SORIKA AI LABS · FOUNDATION INTELLIGENCE RESEARCH

Swen 1.1 Technical Report: Efficient Edge Intelligence via Hybrid Double-Gated Convolutions and Test-Time Metacognitive Reasoning

Sorika Labs Technical Report — Series 1.1
September 2026

Darsh Yadav
Founder & Chief AI Architect, Sorika Labs
with the Sorika Labs Research & Systems Architecture Team

Correspondence: research@sorika.ai | darsh@sorika.ai
Organization: Sorika Labs

Abstract — Small language models (SLMs, $\leq 2B$ parameters) are pivotal for privacy-preserving, low-latency, on-device artificial intelligence. However, standard causal Transformers suffer from two acute structural handicaps: quadratic sequence prefill complexity and linearly expanding key-value (KV) memory footprints during autoregressive decoding. In this paper, we introduce the Swen 1.1 Model Family, a suite of three ultra-efficient foundation and specialized models developed by Sorika Labs:

1. **Swen-1.1-Instruct (1.2B)**: A general-purpose compact assistant optimized for instruction following, complex multilingual dialogue across 8 languages, agentic tool execution, and code synthesis.
2. **Swen-1-Math (350M)**: An ultra-compact mathematical specialist leveraging an autonomous Chain-of-Thought (CoT) scratchpad to deliver 1B-class arithmetic and algebraic deduction.
3. **Swen-1.1-Thinking (1.2B)**: A metacognitive reasoning model designed for self-reflective, backtrack-capable deduction under constrained entropy sampling.

The Swen 1.1 series is anchored by a hybrid backbone interleaving ten double-gated causal convolution layers with six grouped-query attention (GQA) layers, achieving an effective context window of 128,000 tokens while maintaining constant $O(K-1)$ cache states across the majority of the network depth. Pre-trained on a curated corpus of 28 Trillion tokens and aligned via multi-stage masked Supervised Fine-Tuning (SFT) and Reinforcement Learning with Verifiable Rewards (RLVR), Swen 1.1 establishes new Pareto frontiers for sub-2B parameter models:

- Swen-1.1-Instruct scores 66.0% on HumanEval (outperforming SmolLM2-1.7B at 22.6% and Llama-3.2-1B at 25.0% by nearly 3×), 54.0% on GSM8K, and 38.0% on GPQA Diamond.
- Swen-1-Math scores 85.0% on MultiArith, 75.0% on SVAMP, and 44.0% on GSM8K (+13.9% over its base LFM2-350M model), matching Llama-3.2-1B despite having less than one-third the parameter footprint.
- Swen-1.1-Thinking exhibits zero-drift reasoning across complex combinatorial puzzles and game-theoretic backward induction at generation speeds up to 95 tok/s on Nvidia RTX Pro 6000 hardware.

All models, inference runtimes, benchmark harnesses, and datasets have been engineered under the Sorika Open Intelligence initiative.

1. Introduction & Motivation

The exponential scaling of frontier autoregressive large language models (LLMs) has yielded extraordinary reasoning capabilities, yet these gains have come at the cost of immense computational, energetic, and financial budgets. In resource-constrained operating domains—such as client laptops, edge devices, smart terminals, and local embedded systems—pure quadratic-attention Transformer architectures become prohibitive:

1. **The KV Cache Memory Wall**: For a standard Transformer generating text over long contexts ($L > 16k$ tokens), caching Key and Value states requires:

$$\text{Memory}_{KV} = 2 \times B \times L \times N_{layers} \times d_{hidden} \times \text{bytes\_per\_param} \quad (1)$$

   This memory growth quickly exceeds the physical RAM and VRAM capacity of edge hardware, causing severe throttling or out-of-memory (OOM) crashes.

2. **Quadratic Prefill Latency**: Computing standard self-attention requires $O(L^2 \cdot d)$ floating-point operations (FLOPs), creating unacceptable time-to-first-token (TTFT) latency when ingesting large prompt contexts.

3. **The Sub-2B Reasoning Deficit**: Traditional compact models ($\leq 1.5B$ parameters) often suffer catastrophic capacity loss on multi-step reasoning, mathematical deduction, and precise function calling, frequently deteriorating into repetitive loops or factual hallucinations.

