RESEARCH ARTICLE

# Nano-jev: Calibrated Cross-Encoder Micro-Decisions for Retrieval-Augmented Generation with Confidence Cascading

**Subham Divakar**[1,*] **and Rojalina Priyadarshini**[2,*]

[1] Department of Electronics and Telecommunications, C. V. Raman Global University, Bhubaneswar 752054, India
[2] Department of Computer Science, C. V. Raman Global University, Bhubaneswar 752054, India

## Abstract

**Retrieval-Augmented Generation (RAG) pipelines repeatedly make small judgments, such as whether a passage is relevant, whether the retrieved context is sufficient and whether an answer is grounded. These are usually delegated to the generating large language model (LLM), at notable latency and cost, or to score thresholds without a probabilistic interpretation. We study nano-jev, a 33.4 M-parameter cross-encoder that treats *relevance* (3-way), *sufficiency* and *groundedness* (binary) as typed, closed-set classification problems and returns temperature-calibrated probabilities. Each (question, option, context) triple is scored by a single-logit head, which makes predictions invariant to option order. On held-out halves of the HotpotQA, SQuAD 2.0 and MultiNLI validation sets, nano-jev reaches mean accuracies of 0.812 (relevance), 0.843 (sufficiency) and 0.841 (groundedness) over five seeds, comparable to a 1-pass classifier trained on the same data, which is 2.5× faster. Dedicated checkers such as MiniCheck remain stronger on groundedness (0.866). nano-jev runs 55–93× faster than a prompted 4 B-parameter LLM. On unseen datasets (MuSiQue, VitaminC), accuracy drops by 15–20 points and ECE rises from about 0.03 to about 0.17, so target-domain re-calibration is advisable. A confidence cascade that escalates uncertain decisions to the LLM improves in-distribution groundedness (0.868 vs. 0.841, $p = 0.02$) with about a quarter of decisions escalated, but trades accuracy for latency out of distribution. In a small end-to-end test, nano-jev relevance filtering improves HotpotQA answer F1 by 2.1 points at about 39 ms per question. Code and weights are released as an open-source package.**

**Keywords**: retrieval-augmented generation, cross-encoder, decision model, calibration, confidence cascade, natural language inference, question answering.

**Academic Editor:**
Javier Bajo

**Submitted:** 27 September 2026
**Revised:** 02 November 2026
**Accepted:** 05 November 2026
**Published:** 16 November 2026

**Vol.** 3, **No.** 4, 2026.
10.62762/TIS.2026.197904

***Corresponding authors:**
✉ Subham Divakar
shubham.divakar@gmail.com
✉ Rojalina Priyadarshini
priyadarshini.rojalina@gmail.com

## 1 Introduction

Retrieval-Augmented Generation (RAG) [1] is a widely used approach for grounding large language model (LLM) outputs in external knowledge. A

**Citation**
Divakar, S., & Priyadarshini, R. (2026). Nano-jev: Calibrated Cross-Encoder Micro-Decisions for Retrieval-Augmented Generation with Confidence Cascading. *ICCK Transactions on Intelligent Systematics*, 3(4), 245–260.

© 2026 ICCK (Institute of Central Computation and Knowledge)

typical RAG pipeline retrieves passages, determines which are useful, decides whether the retrieved context suffices, generates an answer, and checks whether the answer is faithful to the evidence. Each intermediate step involves a *judgment*: a small decision with a fixed set of outcomes.

These judgments are usually handled in one of three ways. (i) The generating LLM is prompted to self-assess, which adds latency and cost, and whose confidence estimates are not always reliable: the calibration of LLM self-evaluation depends on model scale and task [2], and token-level uncertainty in free-form generation is confounded by semantically equivalent outputs [3]. (ii) A reranker or NLI model provides a single score that is mapped to the decision's categories through hand-tuned thresholds. (iii) Deterministic heuristics (e.g. a retrieval-score cut-off) are used, which carry no uncertainty information.

In this paper we explore a complementary option: small, fast, *typed* decision models that take a question, a closed set of options and a context, and return calibrated probabilities. Such a model (i) has a well-defined output space per decision type, (ii) never generates free text, avoiding parsing failures, and (iii) can expose probabilities whose thresholds have an approximate frequency interpretation, at least in-distribution, which in turn allows confidence-based escalation to a larger model. Several lightweight evaluators already address parts of this space, notably for factual consistency [4, 5] and RAG evaluation [6, 7]. Our aim is not to replace them, but to examine how far a single small, calibrated, multi-decision model can go, and where it falls short.

**Contributions.**

1. **Typed decision formulation.** We cast three recurring RAG judgments (relevance, sufficiency, groundedness) as closed-set classification problems with a shared cross-encoder and an explicit option schema (Section 3.2).
2. **The nano-jev model and a controlled comparison.** We describe a 33.4 M-parameter cross-encoder that scores each (question, option, context) triple independently, and compare it with a standard 1-pass classifier trained on identical data and with compact task-specific checkers. The comparison clarifies when the per-option design is worth its $K$-fold inference cost (Sections 3.3, 5.3).
3. **Calibration analysis under shift.** Using a protocol in which checkpoint selection and temperature fitting use disjoint data, we find that temperature scaling gives low in-distribution ECE but degrades markedly on unseen domains (Section 5.4).
4. **Confidence cascade across all decision types.** We evaluate escalation to a 4 B-parameter LLM for relevance, sufficiency and groundedness, including cases where escalation does not help (Section 5.5).
5. **Robustness and realism checks.** We report multi-seed results with confidence intervals, a systematic evaluation of custom option sets, an analysis of 512-token truncation, RAG-style groundedness on LLM-AggreFact, and a small end-to-end RAG evaluation (Sections 5.6–5.9).
6. **Open release.** Code, training pipeline, evaluation scripts, baselines and weights are publicly available as a pip-installable package.

Section 2 reviews related work, Section 3 describes the method, Section 4 the experimental protocol, Section 5 the results, and Section 6 concludes.

## 2 Related Work

### 2.1 Retrieval-Augmented Generation

RAG [1] augments LLM generation with retrieved passages, and REALM [8] jointly trains retriever and reader. Self-RAG [9] adds reflection tokens so that the LLM critiques its own retrieval and generation. Corrective RAG (CRAG) [10] uses a lightweight evaluator to triage retrieved documents before generation. Like CRAG, we separate retrieval evaluation from generation, but study a typed, calibrated decision model rather than special tokens or a single-purpose evaluator.

### 2.2 Lightweight Evaluators for RAG and Factual Consistency

A growing body of work replaces LLM judges with compact models. SummaC [11] and the TRUE benchmark [12] show that NLI models, applied at the right granularity, are strong factual-consistency detectors. AlignScore [4] trains a unified alignment function on many tasks, and MiniCheck [5] trains small fact-checkers on synthetic data that approach LLM performance on the LLM-AggreFact benchmark at a fraction of the cost. For RAG evaluation, ARES [6] fine-tunes lightweight judges for context relevance, answer faithfulness and answer relevance, while RAGAS [7] proposes reference-free metrics computed with LLMs. RAGTruth [13] provides word-level

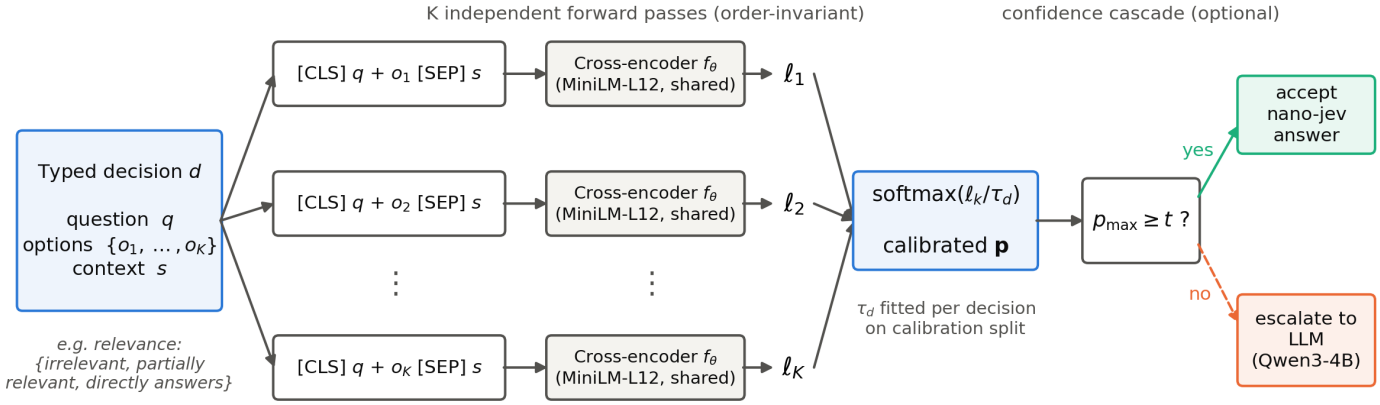



**Figure 1.** Nano-jev inference pipeline. A typed decision (e.g. *relevance*) specifies a question template and option set. Each (question + option, context) pair is encoded by a cross-encoder that outputs one logit. The option logits are temperature-scaled and softmaxed. If the cascade is active, decisions with top probability below threshold $t$ are escalated to an LLM.

hallucination annotations for RAG outputs. nano-jev is closest to ARES in covering several judgments with small models; it differs in using a single shared model with an explicit option schema, in emphasising calibrated probabilities for thresholding and cascading, and in targeting online decisions inside a pipeline rather than offline evaluation. On groundedness alone, dedicated checkers such as MiniCheck and AlignScore are strong references, and we include them as baselines.

