about which it is questioned. Executable tasks supply exact labels for common deployment judgments. Because fast decision models already execute the steps of browser and mobile agents (Browser Use, 2026; Zhang, 2026), computer-use questions ask about task progress, errors, risks and candidate actions over accessibility snapshots (Microsoft, 2026).

Labels and objective. We fork every legal action from a root and label it with an outcome distribution. The distribution is either a rollout frequency under the continuation policy stated in the question or the exact result of an oracle, such as dynamic programming or an endgame solver. We train on it directly with a proper scoring rule, the cross-entropy against the full outcome distribution. The judgment model is the only learned component, with no value network or reward model.

When a policy gradient is calibrated. RL usually trains with a policy gradient, and the reward decides whether that gradient is calibrated. Consider a question with model distribution q and an outcome distribution p that stays fixed during the update. We sample a predicted outcome $A \sim q$, which the environment never executes, and score it with $C = 1[A = Y]$, where $Y \sim p$. With the reward $R = C - q_A$ held constant, the policy-gradient (score-function) estimator (Williams, 1992) satisfies

$$\mathbb{E}[(C - q_A) \nabla_{\theta} \log q_A] = \sum_a (p_a - q_a) \nabla_{\theta} q_a = -\frac{1}{2} \nabla_{\theta} \|q - p\|_2^2. \quad (7)$$

This expectation is half the negative Brier gradient (Brier, 1950), so the fixed point is the true outcome probability rather than certainty. A correctness-only reward, or normalisation by the group standard deviation, instead pushes the top class towards one.

For a fixed observed Y , however, the expectation of the estimator over A is exactly the negative gradient of the single-outcome Brier loss $L(q, Y) = \frac{1}{2} \sum_k (q_k - 1[Y = k])^2$. The direct loss therefore has the expected gradient of Equation (7) without the extra sampling noise. Sampling A helps only when feedback reveals no more than whether a sampled answer was correct. Our environments reveal the outcome itself, so we train with direct proper-scoring losses. Section 7.8 reports calibration before and after temperature fitting.

Warm start. Stage 3 opens with supervised updates that introduce question-aware encoding (Section 3.3). This warm start combines replayed supervised records, fixed sandbox samples and additional decision questions, including routing, multi-question workflows and larger choice sets.

Rounds and curriculum. Sandbox rounds follow the warm start. Each round plays episodes with the current model, labels legal actions at new roots and measures skill on these fresh roots before any

Table 7: Training stack of stages 1 and 2 on a single B200 (PyTorch 2.14, CUDA 13.0).

Component	Implementation
Feed-forward blocks	68 fused Transformer Engine blocks (RMSNorm, gate and up projections, GELU gating, down projection) with NVFP4 matrix multiplications (NVIDIA, 2026, 2025)
Attention projections	272 projections as rowwise-scaled FP8 matrix multiplications (Micikevicius et al., 2022; PyTorch Team, 2026)
Attention	FlashAttention-4 in variable-length mode (Zadouri et al., 2026; Dao et al., 2022), FlexAttention for sliding-window sequences (Dong et al., 2024), BF16 arithmetic
Storage and optimiser	BF16 parameters and gradients with no FP32 master copy, and 8-bit AdamW (Dettmers et al., 2022) with stochastic rounding (Gupta et al., 2015)
Batching and memory	Unpadded packs of 24,576–28,672 tokens (131,072 tokens per update), state keys and values stored once per record, and recomputation (Chen et al., 2016) only for records above the pack limit
Compilation	torch.compile regions (Ansel et al., 2024), and quantised weights reused until the parameters change

update. Updates combine recent roots with replayed supervised records. Environment sampling follows measured skill: tasks at intermediate skill receive more weight, while every environment retains a share.

Selection. A frozen panel of 16,992 questions at 1,953 roots tracks the stage every two rounds. Panel accuracy rose from 50.6% to 64.8%, and the Brier score fell from 0.482 to 0.289. Nearly all of this gain came by round 8. The final model is therefore the uniform average of the five checkpoints from rounds 8 to 15 (Izmailov et al., 2018; Wortsman et al., 2022). On evaluation sets excluded from stage 3 training, the average matched or beat the individual checkpoints on most measures.

5 Full-parameter post-training on one GPU

Every stage trains all 7.09 billion trainable parameters on a single GPU. Stages 1 and 2 ran on an NVIDIA B200 (183 GB) with the stack in Table 7 (details in Appendix E) and took about 51 and 23.4 hours.

Stage 3 ran on a B300 (288 GB). Its warm start used NVFP4 Transformer Engine linear layers for the feed-forward projections, BF16 attention projections and packs of 49,152 tokens, with the rest of the stack unchanged, and took 8.5 hours. The sandbox rounds trained in BF16 and took 2.5 hours in total.

Three choices account for most of the efficiency.

- **NVFP4 feed-forward blocks.** These blocks hold most of the parameters and most of the matrix work. At 8,192 tokens, their down-projection kernels run 2.1 times faster than in FP8 (0.75 against 1.56 ms). The fused block reproduces the function of the original module.
- **Merged attention in one FlashAttention-4 call.** The keys of each question are laid out as its state followed by its own prefix. The bottom-right-aligned causal mask of the kernel therefore yields the joint softmax of self- and cross-attention. No second pass or log-sum-exp merge is necessary.
- **Unpadded packing.** Projections and feed-forward blocks run over the whole pack as single matrix multiplications, and only attention separates the sequences.

A steady-state profile shows that the low-precision matrix multiplications run at about 2.9 PFLOP/s but take only a quarter of GPU kernel time. Fusion and weight reuse target quantisation, transposes, element-wise work and launch overhead.

6 Serving and calibration

The command `bongard serve` implements the System One endpoints of Section 2. We fit one temperature per primitive, and one per option count where enough held-out questions exist (Guo et al., 2017). The fitting pool resembles deployment: external development sets, exact-probability questions and teacher-authored held-out questions. We bind the temperatures to the checkpoint hash. For the final model, the fitted temperatures lie between 1.2 and 2.15.

Because h_i sees only candidate i and the candidates before it, Choice probabilities are not guaranteed to be invariant to option order. Training reshuffles the options for reviewed templates. An optional serving mode, off by default, also averages each Choice question over cyclic rotations of its options within one request.

The model also runs on a single 24 GB consumer GPU: on an RTX 4090, JevBench’s own client measures a median latency of 116 ms per public item.

The same checkpoints run on Apple Silicon, where optional Metal kernels fuse attention as well as query–key normalisation with rotary encoding. Their design draws on MLX (Hannun et al., 2023) and Metal FlashAttention (Turner, 2024).