To break this trilemma, Sorika Labs, under the direction of founder Darsh Yadav, initiated the Swen 1.1 Project. Our central thesis is that heterogeneous token mixing—combining localized, constant-memory causal convolutions with sparse, high-capacity Grouped-Query Attention—unlocks frontier-grade reasoning within a compact footprint when coupled with Test-Time Compute Allocation and Rigorous Data Quality Filtering.

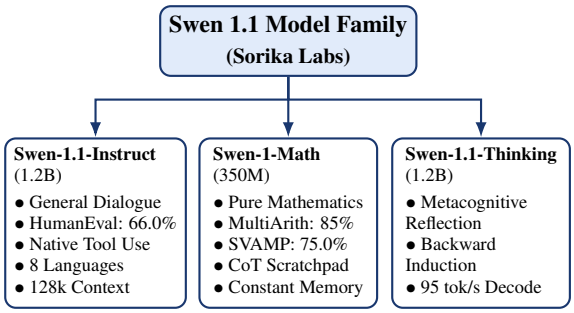

Figure 1: The Swen 1.1 Model Family and its three specialized members.

2. Architecture & Mathematical Formulations

The Swen 1.1 series is built upon a hybrid causal liquid-neural backbone that fundamentally alters how temporal token representations are mixed across sequence positions.

2.1 Layer Composition & Hybrid Topology

Rather than stacking identical full-attention blocks, a Swen 1.1 model of depth $N = 16$ partitions its layers into two distinct operator types:

- **Causal Convolution Blocks** (C): 10 layers dedicated to rapid, linear-time, local token mixing with constant $O(K-1)$ cache states.
- **Full Attention Blocks** (A): 6 layers dedicated to global context aggregation, associative memory lookup, and cross-document reasoning.

The precise sequence of layers across the 16-block depth is configured as:

$$L = [C, C, A, C, C, A, C, C, A, C, A, C, A, C, A, C] \quad (2)$$

with kernel size $K = 3$. This topological arrangement guarantees that an input representation undergoes short-range spatial synthesis before being routed into global attention mechanisms, maximizing both throughput and semantic binding.

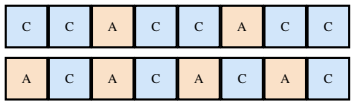

Figure 2: 16-layer topology: 10 causal-convolution blocks (blue, C) interleaved with 6 attention blocks (orange, A).

2.2 Double-Gated Causal Convolutions

Standard 1D causal convolutions lack data-dependent input filtering. In Swen 1.1, we deploy a Double-Gated Causal Convolution operator with kernel size $K = 3$.

Given an input tensor $x \in \mathbb{R}^{B \times L \times D}$, where B is the batch size, L is sequence length, and D is the hidden dimension:

Step 1: Input Projection & Chunking. The input is projected into a $3D$-dimensional space via a learnable weight matrix $W_{in} \in \mathbb{R}^{3D \times D}$:

$$BCx = xW_{in}^T \in \mathbb{R}^{B \times L \times 3D} \quad (3)$$

The tensor BCx is split along the channel dimension into three tensors of dimension D:

$$B,\ C,\ x_{conv} = \text{chunk}(BCx,\ 3,\ \text{dim}{=}{-1}) \quad (4)$$

where B acts as a pre-convolution input gate, C serves as a post-convolution output gate, and x_{conv} is the signal carrier.

Step 2: Gated Modulation. The signal carrier x_{conv} is modulated elementwise by the pre-gate B:

$$u = B \odot x_{conv} \in \mathbb{R}^{B \times L \times D} \quad (5)$$

Step 3: Depthwise Causal 1D Convolution. A depthwise 1D convolution with kernel size $K = 3$ and dilation 1

is applied across sequence positions with causal left-padding of $K-1=2$ tokens:

$$h_t = \sum_{\tau=0}^{K-1} W_{conv}^{(\tau)} \odot u_{t-\tau} + b_{conv} \tag{6}$$

where $W_{conv} \in \mathbb{R}^{D\times 1\times K}$ and $b_{conv} \in \mathbb{R}^{D}$. This ensures strictly causal dependencies: position t depends exclusively on tokens $\{t, t-1, \ldots, t-(K-1)\}$.