### 2.3 Cross-Encoders, NLI and Label-as-Input Classification

Cross-encoders [14] process query–document pairs jointly with full cross-attention and are widely used as rerankers in multi-stage ranking pipelines. In contrast, bi-encoders such as Sentence-BERT [15] encode each text independently, which allows precomputed embeddings but forgoes joint attention over the pair. MS MARCO-trained MiniLM cross-encoders and BGE rerankers, which build on the BGE-M3 backbone [16], are common choices. BERT [17], RoBERTa [18] and DeBERTa-v3 [19] fine-tuned on NLI data remain strong entailment baselines. Encoding the candidate label together with the input, as nano-jev does, is related to entailment-based zero-shot classification [20], which we use as a reference for custom option sets.

### 2.4 Confidence Estimation and Calibration

Modern neural networks are often poorly calibrated [21]. Temperature scaling [21] fits a single scalar per task, and Platt scaling [22] fits a logistic regression. Kadavath et al. [2] find that larger LLMs are better calibrated in self-evaluation but still unreliable in some settings. We use per-decision temperature scaling and examine how well it transfers across domains.

### 2.5 Cascading and Adaptive Inference

FrugalGPT [23] cascades through progressively larger LLMs based on confidence, routing approaches [24] choose between local and cloud models per query, and speculative decoding [25] likewise pairs a small and a large model, but at the token level: the small model drafts and the large model verifies, to speed up generation without changing its output distribution. Our cascade instead operates on whole closed-set decisions and is a simple single-threshold rule on calibrated probabilities.

### 2.6 The Jev Decision-Model Paradigm

The Jev architecture [26] (TypeSafe AI) proposed compact, typed, non-generative "System 1" decision models for agentic workflows, arguing that runtime judgments such as classification, routing and scoring can be handled by a small model rather than a generative LLM. nano-jev is an independent open-source realisation of this idea for RAG-specific decisions, with explicit probability calibration, a cascade strategy and evaluation on held-out domains.

## 3 Proposed Method

### 3.1 Overview

Figure 1 illustrates the inference pipeline. A typed decision specifies a question template, a set of options and a context. For each option, a cross-encoder produces one logit; the logits are divided by a decision-specific temperature $\tau_d$ and softmaxed. In

**Table 1.** Built-in decision types. Each type has a question template, a fixed option set, and a context specification.

| Type | Options | Question template | Context |
|---|---|---|---|
| relevance | {irrelevant, partially relevant, directly answers} | How relevant is this passage to the query: `{query}` | single passage |
| sufficient | {yes, no} | Does the context contain enough information to answer: `{query}` | multiple passages |
| grounded | {yes, no} | Is this claim supported by the context? Claim: `{claim}` | evidence text |

the cascade variant, if the top probability falls below a threshold $t_d$, the decision is forwarded to an LLM.

### 3.2 Typed Decision Schema

A *decision type* $d$ is a triple $(\mathcal{Q}_d, \mathcal{O}_d, \mathcal{S}_d)$: a question template $\mathcal{Q}_d$, a fixed option set $\mathcal{O}_d = \{o_1, \ldots, o_K\}$, and a context type $\mathcal{S}_d$. Table 1 lists the three built-in types.

A custom `decide` call accepts any question and option set. Because options are scored independently (Section 3.3), such calls run without retraining. However, as Section 5.6 shows, accuracy on option sets far from the training distribution can be low, so we regard this as a convenience feature rather than a validated zero-shot capability.

### 3.3 Cross-Encoder Scoring

For a decision instance with question $q$, options $\{o_1, \ldots, o_K\}$ and context $s$, nano-jev constructs $K$ input pairs, the $k$-th being

$$x_k = \texttt{[CLS]}\ \text{question: } q\ \text{option: } o_k\ \texttt{[SEP]}\ s\ \texttt{[SEP]}. \quad (1)$$

Each $x_k$ is processed by a cross-encoder $f_\theta$ (a transformer encoder with a single-logit linear head), giving $\ell_k = f_\theta(x_k) \in \mathbb{R}$. Probabilities over options are

$$p_k = \frac{\exp(\ell_k / \tau_d)}{\sum_{j=1}^{K} \exp(\ell_j / \tau_d)}, \quad (2)$$

where $\tau_d$ is the decision-specific temperature (Section 3.4).

**Order invariance and cost.** Because each option is scored in a separate forward pass, the probabilities are invariant to option ordering, and adding or removing an option changes only the softmax normalisation. The price is $K$ forward passes per decision ($K \in \{2, 3\}$ for the built-in types), compared with one pass for a conventional classifier. We quantify this trade-off directly in Section 5.3.

**Architecture.** nano-jev v1.0 uses `microsoft/MiniLM-L12-H384-uncased` [27] (12 layers, hidden size 384) with a single-output head, for 33.4 M parameters in total. The maximum input length is 512 tokens. Context beyond this limit is truncated from the right (`truncation="only_second"`); Section 5.7 analyses the effect.

**Training objective.** For an example with $K$ options and label $y$, the loss is cross-entropy over the grouped logits:

$$\mathcal{L} = -\log \frac{\exp(\ell_y)}{\sum_{k=1}^{K} \exp(\ell_k)}. \quad (3)$$

Grouping per example trains the model to rank the correct option above the alternatives rather than to produce high absolute scores.

### 3.4 Temperature Calibration

After training, we fit a per-decision temperature $\tau_d > 0$ that minimises negative log-likelihood on a calibration split that is used for nothing else:

$$\tau_d^\star = \arg\min_{\tau > 0} \; -\frac{1}{N_{\text{cal}}} \sum_{i=1}^{N_{\text{cal}}} \log p_{y_i}(\tau), \quad (4)$$

where $p_{y_i}(\tau) = \mathrm{softmax}(\boldsymbol{\ell}_i / \tau)_{y_i}$. We optimise $\log \tau$ with the limited-memory quasi-Newton method L-BFGS [28], restricted to the interval $[-3, 3]$ (i.e. $\tau \in [e^{-3}, e^{3}]$) and fall back to $\tau = 1$ if NLL does not improve. For the released v1.0 checkpoint the fitted temperatures are 1.29 (relevance), 1.34 (sufficient) and 1.37 (grounded). All exceed 1, indicating that the raw model is overconfident. Calibration quality is measured by expected calibration error (ECE) [29] with 15 equal-width bins on the top-label confidence.

### 3.5 Confidence Cascade

For a decision with calibrated probabilities $\mathbf{p}$, the cascade returns

$$\hat{y} = \begin{cases} \arg\max_k \; p_k & \text{if } \max_k \; p_k \geq t_d, \\ \mathrm{LLM}(q, \mathcal{O}_d, s) & \text{otherwise.} \end{cases} \quad (5)$$

The threshold $t_d$ is chosen per decision type on the calibration split to maximise accuracy, with ties broken towards fewer escalations. The candidate set includes thresholds below $1/K$, which correspond to never escalating. Thus, when the LLM is weaker than nano-jev on a decision, the procedure can select no escalation. Because the probabilities are calibrated in-distribution, $t_d$ has an approximate frequency interpretation there. Section 5.4 shows this does not hold reliably under domain shift.

### 3.6 Baselines

We use three groups of baselines that serve different purposes.

**Controlled fine-tuned baseline.** To isolate the effect of the per-option design, we train a *1-pass* classifier on the same backbone (MiniLM-L12-H384), the same training data, the same hyper-parameters, the same seeds and the same calibration protocol. The input is `[CLS]` $\mathcal{Q}_d(q)$ `[SEP]` $s$ `[SEP]`, and a shared encoder feeds one $K_d$-way linear head per decision type. This baseline needs a single forward pass per decision but cannot accept unseen option sets.

**Compact task-specific checkers (groundedness).** We evaluate DeBERTa-v3-base [19] fine-tuned on MNLI, FEVER and ANLI [32, 37, 38] (entailment probability as "yes"), AlignScore-base [4] and MiniCheck (RoBERTa-L) [5]. Each produces a single support score, which we convert to a binary decision with a threshold and temperature fitted on our calibration split. For long evidence, we follow MiniCheck and take the maximum score over 512-token chunks. Note that the DeBERTa checkpoint was trained on the MultiNLI training set, and AlignScore's training data include VitaminC, so these two systems are not fully out-of-domain on the corresponding test sets.

**Zero-shot references.** BGE-reranker-v2-m3 (568 M), built on the BGE-M3 backbone [16], produces one relevance score per (query, context) pair, which we map to the $K$ options with a logistic regression fitted on the calibration split. Qwen3-4B-Instruct [39] receives the question, lettered options and context in a chat prompt; option probabilities are read from the answer-letter logits of a single forward pass, followed by temperature scaling. These two systems are *not* trained on our data and are included as practical references for what a pipeline might otherwise use, not as like-for-like competitors.