7 Evaluation

We evaluate the final model, the uniform average of the stage-3 checkpoints from rounds 8 to 15, with its serving temperatures. Unless noted, timings use one NVIDIA RTX PRO 6000 in BF16. Numbers for other systems come from the public leaderboards, reports and dataset cards named in each table, except our own Jev runs in Section 7.2.

The experiments in this section complement the stage-level results of Sections 4.2.5 and 4.3: they measure the judgment quality of the final model across tasks and the cost of repeated judgments over shared evidence.

7.1 System One benchmarks

Table 8 summarises the public benchmarks. With the official DecisionBench runner, Bongard answers 78.05% of all 23,900 rows correctly and ranks fourth of 61 systems, behind only the benchmark authors’ own models and Imajev-4B. It ranks ahead of Jev, Winnow-12B and frontier language models such as GPT-5.6 Luna and DeepSeek V4.1 Flash. Its probabilities are also substantially more reliable than Jev’s, with an ECE of 0.063 against 0.128 under the leaderboard’s 15 bins and an NLL of 0.72 against 2.43.

On typed-decisions, Bongard’s distributions are closer to the teacher’s than Jev’s are, with a KL of 0.256 against 1.442 and a Brier score of 0.132 against 0.148. Jev agrees with the teacher’s top label more often, at 72.7% against Bongard’s 59.4%. The two kinds of measure capture different properties of a judgment: accuracy concerns only the top-ranked option, whereas KL and Brier score compare the full distribution with the target. ImajevBench asks for decisions from photos and rules, and with the photo withheld, Bongard answers “unknown” on 102 of the 103 visual items instead of guessing.

7.2 Same-item comparison with Jev

Because leaderboards compare systems on different samples and settings, we sent Bongard’s requests unchanged to Jev 1.13 through TypeSafe’s API on 30 September 2026 (Figure 4). Most item sets come from the published benchmark code of Laya and Jeff (Convai Innovations, 2026b; Strasser, 2026), and the others are complete public splits (Table 12). Both models answered every item. Our API runs reproduce four independent Jev measurements within one point (Bakhta, 2026; nibzard, 2026).

The open 4B-4B Bongard leads the hosted Jev on six benchmarks and stays within three points on

Table 8: Public System One benchmarks. DecisionBench, typed-decisions and ImajevBench references come from their leaderboards and dataset cards on 29 September 2026, and JevBench references from its v1.4.2.2 results. On typed-decisions, the gold is a teacher distribution, so the scores measure agreement with that teacher.

Benchmark	Measure	Bongard	Reference systems
DecisionBench 1.0 (Hanno Labs, 2026)	accuracy, all 23,900 rows	78.05% (4th of 61)	Imajev-4B 79.7%, Winnow-12B 76.7%, Jev 1.13 72.0%, DeepSeek V4.1 Flash 71.0%, GPT-5.6 Luna 69.9%
	ECE and NLL	0.063 and 0.72	Imajev-4B 0.069 and 0.69, Jev 1.13 0.128 and 2.43
JevBench v1.4 (Standhartinger and contributors, 2026), public items	accuracy, easy / standard	100% / 97.2%	Imajev-4B 100% / 99.0%
typed-decisions (LocalLLaMA, 2026), test	KL from gold, Brier	0.256, 0.132	Jev 1.13 1.442, 0.148
behavior-benchmark (Respan AI, 2026)	accuracy F_1 of <i>present</i> , core / multilingual	0.594 0.458 / 0.668	Jev 1.13 0.727, Jeff-Gemma4-E2B 0.561 no public results
ImajevBench v2.0-lite (Garg, 2026b)	accuracy, dev and calibration splits	66.9% (chance 26.5%)	test split, submitted

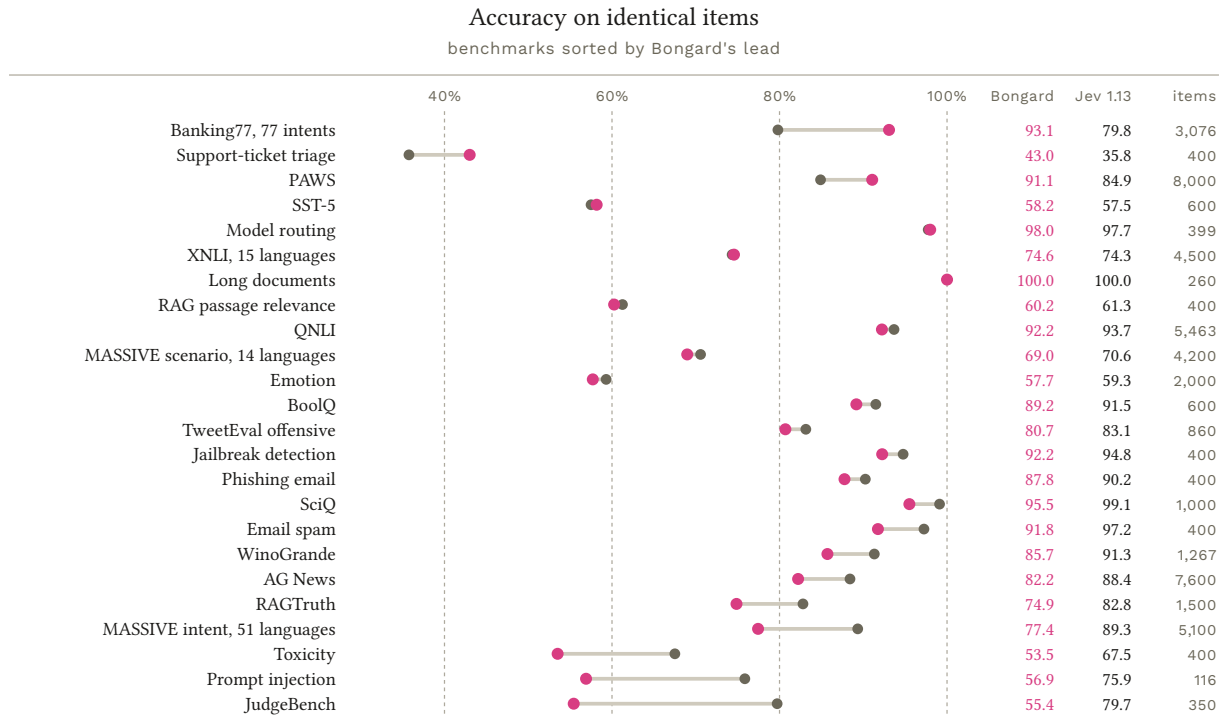


Figure 4: Same-item comparison with Jev 1.13. Both models answered the same 49,291 items from 24 public benchmarks, with identical states, instructions and candidates. Jev answered through TypeSafe’s API on 30 September 2026. Dots mark accuracy, and the benchmarks are sorted by Bongard’s lead. Table 12 lists the item sets.