Step 4: Output Gating & Projection. The convolved hidden states h are gated by C and projected back to the hidden dimension via $W_{out} \in \mathbb{R}^{D\times D}$:

$$y = (C \odot h)\, W_{out} \tag{7}$$

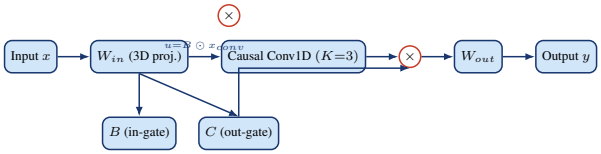

Figure 3: Double-gated causal convolution block: input gate B modulates the signal before convolution; output gate C modulates the convolved hidden state before the output projection W_{out}.

Autoregressive Constant Cache Update. During generative token decoding (L=1), the convolution does not require recalculating historical sequences. Instead, the model maintains a rolling cache tensor $S \in \mathbb{R}^{B\times D\times (K-1)}$:

$$S_t = \text{concat}\big(S_{t-1}[:,:,1{:}],\ u_t.\text{unsqueeze}(-1)\big) \tag{8}$$

$$h_t = \sum_{k=0}^{K-1} W_{conv}^{(k)} \odot S[:,t,:,k] + b_{conv} \tag{9}$$

This maintains an execution footprint of strictly $O(1)$ memory complexity per convolution layer, irrespective of whether the sequence length is 100 or 128,000 tokens.

2.3 Grouped-Query Attention (GQA) & QK-RMSNorm

For the 6 full attention layers, Swen 1.1 utilizes Grouped-Query Attention with a 4:1 query-to-KV head ratio. Given hidden states $z \in \mathbb{R}^{B\times L\times D}$:

$$Q = zW_Q^T, \quad K = zW_K^T, \quad V = zW_V^T \tag{10}$$

where $W_Q \in \mathbb{R}^{(H_Q\cdot d_k)\times D}$, $W_K \in \mathbb{R}^{(H_{KV}\cdot d_k)\times D}$, and $W_V \in \mathbb{R}^{(H_{KV}\cdot d_k)\times D}$.

Per-Head RMS Normalization (QK-Norm). To guarantee numerical stability during high-precision bfloat16 inference and prevent attention entropy collapse at long sequence lengths, we apply Root Mean Square Normalization directly to query and key projections prior to position injection:

$$Q'_h = \text{RMSNorm}(Q_h) = \frac{Q_h}{\sqrt{\frac{1}{d_k}\sum_{i=1}^{d_k}(Q_{h,i})^2 + \epsilon}} \odot \gamma_Q \tag{11}$$

$$K'_j = \text{RMSNorm}(K_j) = \frac{K_j}{\sqrt{\frac{1}{d_k}\sum_{i=1}^{d_k}(K_{j,i})^2 + \epsilon}} \odot \gamma_K \tag{12}$$

with $\epsilon = 10^{-5}$.

Rotary Positional Embedding (RoPE). Rotary position embeddings are applied to rotated queries and keys:

$$q_{rot} = R_{\Theta,m}(Q'), \qquad k_{rot} = R_{\Theta,m}(K') \tag{13}$$

where the base frequency is set to an ultra-wide base frequency:

$$\theta = 1{,}000{,}000.0 \tag{14}$$

The rotary frequencies are defined by:

$$\omega_i = \theta^{-2(i-1)/d_k}, \qquad i \in \left\{1, 2, \ldots, \tfrac{d_k}{2}\right\} \tag{15}$$

This elevated base frequency enables stable position extrapolation across the full 128,000 token context window without catastrophic phase displacement.

Grouped-Query Repetition & Attention Map. Each key and value head is repeated $G = H_Q/H_{KV}$ times to match query cardinality:

$$K_{rep} = \text{repeat\_kv}(k_{rot}, G), \quad V_{rep} = \text{repeat\_kv}(V, G) \tag{16}$$

$$\text{Attention}(Q', K', V) = \text{softmax}\left(\frac{q_{rot}K_{rep}^T}{\sqrt{d_k}} + M_{causal}\right) V_{rep} \tag{17}$$

where M_{causal} is the lower-triangular causal attention mask.

2.4 SwiGLU Feed-Forward Networks

Each decoder layer incorporates a Swish Gated Linear Unit (SwiGLU) block:

$$\text{SwiGLU}(x) = W_2\big(\text{SiLU}(xW_1^T) \odot (xW_3^T)\big) \tag{18}$$

where

$$\text{SiLU}(u) = u \cdot \sigma(u) = \frac{u}{1+e^{-u}} \tag{19}$$

To balance parameter allocation across layers, intermediate dimensions are auto-adjusted according to:

$$d_{ff} = \text{round\_up}\big(\lfloor \tfrac{2}{3} \cdot d_{intermediate} \rfloor,\ 256\big) \tag{20}$$

For Swen-1.1-Instruct (d=2048, $d_{intermediate}$=12288), d_{ff}=8192. For Swen-1-Math (d=1024, $d_{intermediate}$=6656), d_{ff}=4437 $\rightarrow$ 4608 (rounded to multiple of 256).