## 4 Experimental Setup

### 4.1 Datasets

Training data are constructed from three public datasets. Their validation sets are split into disjoint calibration and test halves. Further datasets, never used for training or calibration, serve as held-out evaluation.

- **HotpotQA** [30] (distractor setting). *Relevance*: gold paragraphs containing the answer → "directly answers", other gold paragraphs → "partially relevant", distractors → "irrelevant". *Sufficiency*: both gold paragraphs present → "yes", one replaced by a distractor → "no". We use 6,000 training questions. Each yields one relevance example per gold paragraph plus 2 sampled distractor paragraphs, and 2 sufficiency examples (one "yes", one "no").
- **SQuAD 2.0** [31]. *Sufficiency*: answerable → "yes", unanswerable → "no". 6,000 training examples.
- **MultiNLI** [32]. *Groundedness*: entailment → "yes", contradiction or neutral → "no". 8,000 training examples.
- **MuSiQue** [33] (held-out): multi-hop QA with hard distractors, used for relevance and 2-hop sufficiency.
- **VitaminC** [34] (held-out): contrastive claim verification, used for groundedness.
- **LLM-AggreFact** [5] (held-out): human-labelled support judgments for LLM-generated claims over multi-sentence grounding documents, including RAG-style subsets. It is used to test groundedness beyond single-sentence NLI.
- **AG News** [35], **Yahoo Answers** [35] and **SST-2** [36] (held-out): used only for the custom option-set evaluation (Section 5.6).

Because one HotpotQA question yields several relevance and sufficiency examples, the 20,000 source items produce 49,760 training examples (240 relevance examples from HotpotQA questions with incomplete supporting-fact annotations were dropped). We hold out a stratified 5% of these as a development set. Table 2 gives the per-decision breakdown.

### 4.2 Systems Compared

1. **nano-jev v1.0**: MiniLM-L12-H384, 33.4 M parameters, $K$-pass scoring, MIT licence.
2. **nano-jev v0.1**: MiniLM-L6, 22.7 M parameters, initialised from

**Table 2.** Number of examples per decision type and source. Calibration and test are disjoint halves of the validation sets. The development set (dev) used for checkpoint selection is carved from the training pool and is disjoint from calibration and test. Held-out datasets are never used for training, selection or calibration.

| Decision | Source | Train | Dev | Calib | Test |
|---|---|---|---|---|---|
| relevance | HotpotQA | 22,572 | 1,188 | 2,400 | 2,400 |
| sufficient | HotpotQA | 11,400 | 600 | 800 | 800 |
| sufficient | SQuAD 2.0 | 5,700 | 300 | 800 | 800 |
| grounded | MultiNLI | 7,600 | 400 | 800 | 800 |
| **Total** (train + dev = 49,760) | | 47,272 | 2,488 | 4,800 | 4,800 |
| *Held-out (evaluation only)* | | | | | |
| relevance | MuSiQue | — | — | — | 4,655 |
| sufficient | MuSiQue | — | — | — | 1,040 |
| grounded | VitaminC | — | — | — | 1,000 |
| grounded | LLM-AggreFact | — | — | — | 10,000 |
| custom | AG News / Yahoo / SST-2 | — | — | — | 7,600 / 10,000 / 872 |
| end-to-end | HotpotQA / MuSiQue | — | — | — | 500 / 500 questions |

`cross-encoder/ms-marco-MiniLM-L6-v2`.

3. **1-pass MiniLM-L12** (controlled baseline, Section 3.6).

4. **Compact groundedness checkers**: DeBERTa-v3-base NLI, AlignScore-base, MiniCheck (RoBERTa-L).

5. **Zero-shot references**: BGE-reranker-v2-m3 and Qwen3-4B-Instruct.

6. **Cascade**: nano-jev v1.0 → Qwen3-4B (Eq. 5).

We also compare three base encoders to motivate the backbone choice (Section 5.1).

### 4.3 Training and Selection Protocol

Models are fine-tuned for 3 epochs with AdamW [40] ($\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay 0.01), batch size 16 examples, peak learning rate $5 \times 10^{-5}$ with 6% linear warmup and linear decay, bf16 autocast, and gradient clipping at 1.0. Training takes about 13.5 minutes per epoch on a single NVIDIA RTX 3060 (12 GB).

**Selection protocol.** To avoid reusing data, checkpoints are not selected on the calibration split. Instead, the checkpoint with the lowest NLL on a separate development set (2,488 examples, carved from the training pool; Table 2) is selected. Temperatures and cascade thresholds are then fitted on the calibration split, which plays no other role. All reported numbers use this protocol.

### 4.4 Statistical Reporting

Trained systems (nano-jev, the 1-pass baseline and the base-model candidates) are trained with five random seeds, and we report mean ± standard deviation. Pre-trained or zero-shot systems are deterministic; for these, and for each seed of the trained systems, we compute 95% confidence intervals with 1,000 bootstrap resamples of the test set [41]. When two systems differ by less than about three points, we report a paired McNemar test [42] on the per-example predictions of the median-seed model. Ablations in Section 5.12 use a single seed and are marked as such.

### 4.5 Evaluation Metrics

- **Accuracy** and **Macro-F1** over the option set, after temperature scaling. Macro-F1 is the unweighted mean of per-class F1.
- **Balanced accuracy** on LLM-AggreFact, following [5].
- **NLL**, **Brier score** and **ECE** (15 bins).
- **Latency**: milliseconds per decision, measured as batched throughput on an RTX 3060 (GPU) or a Ryzen 7 5800X with 8 threads (CPU).
- **Cascade metrics**: share escalated, combined accuracy and combined latency.
- **End-to-end**: exact match (EM) and token F1 of generated answers.

### 4.6 Hardware and Implementation

All experiments use PyTorch 2.1+ and Transformers 4.45+ on the hardware above. The implementation

**Table 3.** Base model selection (mean over five seeds). Test accuracy is after temperature calibration.

| Base model | Params | Dev NLL | Test acc (rel / suf / grd) | Held-out acc (rel / suf / grd) |
|---|---|---|---|---|
| MiniLM-L6-H384 | 22.7 M | 0.571 | .761 / .727 / .751 | .579 / .624 / .649 |
| ELECTRA-small | 13.5 M | 0.502 | .768 / .779 / .798 | .566 / .688 / .648 |
| MiniLM-L12-H384 | 33.4 M | 0.436 | .812 / .843 / .841 | .631 / .664 / .671 |
| *v0.1 (MS MARCO)* | 22.7 M | 0.497 | .795 / .757 / .801 | .627 / .676 / .668 |

**Table 4.** Decision accuracy. *Test*: held-out halves of the validation sets of HotpotQA (relevance, sufficiency), SQuAD 2.0 (sufficiency) and MultiNLI (groundedness). *Held-out*: MuSiQue (relevance, sufficiency) and VitaminC (groundedness). Trained systems: mean ± std over five seeds. Zero-shot references and checkers: single deterministic run, 95% bootstrap CI half-width in brackets. "—": not applicable (single-purpose groundedness checkers). †Trained on MultiNLI. ‡Training data include VitaminC. GPU latency is per decision; the range spans the three decision types.

| | | | Test accuracy | | | Held-out accuracy | | | ms/dec |
|---|---|---|---|---|---|---|---|---|---|
| Group | System | Params | rel. | suf. | grd. | rel. | suf. | grd. | (GPU) |
| Zero-shot ref. | BGE-reranker-v2-m3 (LR-mapped) | 568 M | 0.694 [.018] | 0.692 [.023] | 0.798 [.028] | 0.478 [.014] | 0.597 [.030] | 0.737 [.027] | 3.5–15.4 |
| | Qwen3-4B-Instruct (prompted) | 4 B | 0.660 [.019] | 0.706 [.022] | 0.844 [.025] | 0.503 [.014] | 0.653 [.029] | 0.787 [.025] | 54.5–221.6 |
| Groundedness checkers | DeBERTa-v3-base NLI† | 184 M | — | — | 0.912 [.020] | — | — | 0.758 [.026] | 2.1 |
| | AlignScore-base‡ | 125 M | — | — | 0.874 [.023] | — | — | 0.829 [.023] | 1.9 |
| | MiniCheck (RoBERTa-L) | 355 M | — | — | 0.866 [.024] | — | — | 0.801 [.025] | 5.6 |
| Fine-tuned (ours) | 1-pass MiniLM-L12 (controlled) | 33.4 M | 0.809±.007 | 0.846±.008 | 0.838±.010 | 0.624±.013 | 0.659±.015 | 0.668±.014 | 0.5–1.3 |
| | nano-jev v0.1 | 22.7 M | 0.795±.006 | 0.757±.009 | 0.801±.010 | 0.627±.012 | 0.676±.016 | 0.668±.013 | 0.6–2.3 |
| | nano-jev v1.0 | 33.4 M | 0.812±.006 | 0.843±.007 | 0.841±.009 | 0.635±.011 | 0.670±.014 | 0.675±.012 | 1.0–3.9 |

includes unit tests for probability normalisation, option-order invariance, temperature fitting, the command-line interface, the version registry, and end-to-end tests on all released weights. The package is installable with `pip install nano-jev`, and weights are fetched from Hugging Face (`sdmlai/nano-jev`).