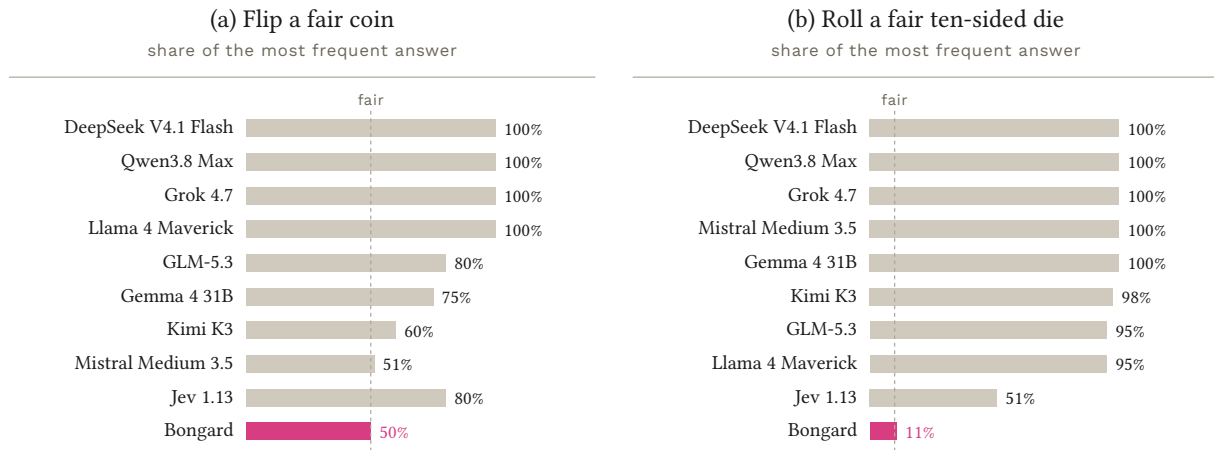


Figure 5: Asked for a fair random choice, chat models repeat one answer, while Bongard returns the uniform distribution. Each chat model answered each question 40 times at its default temperature. For Jev and Bongard, the bar is the largest returned probability. The dashed line marks a fair choice.

nine more. Its largest lead is on Banking77, where all 77 intents are candidates: 93.1% against 79.8%. It also leads on PAWS, with 91.1% against 84.9%, and on support-ticket triage.

On support-ticket triage, emotion and SST-5, whose labels are subjective, Jev is overconfident, with an ECE of 0.481, 0.277 and 0.179 against 0.077, 0.032 and 0.051 for Bongard.

7.3 Languages and long documents

Laya publishes results across 51 languages and for documents of up to 8,192 tokens, and we rebuilt its items from its code (Convai Innovations, 2026a,b). In its MASSIVE intent sweep, each question offers 20 candidate intents, and 100 questions cover each language (FitzGerald et al., 2023). Bongard outperforms both Laya checkpoints in every language. Its mean accuracy is 77.4%, against 36.6% for Laya’s multilingual checkpoint and 22.7% for its English one. Its weakest language, Welsh, still reaches 48%.

Laya’s long-document test places a support request after up to 7,000 tokens of meeting notes, and we extend it to 30,000 tokens. The model must assign each request to a department. Bongard assigns all 260 requests correctly, at every length. Laya’s multilingual checkpoint reads at most 8,192 tokens, and it assigns 123 of the 160 requests up to 7,000 tokens correctly.

7.4 Comparison with the pretrained backbone

The T5Gemma 2 report scores the pretrained 4B-4B model on three tasks that we also evaluate (Zhang et al., 2025c), so these tasks show how post-training converts the knowledge of the backbone into direct judgments. Bongard answers each question in one pass of its judgment head, with no examples in the request. It reaches 89.2% on BoolQ against 79.3%, and 60.9% on SocialIQA against 49.9% (Clark et al., 2019; Sap et al., 2019). On WinoGrande, it reaches 85.7% against 71.6% for the backbone with five examples (Sakaguchi et al., 2020).

7.5 Fair random choices

A decision model should also recognise when the evidence favours no option. We asked eight chat models 18 fair-chance questions, such as a coin flip, a die roll, a card suit and a tie-break, 40 times each (Figure 5). DeepSeek V4.1 Flash, Qwen3.8 Max, Grok 4.7 and Llama 4 Maverick answered “heads” 40 times out of 40. On a ten-sided die, seven of the eight models answered “7” in 95–100% of trials. Jev gives



Figure 6: Time and cost per decision. Bongard runs on one RTX PRO 6000 in BF16, and its cost assumes \$1.79 per GPU hour. Latencies of the other systems come from the DecisionBench leaderboard and include network time. Costs come from the JevBench results, as dollars per 1,000 decisions times 1,000.

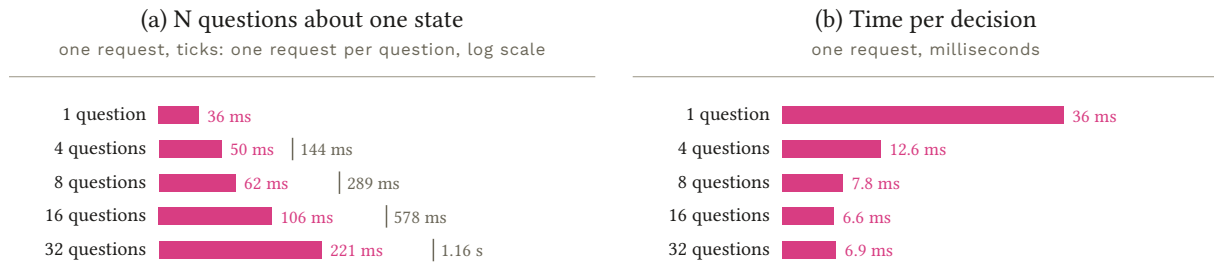


Figure 7: Latency of shared-state requests. One request asks N questions about the same JevBench state, and the encoder reads the state once. Ticks mark N separate single-question requests, at N times the single-question latency. Thirty-two questions take 221 ms in one request instead of 1.16 s, so the time per decision falls from 36 to 6.9 ms. One RTX PRO 6000, BF16.