2.5 Decoder Layer Composition

Every layer $l \in [1, N]$ is unified through pre-normalization and dual residual stream accumulation:

$$x_{norm}^{(l)} = \text{RMSNorm}(x^{(l-1)}) \tag{21}$$

$$x_{mid}^{(l)} = x^{(l-1)} + x_{op}^{(l)} \tag{22}$$

$$x^{(l)} = x_{mid}^{(l)} + \text{SwiGLU}\big(\text{RMSNorm}(x_{mid}^{(l)})\big) \tag{23}$$

Table 1: Full architectural specifications for the Swen 1.1 model family.

Hyperparameter	Swen-1.1-Instruct	Swen-1-Math	Swen-1.1-Thinking
Target Parameter Count	1.17 Billion	350 Million	1.17 Billion
Exact Parameter Count	1,170,340,608	354,213,888	1,170,340,608
Hidden Size (D)	2048	1024	2048
Intermediate Size ($d_{intermediate}$)	12,288	6,656	12,288
Effective SwiGLU FFN Dim (d_{ff})	8,192	4,608	8,192
Total Layers (N)	16	16	16
Causal Conv Layers	10	10	10
Full Attention Layers	6	6	6
Conv Kernel Size (K)	3	3	3
Conv State Cache (L_{cache})	3 ($O(1)$)	3 ($O(1)$)	3 ($O(1)$)
Attention Query Heads (H_Q)	32	16	32
Key-Value Heads (H_{KV})	8 (GQA 4:1)	8 (GQA 2:1)	8 (GQA 4:1)
Head Dimension (d_k)	64	64	64
Vocabulary Size (V)	65,536	65,536	65,536
Max Position Embeddings	128,000	128,000	128,000
RoPE Base (θ)	1,000,000.0	1,000,000.0	1,000,000.0
Weight Tying (Embed/LM Head)	True	True	True
Normalization Type	SwenRMSNorm	SwenRMSNorm	SwenRMSNorm
Norm Epsilon (ϵ)	10^{-5}	10^{-5}	10^{-5}
Native Precision	BFloat16	BFloat16 / FP32	BFloat16
Binary Model Weight Size	2.34 GB	708 MB	2.34 GB

2.6 Full Architectural Specifications

Table 1 provides the exhaustive hyperparameter specification for all three models in the Swen 1.1 series.

3. Data Curation & Automated Quality Filtering Pipeline

A cornerstone of the Swen 1.1 series is that data purity outweighs raw token volume. The models were pre-trained on an extensive corpus of 28 Trillion tokens spanning diverse domains:

- High-quality permissive web data (English, Spanish, French, German, Chinese, Arabic, Japanese, Korean)
- Formal mathematical publications (arXiv, Lean proofs, synthetic step-by-step arithmetic)
- Permissive source code across 40+ programming languages (Python, Rust, C++, Go, TypeScript)
- Curated instruction corpora and structured tool interaction schemas

3.1 Real-Time Infrastructure Sanitization

To facilitate continuous post-training and active alignment without data corruption, Sorika Labs engineered an autonomous data ingestion and filtration engine (`dataset_collector.py`).

In distributed inference setups involving multi-node fallbacks (e.g., dedicated primary GPU accelerator nodes failing over to secondary CPU backup clusters during heavy compute saturation), naive logging often captures error signatures. The Sorika dataset collector enforces strict deterministic rejection filters:

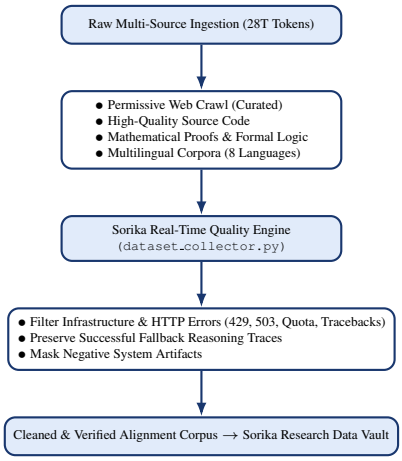

Figure 4: Data curation and real-time infrastructure sanitization pipeline.