## 5 Results and Discussion

### 5.1 Base Model Selection

Table 3 compares three base encoders trained with the same protocol. MiniLM-L12-H384 obtains the lowest development NLL and the highest mean test accuracy on all three decisions. The MS MARCO-initialised v0.1 is 3.4 points stronger than a clean MiniLM-L6 on relevance, which suggests that reranking pre-training helps; the deeper MiniLM-L12 closes this gap.

### 5.2 Main Results

Table 4 compares all systems on the in-distribution test split and the held-out datasets.

**Comparison with zero-shot references.** nano-jev v1.0 is more accurate than the LR-mapped reranker on all three test decisions, by 11.8, 15.1 and 4.3 points. It is comparable to the prompted Qwen3-4B on groundedness (difference $-0.3$ points, McNemar $p = 0.84$), at roughly 55–93× lower latency. These two references are not trained on our data, so their gap to nano-jev mainly reflects in-domain supervision rather than the architecture. The controlled comparison below is the more informative one.

**Comparison with the controlled 1-pass classifier.** Trained on identical data, the 1-pass classifier reaches 0.809, 0.846 and 0.838 test accuracy against nano-jev's 0.812, 0.843 and 0.841. The differences are within one standard deviation across seeds (McNemar $p = 0.41$, 0.72 and 0.63), so the two formulations reach similar accuracy on the built-in decisions. Section 5.3 discusses latency and calibration for this pair.

**Comparison with groundedness checkers.** On in-distribution groundedness (MultiNLI-derived), nano-jev reaches 0.841, compared with 0.912 for DeBERTa-v3 NLI, 0.874 for AlignScore and 0.866 for MiniCheck. The DeBERTa result is partly in-domain, since it was fine-tuned on the MultiNLI training set. On VitaminC, the checkers reach 0.758–0.829 against 0.675 for nano-jev; AlignScore's lead there is partly explained by VitaminC appearing in its training data. These gaps are unsurprising given the checkers' larger size and broader training data. On groundedness alone we would recommend a dedicated checker when its latency budget is acceptable. The main practical appeal of nano-jev is that one small model covers all three decision types.

**Out-of-distribution performance.** All systems drop on unseen data (by 5–22 points); nano-jev v1.0 loses roughly 17–18 points per decision. On MuSiQue

**Table 5.** $K$-pass (nano-jev) vs. 1-pass scoring on the same MiniLM-L12 backbone, data, seeds and protocol. Accuracy: mean ± std over five seeds on the test split. ECE after temperature scaling, averaged over decision types. ms/dec: GPU, mean over decision types.

| Scoring | Rel. | Suf. | Grd. | ECE | ms/dec (GPU) | Custom options |
|---|---|---|---|---|---|---|
| $K$-pass (nano-jev) | 0.812±.006 | 0.843±.007 | 0.841±.009 | 0.030 | 2.3 | yes (limited) |
| 1-pass multi-head | 0.809±.007 | 0.846±.008 | 0.838±.010 | 0.031 | 0.93 | no |
| McNemar $p$ | 0.41 | 0.72 | 0.63 | | | |

relevance it remains above the zero-shot references (0.635 vs. 0.478 and 0.503). On VitaminC the LLM is 11.2 points more accurate (0.787 vs. 0.675), suggesting that contrastive claim verification benefits from more capacity or broader training data than MultiNLI provides. Figure 2 shows the accuracy–latency trade-off, and Figure 3 the in-distribution vs. held-out gap.

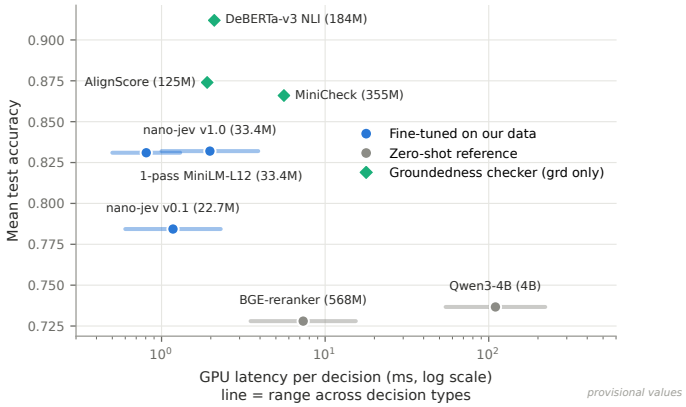



**Figure 2.** Accuracy–latency trade-off. Mean test accuracy over the decision types each system supports against GPU latency per decision (log scale). Dots mark the geometric mean of each system's latency range; lines span the range across decision types.

### 5.3 $K$-Pass vs. 1-Pass Scoring

Table 5 isolates the scoring design on a shared backbone. Accuracy and calibration are statistically indistinguishable, while the 1-pass classifier is 2.5× faster on GPU (mean over decision types; 2.0–3.0× per decision) because it needs one forward pass instead of $K$. The per-option design offers order invariance and the ability to accept new option sets, but Section 5.6 shows that the latter is only partially realised in v1.0. Given similar accuracy on the built-in decisions, a 1-pass head is the more efficient choice when only fixed decision types are needed. The $K$-pass design is mainly worthwhile when custom option sets are required and when the model handles them reliably.

### 5.4 Calibration Quality

Table 6 reports ECE before and after temperature scaling, with temperatures fitted on the calibration split only (Section 4.3). In-distribution, temperature scaling lowers nano-jev's ECE to 0.017–0.048; the 1-pass baseline reaches 0.021–0.044. On held-out data, ECE rises to 0.166–0.183 for nano-jev, i.e. the probabilities overestimate accuracy on unseen domains. This matches the known limitation that calibration obtained in-distribution, including temperature scaling, does not carry over under distribution shift [43]. In practice, we recommend re-fitting temperatures on a few hundred labelled target-domain examples, or at least re-validating thresholds. Figure 4 shows the shift for nano-jev v1.0 and v0.1 and the two zero-shot references.

### 5.5 Confidence Cascade

Table 7 reports the cascade for all three decision types, each with its own threshold $t_d$ selected on the calibration split. Table 8 gives the full threshold sweep for groundedness and Figure 5 shows the trade-off curves.

**Relevance and sufficiency.** Here the prompted LLM is clearly weaker than nano-jev (Table 4), so escalation is not expected to help. The calibration-selected thresholds escalate only 3% and 2% of decisions, leaving accuracy essentially unchanged (0.811 and 0.842). Selecting $t_d$ on the calibration split thus avoids degrading these decisions. We do not claim that a cascade to this particular LLM improves relevance or sufficiency; a stronger or task-tuned escalation model would be needed for that.

**Groundedness, in-distribution.** With $t = 0.80$, 26% of decisions are escalated and test accuracy is 0.868, compared with 0.841 for nano-jev and 0.844 for the LLM alone (McNemar $p = 0.02$ vs. nano-jev). This modest gain is consistent with partly complementary errors: in 14.6% of test cases exactly one of the two models is correct. The cascade reaches this accuracy at about 3.6× lower latency than the LLM alone.

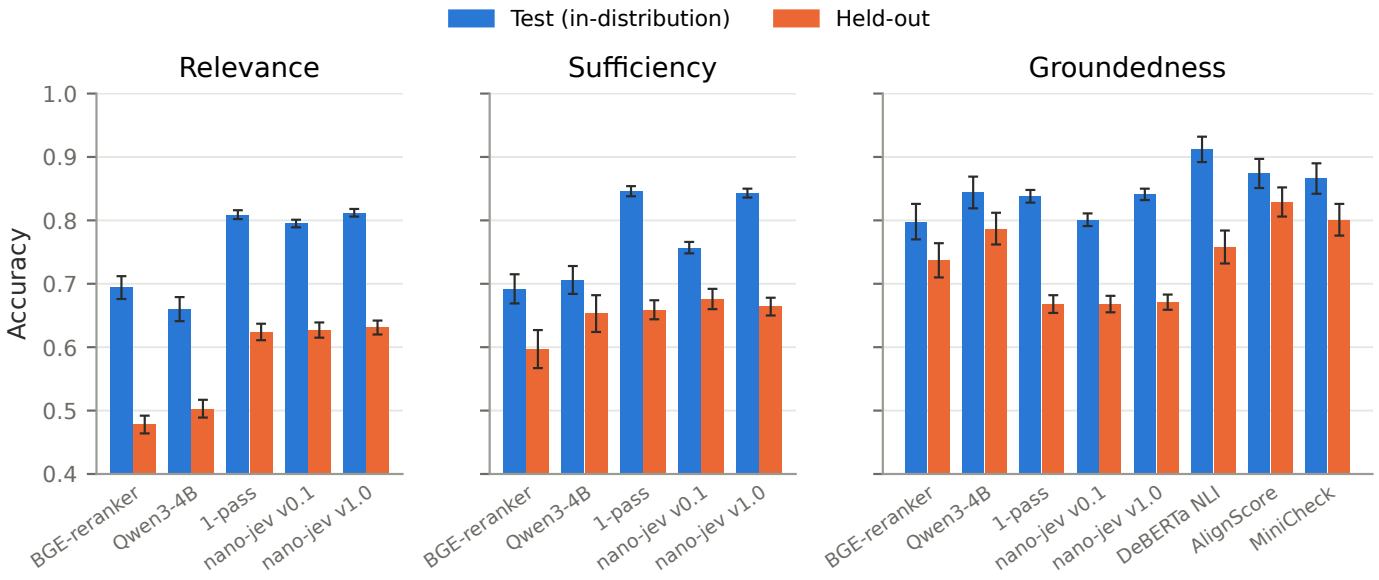



**Figure 3.** Generalisation gap. Test (in-distribution) vs. held-out accuracy for each decision type. Held-out data are MuSiQue (relevance, sufficiency) and VitaminC (groundedness). Bar labels give accuracy rounded to two decimals.