“heads” a probability of 0.80. Bongard returns 0.500 for each side, and its mean KL from the uniform distribution over all 18 questions is 0.0009, against 0.769 for Jev.

7.6 Speed and cost

One request at a time, the median latency is 36 ms on short JevBench items, and the mean over DecisionBench rows is 91 ms (Figure 6). Questions about the same state share one encoder pass (Figure 7). Thirty-two such questions take 221 ms in one request, 5.2 times less than as separate requests, or 6.9 ms per decision. Batched requests reach about 400,000 decisions per hour on one GPU. At \$1.79 per GPU hour, a million decisions cost about \$4.4. Jev costs \$40 and GPT-6 Luna \$127 for the same number.

7.7 Decisions in games

Games test decisions that must follow each other quickly (Table 9). In each board game, the engine computes objective facts about each legal move, such as the score, the free cells and the replies it allows. It writes them into the state, and a single sentence states the strategy. Bongard then chooses the move, so the same model can play a new game from a new sentence. In 2048 it reached the 2048 tile in 9 of 12 games. In Tetris it never topped out in 1,500 pieces.

7.8 What stage 3 adds

Stage 3 raises accuracy on every held-out set in Table 10. The largest gains are on hard-style decisions, from 34.9% to 50.5%, and on frontier-style decisions, from 45.2% to 73.9%. Fair-chance questions become almost exactly uniform. Serving temperatures improve calibration further. On the external development sets, NLL falls from 0.824 to 0.691 and ECE from 0.141 to 0.030.

Table 9: Games played by Bongard. The game engine lists the legal moves and states the facts of each move in the state. The question gives a one-line strategy in plain English, and the model chooses every move. For 2048 and Tetris, the facts include the engine’s rating of each move. Doom is ViZDoom Defend the Center (Kempka et al., 2016), and the model reads a one-line description of the scene every 0.2 s.

Game	Games	Result
2048 (Cirulli, 2014)	12	reaches the 2048 tile in 9 games and 4096 in 2, best score 55,140
Tetris	5	never tops out within the limit of 1,500 pieces, 596–599 rows cleared per game
Othello	32 + 32	wins 24 of 32 against a greedy engine and 4 of 32 against a two-ply positional engine, one of them 13–0 in nine moves
Minesweeper, 8×8	64	clears the board in 42 games
Snake, 8×8	32	grows to 32 cells, half of the board
Doom	32	10.3 kills per 30-second episode on average, 20 at best

Table 10: Stage 2 (JEPA) and the final model on evaluation sets excluded from stage 3 training. The style-based sets are internal panels, and the public-benchmark samples cover multiple tasks.

Evaluation set	Stage 2	Final
Hard-style decisions (109)	34.9%	50.5%
Sealed-style decisions (200)	35.0%	50.5%
Frontier-style decisions (199)	45.2%	73.9%
DecisionBench, balanced sample (2,580)	64.3%	68.7%
RouterBench routing between cheap and strong models (600)	56.2%	62.5%
JevBench public standard items (72)	93.1%	97.2%
Fair-chance KL from uniform (18 questions)	0.173	0.0009

8 Related work

System One models. TypeSafe introduced System One models, Jev and its typed API (TypeSafe AI, 2026b,a). From the behaviour of that API, Hume (2026) infers a decoder-only backbone with prefix caching. Ling et al. (2026) analyse 2,170 public projects that use Jev as a reusable decision component. Zhang (2026) pairs a planning vision-language model with Jev as a fast executor of mobile GUI actions. This design cuts execution time and cost at a small loss in success rate. Further open systems behind the System One API include the Open-Jev project of Cai (2026), Cygnet (blockbrain, 2026) and decider (Mapika, 2026), and JevBench compares many of them (Standhartinger and contributors, 2026).

Calibration and single-pass reasoning. Decision calibration asks that probabilities be reliable for the decisions that they drive (Zhao et al., 2021a). Proper-scoring rewards train the verbalised confidence of generative models (Bani-Harouni et al., 2025; Damani et al., 2025), and community RLCD-style projects use them too (anthony-maio, 2026; TianyuCodings, 2026). Beyond chess and Othello (Section 1), single-pass transformers can also internalise step-by-step reasoning (Deng et al., 2024).

9 Limitations

The alignment terms are isolated only by a short independent control, which shows a reshaped representation but not yet a decision gain. The head’s subspace also captures less of g than before, so a full-length control with $\lambda = 0$ and a readout that uses the reshaped representation come next. Stage 3 tracks its progress on a panel from its own environments. Evaluation sets excluded from stage 3 training improve as well (Table 10), but judgment of consequences in unseen environments is not yet tested directly. Verifying worked answers is less reliable than other judgments, because the model tends to

accept a wrong answer as correct.

Jev still leads on several kinds of judgment. On typed-decisions, it agrees with the teacher’s top label more often. On identical items, it leads by more than three points on 9 of 24 public benchmarks, most on JudgeBench, prompt injection and toxicity.

The stage-1 and stage-2 checkpoints encode the state alone, so their encoder cannot focus on what a question concerns. Question-aware encoding removes this limit at a cost. An answer can change when other instructions join the request, and the encoder cache serves only requests with the same instructions.

10 Conclusion

We present Bongard, an open System One model that formulates machine intuition as direct, callable probabilistic judgment. It combines an encoder–decoder backbone, which separates the reading of a state from the judgments made about it, with joint-embedding representation learning and sandbox outcome feedback. Each of its three training stages updates all trainable parameters on a single GPU.

Joint-embedding training reshapes the decision representation, and sandbox training improves outcome judgments and held-out task scores. The final model reaches 78.05% accuracy on DecisionBench and answers 32 questions about one state in 221 ms. The released weights and runtime, together with the training method described here, provide a concrete route to machine intuition. Further work will examine how to use the learned representation more fully and how to allocate capacity between the encoder and the decoder.