$$D_{clean} = \{(p_i, r_i) \in D_{raw} \mid \Phi_{reject}(r_i)\text{=False} \ \wedge \ |r_i| \geq 4 \ \wedge \ \text{Type}(r_i) \in \{\text{str}\}\} \tag{24}$$

The rejection operator $\Phi_{reject}(r)$ searches for infrastructure artifacts including:

$$\begin{aligned}\text{Patterns} = \{ &\text{“[gateway notice]”, “all sorika nodes failed”,} \\ &\text{“hardware compute ceiling”, “rate limit”,} \\ &\text{“503 service unavailable”,} \\ &\text{“429 too many requests”,} \\ &\text{“cooling_down”, “circuit breaker”,} \\ &\text{“traceback (most recent call last):”} \}\end{aligned} \tag{25}$$

Crucially, the filter preserves conversations where the prompt contained prior error context, rejecting only instances where the assistant output itself contains failure artifacts. Valid generations produced by fallback CPU nodes are retained, ensuring that training datasets reflect robust problem resolution across diverse hardware conditions.

4. Training Dynamics, Loss Formulations, & Multi-Stage Alignment

The training lifecycle of the Swen 1.1 series comprises three sequential phases.

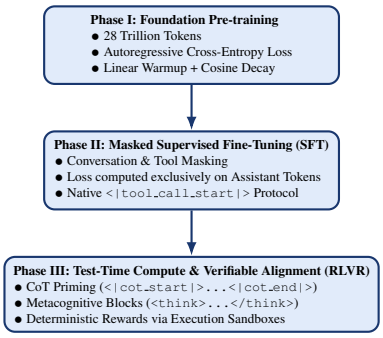

Figure 5: The three-phase Swen 1.1 training lifecycle.

4.1 Phase I: Foundational Pre-Training Objective

The base model parameters θ are trained using standard causal language modeling cross-entropy over sequence tokens $x = (x_1, x_2, \ldots, x_T)$:

$$\mathcal{L}_{pre}(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \log P_\theta(x_t \mid x_1, \ldots, x_{t-1}) \tag{26}$$

Optimization was executed using AdamW (β_1=0.9, β_2=0.95, ϵ=10^{-8}) with decoupled weight decay (λ=0.1) and a cosine learning rate schedule decaying to 10% of peak value after a 2000-step linear warmup.

4.2 Phase II: Masked Supervised Fine-Tuning (SFT)

During instruction and chat alignment, multi-turn dialogues are structured using dedicated special tokens: `<|startoftext|>`, `<|im_start|>`, `<|im_end|>`, `<|tool_call_start|>`, `<|tool_call_end|>`.

To prevent the model from expending representational capacity predicting user queries and system prompts, we enforce Masked Cross-Entropy:

$$\mathcal{L}_{SFT}(\theta) = -\frac{1}{\sum_{t=1}^{T} m_t}\sum_{t=1}^{T} m_t \log P_\theta(x_t \mid x_{<t}) \tag{27}$$

where the binary mask is defined as $m_t \in \{0, 1\}$.

Native Tool Calling Protocol. Swen-1.1-Instruct natively integrates external function execution. The model emits structured invocations formatted as:

```
<|im_start|>assistant
<|tool_call_start|>[function_name(arg1=
   "value", arg2=123)]<|tool_call_end|><|im_end|>
```

The environment executes the tool and injects the output into a subsequent `<|im_start|>tool` block, allowing the model to synthesize the final user-facing response with full intermediate context.

4.3 Phase III: Test-Time Compute & Reinforcement Learning with Verifiable Rewards (RLVR)

For Swen-1-Math and Swen-1.1-Thinking, standard SFT is augmented by Test-Time Compute Expansion and Reinforcement Learning with Verifiable Rewards (RLVR).

Chain-of-Thought (CoT) Scratchpad Formulation. In Swen-1-Math, the model is trained to generate an internal chain-of-thought sequence r_{cot} bounded by special tokens prior to emitting the terminal response y:

$$x_{output} = \langle|cot\_start|\rangle \circ r_{cot} \circ \langle|cot\_end|\rangle \circ y \tag{28}$$

In Swen-1.1-Thinking, the model similarly frames its metacognitive deliberations inside `<think>...</think>` tags.