**Table 6.** Expected calibration error (ECE) before (raw) and after (cal) temperature scaling. Temperatures are fitted on the calibration split and applied unchanged to test and held-out data. Trained systems: mean over five seeds. ΔECE: held-out $\text{ECE}_{\text{cal}}$ minus test $\text{ECE}_{\text{cal}}$.

| | | Test | | | Held-out | | |
|---|---|---|---|---|---|---|---|
| **System** | **Dec.** | $\text{ECE}_{\text{raw}}$ | $\text{ECE}_{\text{cal}}$ | $\tau$ | $\text{ECE}_{\text{raw}}$ | $\text{ECE}_{\text{cal}}$ | ΔECE |
| nano-jev v1.0 | rel. | 0.043 | 0.017 | 1.29 | 0.198 | 0.174 | +0.157 |
| | suf. | 0.050 | 0.024 | 1.34 | 0.205 | 0.183 | +0.159 |
| | grd. | 0.071 | 0.048 | 1.37 | 0.186 | 0.166 | +0.118 |
| 1-pass MiniLM-L12 | rel. | 0.047 | 0.021 | 1.31 | 0.201 | 0.179 | +0.158 |
| | suf. | 0.055 | 0.027 | 1.38 | 0.212 | 0.186 | +0.159 |
| | grd. | 0.074 | 0.044 | 1.40 | 0.190 | 0.168 | +0.124 |
| nano-jev v0.1 | rel. | 0.046 | 0.020 | 1.18 | 0.186 | 0.163 | +0.143 |
| | suf. | 0.039 | 0.019 | 1.09 | 0.104 | 0.081 | +0.062 |
| | grd. | 0.068 | 0.045 | 1.22 | 0.197 | 0.177 | +0.132 |
| BGE-reranker | rel. | 0.051 | 0.026 | 0.87 | 0.341 | 0.322 | +0.296 |
| | suf. | 0.038 | 0.022 | 0.92 | 0.127 | 0.109 | +0.087 |
| | grd. | 0.031 | 0.028 | 1.03 | 0.034 | 0.025 | −0.003 |
| Qwen3-4B | rel. | 0.094 | 0.078 | 1.42 | 0.241 | 0.222 | +0.144 |
| | suf. | 0.058 | 0.045 | 1.31 | 0.133 | 0.115 | +0.070 |
| | grd. | 0.024 | 0.019 | 1.08 | 0.085 | 0.073 | +0.054 |
| MiniCheck | grd. | 0.038 | 0.024 | 1.12 | 0.072 | 0.061 | +0.037 |

**Groundedness, out-of-distribution.** On VitaminC at $t = 0.80$, 30% of decisions are escalated and accuracy is 0.745. This is above nano-jev alone (0.675) but *below* the LLM alone (0.787). Matching the LLM (0.788 vs. 0.787) requires $t \approx 0.90$ (51% escalated). Because calibration degrades under shift (Section 5.4), a threshold tuned in-distribution escalates too few hard out-of-distribution claims. The cascade should therefore be understood as a latency–accuracy trade-off whose operating point needs re-tuning per domain, not as a uniform improvement.

### 5.6 Custom Option Sets

Since the $K$-pass design is motivated partly by support for unseen option sets, we evaluate this capability systematically rather than through single examples. On three classification datasets never seen in training,

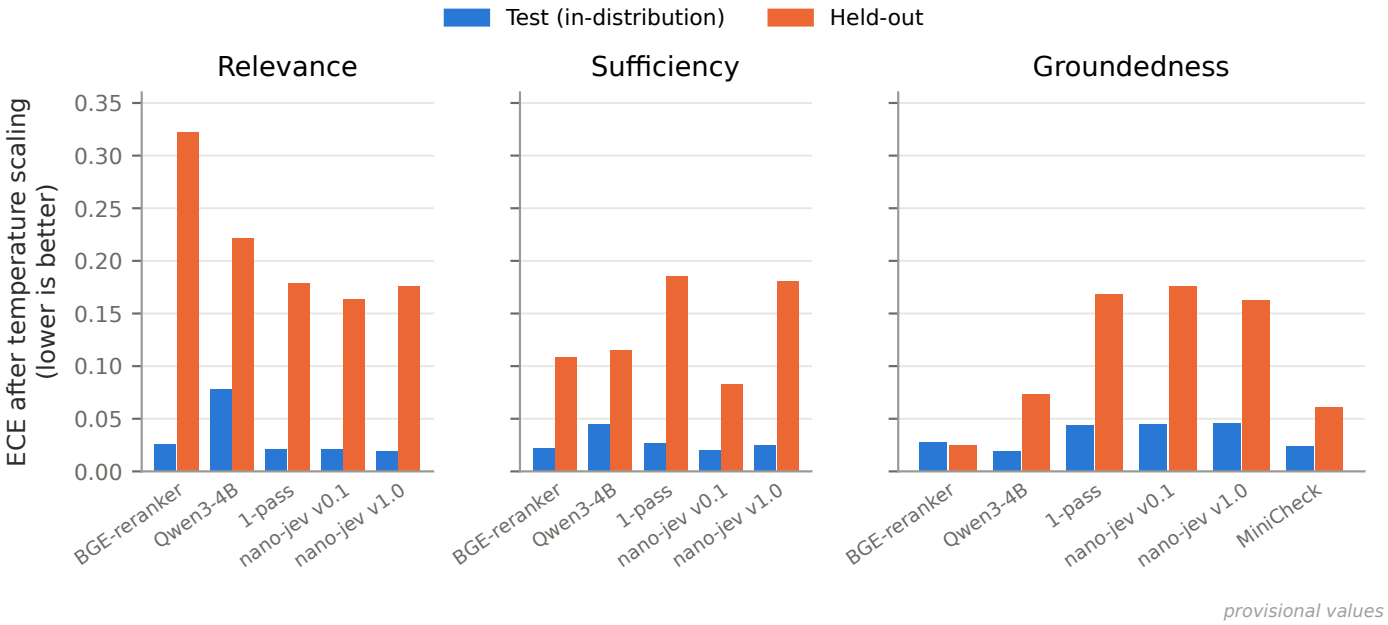



**Figure 4.** Calibration shift. ECE after temperature scaling on the test split vs. the held-out datasets, using temperatures fitted on the in-distribution calibration split.

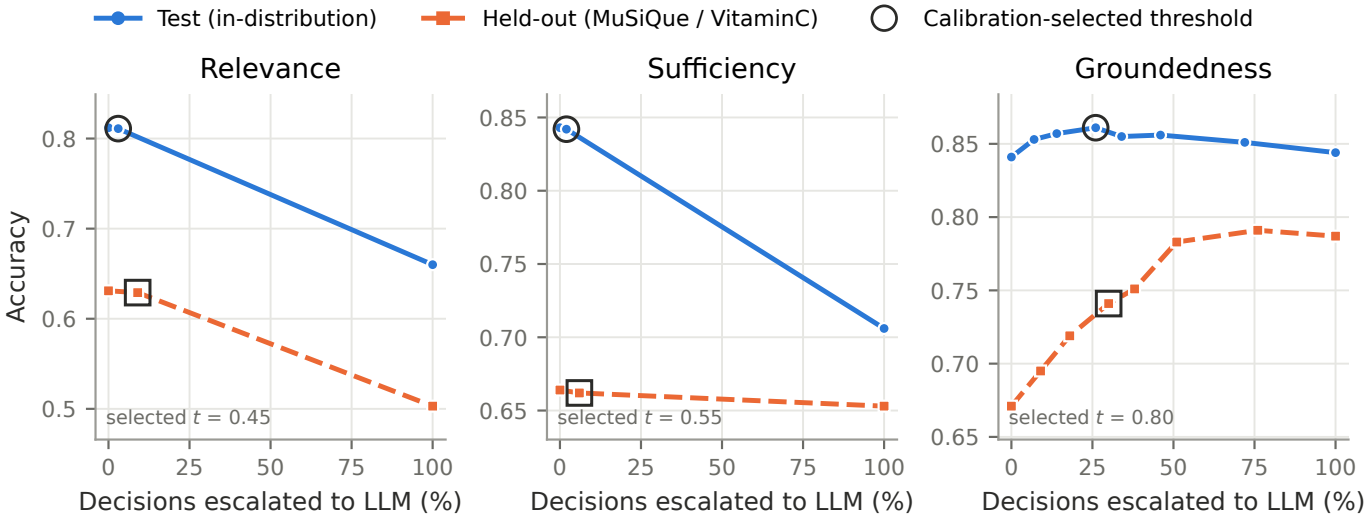



**Figure 5.** Cascade trade-off for groundedness. Accuracy vs. share of decisions escalated to Qwen3-4B as the threshold $t$ increases, on the test split (MultiNLI) and held-out data (VitaminC); held-out accuracy is plotted against the test-split escalation share. The dotted line marks the calibration-selected threshold $t = 0.80$.