References

- allebee. JevK5. <https://github.com/allebee/jevk5>, 2026.
- Jason Ansel et al. 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*, 2024.
- anthony-maio. eve-rlcd. <https://github.com/anthony-maio/eve-rlcd>, 2026.
- Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-Supervised Learning from Images with a Joint-Embedding Predictive Architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.
- Mido Assran et al. V-JEPA 2: Self-Supervised Video Models Enable Understanding, Prediction and Planning. *arXiv preprint arXiv:2506.09985*, 2025.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program Synthesis with Large Language Models. *arXiv preprint arXiv:2108.07732*, 2021.
- Abdelhamid Bakhta. jev-benchmarks: Probability-Aware Evaluation for Typed Decision Models. <https://github.com/AbdelStark/jev-benchmarks>, 2026.
- David Bani-Harouni, Chantal Pellegrini, Paul Stangel, Ege Özsoy, Kamilia Zaripova, Nassir Navab, and Matthias Keicher. Rewarding Doubt: A Reinforcement Learning Approach to Calibrated Confidence Expression of Large Language Models. *arXiv preprint arXiv:2503.02623*, 2025.
- Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, 2020.
- Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting Feature Prediction for Learning Visual Representations from Video. *arXiv preprint arXiv:2404.08471*, 2024.
- blockbrain. Cygnet Recipe. <https://github.com/blockbrain-ai/cygnet-recipe>, 2026.
- Mikhail M. Bongard. *Pattern Recognition*. Spartan Books, New York, 1970.

- Glenn W. Brier. Verification of Forecasts Expressed in Terms of Probability. *Monthly Weather Review*, 78(1):1–3, 1950.
- Browser Use. Jev Ultrafast. <https://github.com/browser-use/jev-ultrafast>, 2026.
- Zefan Cai. Open-Jev. <https://github.com/Zefan-Cai/Open-Jev>, 2026.
- Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. Efficient Intent Detection with Dual Sentence Encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 38–45, 2020.
- Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training Deep Nets with Sublinear Memory Cost. *arXiv preprint arXiv:1604.06174*, 2016.
- Gabriele Cirulli. 2048. <https://github.com/gabrielecirulli/2048>, 2014.
- Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2924–2936, 2019.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. XNLI: Evaluating Cross-lingual Sentence Representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, 2018.
- Contrastive-LM. CLM-v0.1-8B. <https://github.com/Contrastive-LM/CLM>, 2026.
- Convai Innovations. Laya: A Typed-Decisions Model. <https://huggingface.co/convaiinnovations/laya-typed-decisions>, 2026a.
- Convai Innovations. Laya: Code and Benchmark Results. <https://github.com/NandhaKishorM/laya>, 2026b.
- crh225. Plumb-4B. <https://github.com/crh225/plumb>, 2026.
- Mehul Damani, Isha Puri, Stewart Slocum, Idan Shenfeld, Leshem Choshen, Yoon Kim, and Jacob Andreas. Beyond Binary Rewards: Training LMs to Reason About Their Uncertainty. *arXiv preprint arXiv:2507.16806*, 2025.
- Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In *Advances in Neural Information Processing Systems*, volume 35, 2022.
- Anna Dawid and Yann LeCun. Introduction to Latent Variable Energy-Based Models: A Path Towards Autonomous Machine Intelligence. *arXiv preprint arXiv:2306.02572*, 2023.
- deepset. prompt-injections. <https://huggingface.co/datasets/deepset/prompt-injections>, 2023.
- Yuntian Deng, Yejin Choi, and Stuart Shieber. From Explicit CoT to Implicit CoT: Learning to Internalize CoT Step by Step. *arXiv preprint arXiv:2405.14838*, 2024.
- denis-pplx. AutoJev-27B. <https://github.com/denis-pplx/autojev>, 2026.
- Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit Optimizers via Block-wise Quantization. In *International Conference on Learning Representations*, 2022.
- Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex Attention: A Programming Model for Generating Optimized Attention Kernels. *arXiv preprint arXiv:2412.05496*, 2024.
- Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention Is Not All You Need: Pure Attention Loses Rank Doubly Exponentially with Depth. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.
- ekzhang. openjev-sglang. <https://github.com/ekzhang/openjev-sglang>, 2026.
- Kawin Ethayarajh. How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2019.
- Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, Swetha Ranganath, Laurie Crist, Misha Britan, Wouter Leeuwis, Gokhan Tur, and Prem Natarajan. MASSIVE: A 1M-Example Multilingual Natural Language Understanding Dataset with 51 Typologically-Diverse Languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 4277–4302, 2023.

- Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021.
- Mohit Garg. imajev 1.0: Typed Decisions from Photos and App State. <https://mohit67890.github.io/imajev/report/>, September 2026a. Technical report.
- Mohit Garg. ImajevBench v2.0-lite. <https://huggingface.co/datasets/mohit67890/imajev-bench>, 2026b.
- Gemma Team. Gemma 3 Technical Report. *arXiv preprint arXiv:2503.19786*, 2025.
- Taj Gillin, Adam Lalani, Kenneth Zhang, and Marcel Mateos Salles. BERT-JEPA: Reorganizing CLS Embeddings for Language-Invariant Semantics. *arXiv preprint arXiv:2601.00366*, 2026.
- Tilman Gneiting and Adrian E. Raftery. Strictly Proper Scoring Rules, Prediction, and Estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.
- 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*, 2017.
- Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep Learning with Limited Numerical Precision. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.
- Hanno Labs. DecisionBench 1.0. <https://huggingface.co/datasets/Hanno-Labs/decision-bench>, 2026.
- Awni Hannun, Jagrit Digani, Angelos Katharopoulos, and Ronan Collobert. MLX: Efficient and Flexible Machine Learning on Apple Silicon. <https://github.com/ml-explore/mlx>, 2023.
- Hai Huang, Yann LeCun, and Randall Balestriero. LLM-JEPA: Large Language Models Meet Joint Embedding Predictive Architectures. *arXiv preprint arXiv:2509.14252*, 2025.
- Archer Hume. Jev’s Architecture Unmasked. <https://archerhume.com/posts/jevs-architecture-unmasked/>, September 2026. Blog post.
- Gautier Izacard and Edouard Grave. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, 2021.
- Pavel Izmailov, Dmitrii Podoprikin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging Weights Leads to Wider Optima and Better Generalization. In *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence*, 2018.
- Jordan Juravsky, Bradley Brown, Ryan Ehrlich, Daniel Y. Fu, Christopher Ré, and Azalia Mirhoseini. Hydragen: High-Throughput LLM Inference with Shared Prefixes. *arXiv preprint arXiv:2402.05099*, 2024.
- Daniel Kahneman. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, New York, 2011.
- Daniel Kahneman and Gary Klein. Conditions for Intuitive Expertise: A Failure to Disagree. *American Psychologist*, 64(6):515–526, 2009. doi: 10.1037/a0016755.
- Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games*, pages 341–348, 2016.
- kiconcuo. Jevfire. <https://github.com/kiconcuo/jevfire>, 2026.
- Yann LeCun. A Path Towards Autonomous Machine Intelligence. OpenReview preprint, <https://openreview.net/forum?id=BZ5a1r-kVsf>, 2022.
- Delong Li, Xu Wang, Haochen Gong, Rui Lang, and Guangsheng Yu. Replacing Large Language Models with Jev Decision Models for Low-Latency Edge Service Orchestration. *arXiv preprint arXiv:2609.22753*, 2026.
- Kenneth Li, Aspen K. Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent World Representations: Exploring a Sequence Model Trained on a Synthetic Task. In *International Conference on Learning Representations*, 2023.
- Zi Lin, Zihan Wang, Yongqi Tong, Yangkun Wang, Yuxin Guo, Yujia Wang, and Jingbo Shang. ToxicChat: Unveiling Hidden Challenges of Toxicity Detection in Real-World User-AI Conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 4694–4702, 2023.
- Guoming Ling, Muen Xue, and Zijian Ye. Jev in the Wild: A Data-Driven Analysis of the Jev Model’s Functionality, Applications and Ecosystem. *arXiv preprint arXiv:2609.30216*, 2026.
- Zefang Liu. Phishing Email Dataset. <https://huggingface.co/datasets/zefang-liu/phishing-email-dataset>, 2024.
- LocalLLaMA. typed-decisions. <https://huggingface.co/datasets/LocalLLaMA/typed-decisions>, 2026.
- Mapika. decider. <https://github.com/Mapika/decider>, 2026.

- Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. Spam Filtering with Naive Bayes – Which Naive Bayes? In *Third Conference on Email and Anti-Spam*, 2006.
- Paulius Micikevicius, Dusan Stosic, Neil Burgess, Marius Cornea, Pradeep Dubey, Richard Grisenthwaite, Sangwon Ha, Alexander Heinecke, Patrick Judd, John Kamalu, Naveen Mellempudi, Stuart Oberman, Mohammad Shoeybi, Michael Siu, and Hao Wu. FP8 Formats for Deep Learning. *arXiv preprint arXiv:2209.05433*, 2022.
- Microsoft. Playwright. <https://playwright.dev>, 2026.
- MoLeMo Lab. MoJev. <https://huggingface.co/MoLeMo-Lab/mojev>, 2026.
- MotherDuck. Introducing prompt_jev(): Bringing Jev to MotherDuck SQL. <https://motherduck.com/blog/motherduck-supports-jev/>, 2026. Blog post.
- Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A Human Generated MACHine Reading COMprehension Dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches*, 2016.
- nibzard. DMB: Decision-Model Benchmark. <https://github.com/nibzard/decision-model-benchmark>, 2026.
- Cheng Niu, Yuanhao Wu, Juno Zhu, Siliang Xu, KaShun Shum, Randy Zhong, Juntong Song, and Tong Zhang. RAGTruth: A Hallucination Corpus for Developing Trustworthy Retrieval-Augmented Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 10862–10878, 2024.
- Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020.
- NVIDIA. Pretraining Large Language Models with NVFP4. *arXiv preprint arXiv:2509.25149*, 2025.
- NVIDIA. Transformer Engine. <https://github.com/NVIDIA/TransformerEngine>, 2026.
- OpenJev contributors. OpenJev: An Open-Source System One Decision Server. <https://github.com/razorback16/openjev>, 2026.
- Jared Palmer and Kev contributors. Kev: Jev-like Decision Models Built on Qwen. <https://github.com/jaredpalmer/kev>, 2026. Implementation and model cards, accessed 30 September 2026.
- Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Xingjian Du, Haowen Hou, Jiaju Lin, Jiaying Liu, Janna Lu, William Merrill, Guangyu Song, Kaifeng Tan, Saiteja Utpala, Nathan Wilce, Johan S. Wind, Tianyi Wu, Daniel Wuttke, and Christian Zhou-Zheng. RWKV-7 “Goose” with Expressive Dynamic State Evolution. *arXiv preprint arXiv:2503.14456*, 2025.
- PyTorch Team. TorchAO: PyTorch Architecture Optimization. <https://github.com/pytorch/ao>, 2026.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.
- Respan AI. behavior-benchmark. <https://huggingface.co/datasets/respanai/behavior-benchmark>, 2026.
- Anian Ruoss, Grégoire Delétang, Sourabh Medapati, Jordi Grau-Moya, Li Kevin Wenliang, Elliot Catt, John Reid, Cannada A. Lewis, Joel Veness, and Tim Genewein. Amortized Planning with Large-Scale Transformers: A Case Study on Chess. *arXiv preprint arXiv:2402.04494*, 2024.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740, 2020.
- Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense Reasoning about Social Interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4463–4473, 2019.
- Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. CARER: Contextualized Affect Representations for Emotion Recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, 2018.
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, 2013.
- Florian Standhartinger and contributors. JevBench: A Benchmark for System One Decision Models. <https://github.com/fstandhartinger/jevbench>, 2026.
- Mathias Strasser. Jeff. <https://github.com/firelex/jeff>, 2026.