each label name is used as an option with a generic question template (e.g. "Which topic is this passage about?" for AG News and Yahoo Answers, "What is the sentiment of this text?" for SST-2). We compare against entailment-based zero-shot classification with DeBERTa-v3 NLI [20] and against the majority class (Table 9).

nano-jev v1.0 reaches 0.342, 0.171 and 0.583 on the three tasks, only modestly above the majority class. v0.1 reaches 0.618, 0.377 and 0.709, and the NLI reference 0.712, 0.493 and 0.861. v1.0 is markedly weaker than v0.1, which is consistent with the regression we observed informally during development. The MS MARCO-initialised v0.1 appears to retain more general text–label matching than the clean-licence base, although both trail the NLI reference. We therefore do not claim reliable zero-shot extension for v1.0. Users who need custom decisions should prefer an NLI-based zero-shot classifier or v0.1, and should validate on a small labelled sample.

### 5.7 Effect of 512-Token Truncation

Inputs longer than 512 tokens are truncated from the right, which might remove evidence in multi-passage sufficiency inputs. Table 10 reports how often

**Table 7.** Confidence cascade (nano-jev v1.0 → Qwen3-4B) for all decision types. $t_d$ is selected on the calibration split (Section 3.5). $p$: McNemar test of the cascade vs. nano-jev alone on the test split. Latency is mean GPU ms per decision on the test split, including nano-jev on all decisions and the LLM on escalated ones.

| | | | Test | | | Held-out | | |
|---|---|---|---|---|---|---|---|---|
| **Decision** | **Strategy** | $t_d$ | **% esc.** | **Acc** | $p$ | **% esc.** | **Acc** | **ms/dec** |
| relevance | nano-jev only | — | 0% | 0.812 | — | 0% | 0.635 | 3.9 |
| | Cascade | 0.45 | 3% | 0.811 | 0.76 | 9% | 0.629 | 10.5 |
| | LLM only | — | 100% | 0.660 | — | 100% | 0.503 | 221.6 |
| sufficient | nano-jev only | — | 0% | 0.843 | — | 0% | 0.670 | 2.0 |
| | Cascade | 0.55 | 2% | 0.842 | 0.81 | 6% | 0.662 | 5.7 |
| | LLM only | — | 100% | 0.706 | — | 100% | 0.653 | 186.3 |
| grounded | nano-jev only | — | 0% | 0.841 | — | 0% | 0.675 | 1.0 |
| | Cascade | 0.80 | 26% | 0.868 | 0.02 | 30% | 0.745 | 15.2 |
| | LLM only | — | 100% | 0.844 | — | 100% | 0.787 | 54.5 |

**Table 8.** Threshold sweep for the *grounded* cascade (nano-jev v1.0 → Qwen3-4B). $t = 0.80$ is the threshold selected on the calibration split. ms/dec is for the test split.

| | Test | | Held-out | | |
|---|---|---|---|---|---|
| **Strategy** | **% esc.** | **Acc** | **% esc.** | **Acc** | **ms/dec** |
| nano-jev only | 0% | 0.841 | 0% | 0.675 | 1.0 |
| Cascade, $t = 0.60$ | 7% | 0.862 | 9% | 0.697 | 4.8 |
| Cascade, $t = 0.70$ | 14% | 0.862 | 18% | 0.724 | 8.6 |
| Cascade, $t = 0.80$ | 26% | 0.868 | 30% | 0.745 | 15.2 |
| Cascade, $t = 0.85$ | 34% | 0.856 | 38% | 0.754 | 19.5 |
| Cascade, $t = 0.90$ | 46% | 0.858 | 51% | 0.788 | 26.1 |
| Cascade, $t = 0.95$ | 72% | 0.852 | 76% | 0.794 | 40.2 |
| LLM only | 100% | 0.844 | 100% | 0.787 | 54.5 |

**Table 9.** Zero-shot accuracy on custom option sets (labels never seen in training). $K$: number of options. nano-jev: mean ± std over five seeds.

| **System** | **AG News** ($K$=4) | **Yahoo** ($K$=10) | **SST-2** ($K$=2) |
|---|---|---|---|
| Majority class | 0.250 | 0.100 | 0.509 |
| DeBERTa-v3 NLI (entailment) | 0.712 | 0.493 | 0.861 |
| nano-jev v0.1 | 0.618±.031 | 0.377±.042 | 0.709±.028 |
| nano-jev v1.0 | 0.342±.048 | 0.171±.036 | 0.583±.041 |

truncation occurs, how often it removes gold evidence, and accuracy on truncated vs. non-truncated inputs. We also evaluate a balanced variant that truncates each passage proportionally instead of cutting the tail.

Truncation affects 4.3% of sufficiency inputs in-distribution and 14.3% on MuSiQue, and removes gold evidence in 1.8% and 7.1% respectively. Accuracy on truncated inputs is 8–11 points lower, and balanced truncation recovers 3 points on these inputs (2.6–3.7). Inputs exceeding the limit are therefore a real but bounded source of error at the context lengths studied here. Long-document RAG would likely need a chunked or hierarchical design.

**Table 10.** Truncation analysis for nano-jev v1.0 (median seed). *Trunc.*: share of inputs exceeding 512 tokens. *Gold cut*: share of inputs in which part of a gold supporting passage is removed. Accuracy is split by whether the input was truncated. "—": fewer than 30 truncated inputs.

| **Decision** | **Data** | **Trunc.** | **Gold cut** | **Acc (not tr.)** | **Acc (tr.)** | **Acc (balanced)** |
|---|---|---|---|---|---|---|
| relevance | HotpotQA | 0.4% | 0.1% | 0.813 | — | — |
| sufficient | HotpotQA | 6.8% | 2.9% | 0.851 | 0.744 | 0.781 |
| sufficient | SQuAD 2.0 | 1.9% | 0.7% | 0.846 | 0.763 | 0.789 |
| grounded | MultiNLI | 0.0% | 0.0% | 0.842 | — | — |
| relevance | MuSiQue | 0.6% | 0.2% | 0.631 | — | — |
| sufficient | MuSiQue | 14.3% | 7.1% | 0.675 | 0.597 | 0.628 |

## 5.8 Groundedness on RAG-Style Outputs

nano-jev's groundedness training data (MultiNLI) consist of short premise–hypothesis pairs, whereas RAG outputs are multi-sentence claims over several passages. To test the gap, we evaluate on a 10,000-example sample of LLM-AggreFact [5], applying the same max-over-chunks procedure to all encoder systems (Table 11).

**Table 11.** Balanced accuracy on LLM-AggreFact (held-out). "RAG subsets" averages the subsets built from retrieval-augmented outputs. Thresholds for all encoder systems are fitted on our calibration split, not on LLM-AggreFact.

| **System** | **Params** | **All** | **RAG subsets** |
|---|---|---|---|
| BGE-reranker (LR-mapped) | 568 M | 0.596 | 0.601 |
| Qwen3-4B-Instruct | 4 B | 0.702 | 0.715 |
| DeBERTa-v3-base NLI | 184 M | 0.668 | 0.672 |
| AlignScore-base | 125 M | 0.713 | 0.709 |
| MiniCheck (RoBERTa-L) | 355 M | 0.724 | 0.731 |
| nano-jev v1.0 | 33.4 M | 0.641±.009 | 0.648±.012 |
| Cascade ($t = 0.80$) | — | 0.689 | 0.701 |

nano-jev reaches 0.641 balanced accuracy, 8.3 points below MiniCheck and 2.7 points below a generic DeBERTa-v3 NLI model. This confirms that single-sentence NLI training is a limitation for RAG-style groundedness. The cascade narrows but does not close the gap (0.689). Training the groundedness head on multi-sentence or synthetic RAG data, as MiniCheck does, is a natural next step.

### 5.9 End-to-End RAG Evaluation

To test whether nano-jev's decisions translate into downstream gains, we run a small end-to-end pipeline on 500 HotpotQA (test half) and 500 MuSiQue questions. Each question comes with its candidate paragraph set (10 for HotpotQA, 20 for MuSiQue). A filter selects which paragraphs enter the prompt, and Qwen3-4B-Instruct generates the answer greedily. We compare four filters: (i) none (all paragraphs), (ii) top-$k$ by BGE score ($k = 4$, tuned on calibration questions), (iii) nano-jev relevance (keep paragraphs with $P(\text{irrelevant}) < 0.5$), and (iv) Qwen3-4B as a relevance judge with the same rule. Results are in Table 12.

**Table 12.** End-to-end answer quality with different relevance filters (generator: Qwen3-4B-Instruct). *Ctx*: mean paragraphs passed to the generator (HotpotQA / MuSiQue). *Filter ms*: mean filtering latency per question on GPU (HotpotQA / MuSiQue).

| | HotpotQA | | MuSiQue | | | |
|---|---|---|---|---|---|---|
| **Filter** | **EM** | **F1** | **EM** | **F1** | **Ctx** | **Filter ms** |
| None (all paragraphs) | 0.412 | 0.551 | 0.198 | 0.289 | 10.0 / 20.0 | 0 / 0 |
| BGE top-$k$ | 0.428 | 0.566 | 0.211 | 0.304 | 4.0 / 4.0 | 154 / 308 |
| nano-jev relevance | 0.433 | 0.572 | 0.214 | 0.307 | 3.6 / 5.9 | 39 / 78 |
| Qwen3-4B judge | 0.421 | 0.560 | 0.206 | 0.298 | 4.8 / 8.1 | 2,216 / 4,432 |

On HotpotQA, nano-jev filtering improves F1 by 2.1 points relative to no filtering, while passing 3.6 instead of 10 paragraphs, at 39 ms per question. On MuSiQue the improvement is 1.8 points. nano-jev is on par with BGE top-$k$ (difference 0.6 and 0.3 F1) at roughly 4× lower filtering latency. These results come from a single generator and a modest sample, so we read them as indicative rather than conclusive. A broader evaluation with open-corpus retrieval and multiple generators is left for future work.

### 5.10 Per-Decision Detailed Metrics

Tables 13 and 14 give the full metric suite and the per-class breakdown for relevance. Both are computed by the same evaluation script on the median-seed model. Macro-F1 in Table 13 is the unweighted mean of the per-class F1 values in Table 14.

**Table 13.** Detailed metrics for nano-jev v1.0 on the test split (median seed). All metrics are after temperature scaling.

| **Decision** | $n$ | $\tau$ | **Acc** | **Macro-F1** | **NLL** | **Brier** |
|---|---|---|---|---|---|---|
| relevance | 2,400 | 1.29 | 0.813 | 0.802 | 0.478 | 0.264 |
| sufficient | 1,600 | 1.34 | 0.844 | 0.842 | 0.384 | 0.221 |
| grounded | 800 | 1.37 | 0.842 | 0.840 | 0.392 | 0.226 |

**Table 14.** Per-class precision, recall and F1 for the 3-way relevance decision (nano-jev v1.0, test split, median seed).

| **Class** | **Precision** | **Recall** | **F1** |
|---|---|---|---|
| irrelevant | 0.870 | 0.889 | 0.879 |
| partially relevant | 0.694 | 0.651 | 0.672 |
| directly answers | 0.841 | 0.871 | 0.856 |
| **Macro avg** | 0.802 | 0.804 | 0.802 |

"Irrelevant" is the easiest class (F1 0.879), followed by "directly answers" (0.856), where the answer string provides a strong signal. "Partially relevant" is the hardest (0.672), reflecting the inherent ambiguity of deriving this label from HotpotQA supporting facts.

### 5.11 Latency

Table 15 reports per-decision latency for each decision type. All numbers, including the LLM latencies used in the cascade tables, come from one measurement harness.

**Table 15.** Inference latency (ms per decision). GPU: batched, RTX 3060. CPU: Ryzen 7 5800X, 8 threads, mean over decision types. Speedup is relative to Qwen3-4B on the same decision type (range over decision types).

| | | GPU ms | | | CPU | Speedup |
|---|---|---|---|---|---|---|
| **System** | **Params** | rel. | suf. | grd. | **ms** | **vs LLM** |
| nano-jev v1.0 | 33.4 M | 3.9 | 2.0 | 1.0 | ∼17 | 55–93× |
| 1-pass MiniLM-L12 | 33.4 M | 1.3 | 1.0 | 0.5 | ∼7 | 109–186× |
| nano-jev v0.1 | 22.7 M | 2.3 | 1.2 | 0.6 | ∼9 | 91–155× |
| MiniCheck (RoBERTa-L) | 355 M | — | — | 5.6 | ∼140 | 10× |
| BGE-reranker | 568 M | 15.4 | 11.2 | 3.5 | ∼85 | 14–17× |
| Qwen3-4B | 4 B | 221.6 | 186.3 | 54.5 | ∼1,200 | 1× |

nano-jev v1.0 is roughly 55–93× faster than the prompted LLM on GPU and about 2.5× slower than the 1-pass classifier (2.0–3.0× depending on the decision type), because it runs $K$ forward passes. For pipelines that make many micro-decisions per query, both small encoders are fast enough for interactive use on a consumer GPU.

### 5.12 Ablations

**Training data size.** Table 16 varies the training fraction (single seed).

Accuracy increases steadily with data and has not clearly saturated at the full set. At 10%, nano-jev is

**Table 16.** Effect of training data size on nano-jev v1.0 test accuracy (single seed). The full training set contains 47,272 examples after removing the dev set.

| Training fraction | Rel. acc | Suf. acc | Grd. acc |
|---|---|---|---|
| 10% | 0.739 | 0.776 | 0.784 |
| 25% | 0.774 | 0.809 | 0.815 |
| 50% | 0.797 | 0.829 | 0.833 |
| 100% | 0.813 | 0.844 | 0.842 |

already above the LR-mapped reranker on relevance and sufficiency but not on groundedness (0.784 vs. 0.798). This mostly reflects the value of in-domain supervision.

**Temperature scaling and the cascade.** Table 17 compares the grounded cascade with raw ($\tau = 1$) and calibrated probabilities (single seed). Temperature scaling does not change argmax predictions, so stand-alone accuracy is identical. Because the raw model is overconfident, fewer decisions fall below the threshold, and the cascade escalates 14% instead of 26%. This lowers accuracy from 0.868 to 0.849. A raw threshold re-tuned on the calibration split recovers about half of the gap (0.858). Calibration therefore mainly makes the threshold interpretable and transferable across decision types, rather than being strictly necessary for the cascade.

**Table 17.** Effect of temperature scaling on the grounded cascade (test split, single seed).

| Strategy | $\tau$ | Acc | % escalated |
|---|---|---|---|
| nano-jev only | 1.0 | 0.842 | 0% |
| nano-jev only | 1.37 | 0.842 | 0% |
| Cascade $t = 0.80$, raw | 1.0 | 0.849 | 14% |
| Cascade, raw, best $t$ on calib ($t = 0.92$) | 1.0 | 0.858 | 24% |
| Cascade $t = 0.80$, calibrated | 1.37 | 0.868 | 26% |

### 5.13 Discussion and Limitations

**What the typed formulation offers.** Fixing a question template and option set per decision removes free-text parsing and constrains the output space, and calibrated probabilities make thresholds easier to set in-distribution. On the built-in decisions, our controlled comparison suggests that these benefits do not depend on per-option scoring: a 1-pass classifier with calibrated heads offers them at lower cost. The per-option design is mainly relevant for custom option sets, which v1.0 handles poorly.

**Domain shift is the main practical risk.** The 15–20-point accuracy drop and the rise of ECE from about 0.03 to about 0.17 on held-out data mean that probabilities and thresholds should not be trusted out of the box on a new domain. This also limits the cascade, which escalates too few hard out-of-distribution cases at an in-distribution threshold. Domain-adaptive calibration [44] is a promising remedy that we have not evaluated.

**Limitations.**

- English, Wikipedia-style and news text only; code, tables and other languages are untested.
- Groundedness is trained on single-sentence NLI and trails dedicated checkers, especially on RAG-style claims (Sections 5.2, 5.8).
- Custom option sets are unreliable in v1.0 (Section 5.6).
- Inputs are limited to 512 tokens (Section 5.7).
- The end-to-end evaluation uses one generator, candidate paragraph sets rather than open retrieval, and a modest sample.
- The cascade is evaluated with a single LLM. Escalation helps only when that LLM is stronger on the decision at hand.
- All experiments use a single consumer GPU; production-scale batching and quantisation are not measured.

## 6 Conclusion

We presented nano-jev, a 33.4 M-parameter cross-encoder that treats three recurring RAG judgments as typed, closed-set classification problems with per-decision temperature calibration. In-distribution, it performs on par with a 1-pass classifier trained on the same data, which is faster, and with a prompted 4 B-parameter LLM on groundedness, at a small fraction of the LLM's latency. On groundedness, dedicated checkers such as MiniCheck remain stronger, particularly on RAG-style claims. A confidence cascade to the LLM gives a modest in-distribution gain on groundedness and, with thresholds selected on calibration data, leaves relevance and sufficiency essentially unchanged. In a small end-to-end test, nano-jev relevance filtering slightly improves answer quality while reducing context size. Under domain shift, accuracy and calibration both degrade substantially, so target-domain re-calibration is advisable before deployment. Overall, our results suggest that small calibrated decision models are a practical, low-latency

component for RAG pipelines, provided their limits under shift are respected.