- Jianlin Su. Why Are Current LLMs All Decoder-Only Architectures? <https://spaces.ac.cn/archives/9529>, 2023. Blog post, Scientific Spaces (in Chinese).
- Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y. Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. JudgeBench: A Benchmark for Evaluating LLM-Based Judges. In *International Conference on Learning Representations*, 2025.
- Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. UL2: Unifying Language Learning Paradigms. In *International Conference on Learning Representations*, 2023.
- TheoLeeCJ. SemIf. <https://github.com/TheoLeeCJ/SemIf>, 2026.
- TianyuCodings. NanoJev. <https://github.com/TianyuCodings/NanoJev>, 2026.
- Tobi-Bueck. Customer Support Tickets. <https://huggingface.co/datasets/Tobi-Bueck/customer-support-tickets>, 2025.
- Philip Turner. Metal FlashAttention. <https://github.com/philipturner/metal-flash-attention>, 2024.
- TypeSafe AI. System One API Documentation: Primitives and Parallel Questions. <https://docs.typesafe.ai>, 2026a. Accessed September 2026.
- TypeSafe AI. Introducing System One Models and Jev. <https://typesafe.ai/blog/introducing-system-one-models-and-jev>, 2026b. Blog post.
- Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding. *arXiv preprint arXiv:1807.03748*, 2018.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *International Conference on Learning Representations*, 2019.
- Tongzhou Wang and Phillip Isola. Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.
- Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing Multiple Choice Science Questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, 2017.
- Ronald J. Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8:229–256, 1992.
- Timothy D. Wilson and Jonathan W. Schooler. Thinking Too Much: Introspection Can Reduce the Quality of Preferences and Decisions. *Journal of Personality and Social Psychology*, 60(2):181–192, 1991. doi: 10.1037/0022-3514.60.2.181.
- Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model Soups: Averaging Weights of Multiple Fine-Tuned Models Improves Accuracy without Increasing Inference Time. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.
- Ted Zadouri, Markus Hoehnerbach, Jay Shah, Timmy Liu, Vijay Thakkar, and Tri Dao. FlashAttention-4: Algorithm and Kernel Pipelining Co-Design for Asymmetric Hardware Scaling. In *Proceedings of Machine Learning and Systems*, 2026.
- Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid Loss for Language Image Pre-Training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.
- Biao Zhang, Yong Cheng, Siamak Shakeri, Xinyi Wang, Min Ma, and Orhan Firat. Encoder-Decoder or Decoder-Only? Revisiting Encoder-Decoder Large Language Model. *arXiv preprint arXiv:2510.26622*, 2025a.
- Biao Zhang, Fedor Moiseev, Joshua Ainslie, Paul Suganthan, Min Ma, Surya Bhupatiraju, Fede Lebron, Orhan Firat, Armand Joulin, and Zhe Dong. Encoder-Decoder Gemma: Improving the Quality-Efficiency Trade-Off via Adaptation. *arXiv preprint arXiv:2504.06225*, 2025b.
- Biao Zhang, Paul Suganthan, Gaël Liu, Ilya Philippov, Sahil Dua, Ben Hora, Kat Black, Gus Martins, Omar Sanseviero, Shreya Pathak, Cassidy Hardin, Francesco Visin, Jiageng Zhang, Kathleen Kenealy, Qin Yin, Olivier Lacombe, Armand Joulin, Tris Warkentin, and Adam Roberts. T5Gemma 2: Seeing, Reading, and Understanding Longer. *arXiv preprint arXiv:2512.14856*, 2025c.
- Linghua Zhang. Jev-Mobile: Jev as an Executor for Mobile GUI Agents. *arXiv preprint arXiv:2609.30186*, 2026.
- Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level Convolutional Networks for Text Classification. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

- Yuan Zhang, Jason Baldridge, and Luheng He. PAWS: Paraphrase Adversaries from Word Scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1298–1308, 2019.
- Shengjia Zhao, Michael P. Kim, Roshni Sahoo, Tengyu Ma, and Stefano Ermon. Calibrating Predictions to Decisions: A Novel Approach to Multi-Class Calibration. In *Advances in Neural Information Processing Systems*, volume 34, 2021a.
- Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate Before Use: Improving Few-Shot Performance of Language Models. In *Proceedings of the 38th International Conference on Machine Learning*, 2021b.
- Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. Large Language Models Are Not Robust Multiple Choice Selectors. In *International Conference on Learning Representations*, 2024.
- Zhanchao Zhou, Tianyi Wu, Zhiyun Jiang, Fares Obeid, and Zhenzhong Lan. Value Residual Learning. *arXiv preprint arXiv:2410.17897*, 2024.
- Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. RankT5: Fine-Tuning T5 for Text Ranking with Ranking Losses. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2023.

A Interface details

Each question compiles to a separate decoder sequence. The sequence for a Choice question is:

```
<bos>type: choice
instructions: "Which stock category applies?"
candidate: {"name":"empty","description":"No items."}<candidate_end>
candidate: {"name":"low","description":"1-7 items."}<candidate_end>
candidate: {"name":"high","description":"8+ items."}<candidate_end>
<decision_end>
```

Noul uses the candidates true and false, and Score lists its levels in order. Choice confidence is the margin of the top probability over the uniform probability, normalised to $[0, 1]$: $(p_{\max} - 1/K)/(1 - 1/K)$. Score confidence is $1 - d/d_u$, where d is the mean distance of the distribution from its mode and d_u is the mean distance of a uniform distribution from the same mode. The following listing shows a request and an illustrative response.

```
{
  "state": {
    "available": 7,
    "questions": {
      "enough": {
        "type": "noul",
        "instructions": "Are at least 8 items available?"
      },
      "stock": {
        "type": "choice",
        "instructions": "Which stock category applies?",
        "criteria": {
          "empty": "No items are available.",
          "low": "Between 1 and 7 items are available.",
          "high": "At least 8 items are available."
        }
      },
      "level": {
        "type": "score",
        "instructions": "Rate the available stock on this scale.",
        "criteria": [
          "No items are available.",
          "Between 1 and 7 items are available.",
          "At least 8 items are available."
        ]
      }
    }
  },
  "model": "bongard-mini",
  "answers": {
    "enough": {
      "type": "noul",
      "noul": 0.03
    },
    "stock": {
      "type": "choice",
      "choice": "low",
      "confidence": 0.94,
      "probabilities": {
        "empty": 0.01,
        "low": 0.96,
        "high": 0.03
      }
    },
    "level": {
      "type": "score",
      "score": 1.03,
      "confidence": 0.925,
      "probabilities": {
        "0": 0.01,
        "1": 0.95,
        "2": 0.04
      },
      "legend": {
        "0": "No items are available.",
        "1": "Between 1 and 7 items are available.",
        "2": "At least 8 items are available."
      }
    }
  },
  "usage": {
    "input_tokens": 212,
    "output_tokens": 0
  }
}
```

B Stage-1 task families

Stage 1 covers language and document judgments, structured reasoning, retrieval, images, computer use, operational decisions and probability questions. Depending on the task, targets come from annotations, executable rules, exact mechanisms or filtered teacher judgments. Related examples remain in the same split.

C Stage-2 details

Table 11: Relation families. Separately encoded families carry the content objective, and the answer-state family carries the answer contrast. Other shared-state families receive supervision on both views but no alignment term.

Family	View A $\leftrightarrow$ view B	Encoding	Alignment
Query $\rightarrow$ passage	Query with a relevance task $\leftrightarrow$ the passage’s content	Separate	Content
Question $\rightarrow$ answer	Question over evidence $\leftrightarrow$ the answer’s content	Separate	Content
Image $\rightarrow$ content	Image with a scoped prediction $\leftrightarrow$ its annotated content	Separate	Content
World prediction	World and action $\leftrightarrow$ observed next state	Separate	Content
Causal prediction	Causal model and intervention $\leftrightarrow$ observed outcome	Separate	Content
Assignment and matching	Problem instance $\leftrightarrow$ verified allocation	Separate	Content
Paraphrase	Sentence $\leftrightarrow$ meaning-preserving rewrite	Separate	Content
Program $\rightarrow$ result	Program and input $\leftrightarrow$ re-executed result	Separate	Content
Answer state	Multiple-choice question $\leftrightarrow$ correct option (yes) and wrong option (no)	Shared	Answer
Logical complement	Noul question $\leftrightarrow$ its negation	Shared	None
Inherited view	Verified equivalent views	Shared	None

Training set and schedule. We balance relation families and replay earlier tasks to limit drift. Held-out split groups are selected before view construction, so related origins do not cross the split. The alignment weight warms up over the first 5% of updates. A pack with fewer than four members of a kind skips the contrast for that kind.