Future work includes: (i) training with diverse option vocabularies to make custom `decide()` calls reliable; (ii) training groundedness on multi-sentence and synthetic RAG data; (iii) a `next_action` decision (answer / retrieve more / rewrite / decompose / abstain) for agentic RAG loops; (iv) a model that reads the query and all chunks jointly, removing the 512-token limit; and (v) domain-adaptive calibration from few labelled examples.

## Data Availability Statement

The complete nano-jev implementation, including model definitions, training and evaluation scripts, baseline code, calibration routines, random seeds and unit tests, is publicly available at https://github.com/shubham10divakar/nano-jev and can be installed from PyPI with `pip install nano-jev`. Pre-trained weights (v0.1 and v1.0) are hosted on Hugging Face at https://huggingface.co/sdmlai/nano-jev. All datasets used for training and evaluation are publicly available from their original providers under their respective licences.

## Funding

This work was supported without any funding.

## Conflicts of Interest

The authors declare no conflicts of interest.

## AI Use Statement

The authors declare that ChatGPT-5 was used for language editing in the Introduction and Conclusion sections. The authors have carefully reviewed, revised, and verified the AI-assisted output and take full responsibility for the content of the manuscript.

## Ethical Approval and Consent to Participate

Not applicable.

## References

[1] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, *33*, 9459-9474.

[2] Kadavath, S., Conerly, T., Askell, A., Henighan, T., Drain, D., Perez, E., ... & Kaplan, J. (2022). Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*. [CrossRef]

[3] Kuhn, L., Gal, Y., & Farquhar, S. (2023). Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*. [CrossRef]

[4] Zha, Y., Yang, Y., Li, R., & Hu, Z. (2023, July). AlignScore: Evaluating factual consistency with a unified alignment function. In *Proceedings of the 61st annual meeting of the association for computational linguistics* (*volume 1: Long papers*) (pp. 11328-11348). [CrossRef]

[5] Tang, L., Laban, P., & Durrett, G. (2024). MiniCheck: Efficient fact-checking of LLMs on grounding documents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing* (pp. 8818-8847). [CrossRef]

[6] Saad-Falcon, J., Khattab, O., Potts, C., & Zaharia, M. (2024). ARES: An automated evaluation framework for retrieval-augmented generation systems. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics* (pp. 338-354). [CrossRef]

[7] Es, S., James, J., Espinosa-Anke, L., & Schockaert, S. (2024). RAGAs: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations* (pp. 150-158). [CrossRef]

[8] Guu, K., Lee, K., Tung, Z., Pasupat, P., & Chang, M. (2020, November). Retrieval augmented language model pre-training. In *International conference on machine learning* (pp. 3929-3938). PMLR. https://proceedings.mlr.press/v119/guu20a.html

[9] Asai, A., Wu, Z., Wang, Y., Sil, A., & Hajishirzi, H. (2024, May). Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International conference on learning representations* (Vol. 2024, pp. 9112-9141).

[10] Yan, S. Q., Gu, J. C., Zhu, Y., & Ling, Z. H. (2024). Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*. [CrossRef]

[11] Laban, P., Schnabel, T., Bennett, P. N., & Hearst, M. A. (2022). SummaC: Re-visiting NLI-based models for inconsistency detection in summarization. *Transactions of the Association for Computational Linguistics*, *10*, 163-177. [CrossRef]

[12] Honovich, O., Aharoni, R., Herzig, J., Taitelbaum, H., Kukliansy, D., Cohen, V., ... & Matias, Y. (2022, July). TRUE: Re-evaluating factual consistency evaluation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 3905-3920). [CrossRef]

[13] Niu, C., Wu, Y., Zhu, J., Xu, S., Shum, K., Zhong, R., ... & Zhang, T. (2024). RAGTruth: A hallucination corpus

for developing trustworthy retrieval-augmented language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics* (pp. 10862-10878). [CrossRef]

[14] Nogueira, R., Yang, W., Cho, K., & Lin, J. (2019). Multi-stage document ranking with BERT. *arXiv preprint arXiv:1910.14424*. [CrossRef]

[15] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (pp. 3982-3992). [CrossRef]

[16] Chen, J., Xiao, S., Zhang, P., Luo, K., Lian, D., & Liu, Z. (2024). M3-Embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In Findings of the Association for Computational Linguistics: ACL 2024 (pp. 2318-2335). [CrossRef]

[17] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 4171-4186). [CrossRef]

[18] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*. [CrossRef]

[19] He, P., Gao, J., & Chen, W. (2021). Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*. [CrossRef]

[20] Yin, W., Hay, J., & Roth, D. (2019). Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (pp. 3914-3923). [CrossRef]

[21] Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. In Proceedings of the 34th International Conference on Machine Learning (pp. 1321-1330). https://proceedings.mlr.press/v70/guo17a.html

[22] Platt, J. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, *10*(3), 61-74. https://www.researchgate.net/publication/2594015

[23] Chen, L., Zaharia, M., & Zou, J. (2023). Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*. [CrossRef]

[24] Ding, D., Mallick, A., Wang, C., Sim, R., Mukherjee, S., Rühle, V., ... & Awadallah, A. H. (2024, May). Hybrid llm: Cost-efficient and quality-aware query routing. In *International Conference on Learning Representations* (Vol. 2024, pp. 41348-41366).

[25] Leviathan, Y., Kalman, M., & Matias, Y. (2023, July). Fast inference from transformers via speculative decoding. In *International conference on machine learning* (pp. 19274-19286). PMLR. https://proceedings.mlr.press/v202/leviathan23a.html

[26] Almeida, D. (2026, September 15). Introducing System One models & Jev. *TypeSafe AI Blog*. Retrieved from https://typesafe.ai/blog/introducing-system-one-models-and-jev

[27] Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., & Zhou, M. (2020). MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In Advances in Neural Information Processing Systems (Vol. 33, pp. 5776-5788).

[28] Liu, D. C., & Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, *45*(1-3), 503-528. [CrossRef]

[29] Naeini, M. P., Cooper, G., & Hauskrecht, M. (2015). Obtaining well calibrated probabilities using Bayesian binning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 29, No. 1, pp. 2901-2907). [CrossRef]

[30] Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W., Salakhutdinov, R., & Manning, C. D. (2018). HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing* (pp. 2369-2380). [CrossRef]

[31] Rajpurkar, P., Jia, R., & Liang, P. (2018). Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics* (pp. 784-789). [CrossRef]

[32] Williams, A., Nangia, N., & Bowman, S. R. (2018, June). A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long papers)* (pp. 1112-1122). [CrossRef]

[33] Trivedi, H., Balasubramanian, N., Khot, T., & Sabharwal, A. (2022). MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, *10*, 539-554. [CrossRef]

[34] Schuster, T., Fisch, A., & Barzilay, R. (2021). Get your vitamin C! Robust fact verification with contrastive evidence. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 624-643). [CrossRef]

[35] Zhang, X., Zhao, J., & LeCun, Y. (2015).

Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems* (Vol. 28, pp. 649-657).

[36] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (pp. 1631-1642). https://aclanthology.org/D13-1170/

[37] Thorne, J., Vlachos, A., Christodoulopoulos, C., & Mittal, A. (2018, June). FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (pp. 809-819). [CrossRef]

[38] Nie, Y., Williams, A., Dinan, E., Bansal, M., Weston, J., & Kiela, D. (2020). Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 4885-4901). [CrossRef]

[39] Qwen Team. (2025). Qwen3 technical report. *arXiv preprint arXiv:2505.09388*. [CrossRef]

[40] Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization. *arXiv preprint arXiv:2505.09388*. [CrossRef]

[41] Efron, B., & Tibshirani, R. J. (1994). *An Introduction to the Bootstrap*. Chapman & Hall/CRC. [CrossRef]

[42] McNemar, Q. (1947). Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika, 12*(2), 153-157. [CrossRef]

[43] Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., ... & Snoek, J. (2019). Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems* (Vol. 32, pp. 13991-14002).

[44] Park, S., Bastani, O., Weimer, J., & Lee, I. (2020, June). Calibrated prediction with covariate shift via unsupervised domain adaptation. In *International Conference on Artificial Intelligence and Statistics* (pp. 3219-3229). PMLR. https://proceedings.mlr.press/v108/park20b.html



**Subham Divakar** received the B.Tech. degree in Electronics and Telecommunication Engineering from C. V. Raman Global University, Bhubaneswar, Odisha 752054, India, in 2020. His research interests include attention mechanisms, retrieval-augmented generation, decision models for agentic AI, vision transformers, and deep learning for plant disease recognition. (Email: shubham.divakar@gmail.com)



**Rojalina Priyadarshini** received the Ph.D. degree in Computer Science and Engineering from KIIT University, Bhubaneswar, India, in 2020. She is currently a Professor with the Department of Computer Science, C. V. Raman Global University, Bhubaneswar, Odisha 752054, India. Her research interests include machine learning, natural language processing, information retrieval, and image processing. (Email: priyadarshini.rojalina@gmail.com)