Panel details. On the frozen panel, conditional-event accuracy for the correct option’s “yes” rose from 64.2% to 85.3%. For a wrong option’s “no”, it rose from 85.4% to 92.8%. Image-evidence invariance improved: the total variation fell from 0.017 to 0.006.

D Sandbox environments

Stage 3 uses games, grid worlds, verifiable judgments, business workflows, computer use, classification and executable generators. States can be text, structured records or accessibility snapshots. The model answers typed questions about candidate actions. Outcome labels come from exact computation, recorded observations or rollouts under the stated continuation policy.

E Training-stack details

NVFP4 feed-forward blocks. NVFP4 is a 4-bit floating-point format with two-level block scaling, and Blackwell GPUs execute it natively (NVIDIA, 2025). The model has 68 feed-forward blocks, 34 per stack. Each block is a fused Transformer Engine module that reuses the original parameters and maps back to the standard checkpoint keys on save. In BF16, the fused block reproduces the original module to a relative L_2 error of 0.0036. Saved checkpoints load in the standard Transformers implementation with identical logits. In stage 3, the feed-forward projections run as NVFP4 Transformer Engine linear layers instead, with one matrix multiplication for the gate and up projections. Its checkpoints also use the standard layout.

FP8 attention projections. In stages 1 and 2, the 272 attention projections, four per layer, are FP8 matrix multiplications with rowwise scaling. Attention arithmetic stays in BF16, because the FlashAttention-4 kernels for head dimension 256 in variable-length layouts do not support FP8 training.

Packing. Records are packed without padding into streams of states and questions, and positions restart at zero for each sequence. In the encoder, each state attends only to its own tokens. In the decoder, each question attends only to its own prefix and its own state. Sequences long enough to trigger the

1,024-token sliding window go through FlexAttention (Dong et al., 2024). Long and short sequences are split only inside attention.

Memory. Parameters, gradients, embeddings and residuals are stored in BF16. The 8-bit AdamW state takes one byte per parameter per moment. Stochastic rounding keeps small updates from vanishing without an FP32 master copy. State keys and values are stored once per record and gathered again for each question in the backward pass, with shared gradients accumulated in FP32. Only records longer than the pack limit use per-layer recomputation, so no input is truncated. Peak allocation was 179 GB in stage 2 and 239 GB in the warm start of stage 3.

Kernel overhead. We batch the readout indexing and the finiteness checks per update. This change cut the kernel count of a steady-state pack from 21,528 to 17,948. Tokeniser settings, compiled requests and readout positions are stored with the data, so training reads precomputed tokens and token costs.

Other hardware. The CPU and Apple Silicon paths share the model code. For smaller machines, a LoRA configuration applies rank-8 adapters to the attention projections and also trains the projector and the head.

F Same-item comparison

Laya and Jeff item sets are rebuilt from their published code, with their states, instructions and candidates (Convai Innovations, 2026b; Strasser, 2026). Elsewhere, one sentence states the task, and the candidates are the benchmark’s own labels. Yes/no tasks are Noul questions (Table 12).

Table 12: Item sets and accuracy of the same-item comparison, in the order of Figure 4. Test and validation splits are complete.

Benchmark	Item set	Items	Accuracy (%)	
			Bongard	Jev
Banking77 (Casanueva et al., 2020)	test	3,076	93.1	79.8
Support-ticket triage (Tobi-Bueck, 2025)	Laya	400	43.0	35.8
PAWS (Zhang et al., 2019)	test	8,000	91.1	84.9
SST-5 (Socher et al., 2013)	Laya	600	58.2	57.5
Model routing (Cobbe et al., 2021; Austin et al., 2021; Zhang et al., 2015)	Laya	399	98.0	97.7
XNLI, 15 languages (Conneau et al., 2018)	Laya	4,500	74.6	74.3
Long documents (Convai Innovations, 2026b)	Laya	260	100.0	100.0
RAG passage relevance (Nguyen et al., 2016)	Laya	400	60.2	61.3
QNLI (Wang et al., 2019)	validation	5,463	92.2	93.7
MASSIVE scenario, 14 languages (FitzGerald et al., 2023)	Laya	4,200	69.0	70.6
Emotion (Saravia et al., 2018)	test	2,000	57.7	59.3
BoolQ (Clark et al., 2019)	Laya	600	89.2	91.5
TweetEval offensive (Barbieri et al., 2020)	test	860	80.7	83.1
Jailbreak detection (Lin et al., 2023)	Laya	400	92.2	94.8
Phishing email (Liu, 2024)	Laya	400	87.8	90.2
SciQ (Welbl et al., 2017)	test	1,000	95.5	99.1
Email spam (Metsis et al., 2006)	Laya	400	91.8	97.2
WinoGrande (Sakaguchi et al., 2020)	validation	1,267	85.7	91.3
AG News (Zhang et al., 2015)	test	7,600	82.2	88.4
RAGTruth (Niu et al., 2024)	Jeff	1,500	74.9	82.8
MASSIVE intent, 51 languages (FitzGerald et al., 2023)	Laya	5,100	77.4	89.3
Toxicity (Lin et al., 2023)	Laya	400	53.5	67.5
Prompt injection (deepset, 2023)	Laya	116	56.9	75.9
JudgeBench (Tan et al., 2025)	Jeff	350	55.4	79.7

G Contributions and acknowledgements

Authors. Li Ding, Haidi Jin and Chen Ji (AgentBull Pte Ltd).

Contributions. Li Ding conceived the project, designed the architecture and the training programme, wrote the code, ran all training and evaluation, and wrote the report. Haidi Jin and Chen Ji processed the training datasets.

Acknowledgements. We thank Google for the open T5Gemma 2 weights, and the maintainers of JevBench, DecisionBench, typed-decisions, behavior-benchmark and ImajevBench for their public benchmarks and results. We also thank the authors of Laya, Jeff, jev-benchmarks and DMB for their open benchmark code. Li Ding is also grateful to Douglas Richard Hofstadter, whose writings and ideas inspired his thinking and whose books introduced him to the Bongard problems that gave the model its name. He thanks his wife, Angela Chan, and his dog, Nieh-Nieh, for their companionship throughout this work.