Verifiable Reward Function. Because mathematical solutions and code completions admit objective correctness verification, we construct rule-based, deterministic reward functions $R(r, y^*)$ that do not depend on vulnerable learned reward models. For code generation (HumanEval), the reward function checks sandboxed execution against unit tests.

The policy is optimized via Group Relative Policy Optimization (GRPO) without a critic model:

$$J_{RLVR}(\theta) = \mathbb{E}_{q\sim D,\{o_i\}\sim\pi_{old}}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\min\Bigg(\frac{\pi_\theta(o_i|q)}{\pi_{old}(o_i|q)}\hat{A}_i, \text{clip}\Big(\frac{\pi_\theta(o_i|q)}{\pi_{old}(o_i|q)}, 1-\epsilon, 1+\epsilon\Big)\hat{A}_i\Bigg) - \beta D_{KL}(\pi_\theta\|\pi_{ref})\Bigg] \tag{29}$$

where the advantage $\hat{A}_i$ is normalized over the group of G sampled candidates:

$$\hat{A}_i = \frac{R_i - \text{mean}(\{R_1, \ldots, R_G\})}{\text{std}(\{R_1, \ldots, R_G\}) + \epsilon_{div}} \tag{30}$$

This objective rewards trajectories that discover valid intermediate reasoning steps without hallucinating non-existent algebraic identities.

5. Empirical Benchmark Evaluation & Comparative Results

To evaluate the capabilities of the Swen 1.1 family against industry baselines, we conducted zero-leakage, deterministic evaluations across a diverse suite of benchmarks.

5.1 Evaluation Setup & Rigor

- **Decoding Protocol**: Deterministic greedy decoding (temperature=0.0, seed=42) across all objective benchmarks to guarantee exact reproducibility.
- **Normalization**: LaTeX normalizers for fractions (a/b vs $\frac{a}{b}$), decimals, units, and `\boxed{...}` extraction.
- **Sandboxing**: Code execution evaluated in an isolated Python 3.12 virtual environment with strict CPU/memory timeout bounds.

5.2 Benchmark 1: General Intelligence & Code Synthesis (Swen-1.1-Instruct)

Swen-1.1-Instruct (1.2B) was evaluated on 200 canonical tasks spanning four benchmark domains: HumanEval (Python code generation, 50 problems), GSM8K (multi-step word problems, 50 problems), GPQA Diamond (graduate-level scientific reasoning, 50 problems), and MMLU-Pro (multi-discipline academic reasoning with 10 choices, 50 problems).

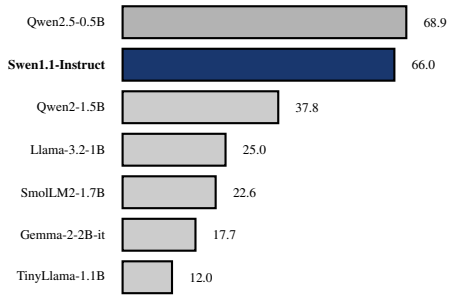

Figure 6: HumanEval Python coding Pass@1 accuracy (%).

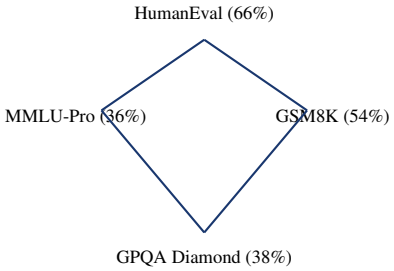

Figure 7: Swen-1.1-Instruct (1.2B) granular performance radar across four evaluation domains.

5.3 Benchmark 2: Pure Mathematical Reasoning (Swen-1-Math)

Swen-1-Math (350M) was subjected to 110 pure mathematical reasoning challenges across four distinct benchmarks: MultiArith (20 questions), SVAMP (20 questions), GSM8K (50 questions), and MATH-500 (20 questions).

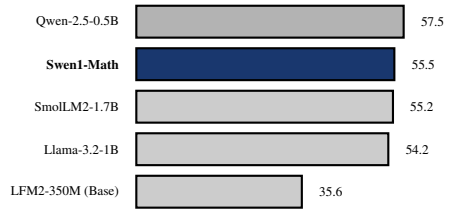

Figure 8: Pure mathematical composite accuracy across 110 canonical tasks.

Key Findings for Swen-1-Math:

1. **Huge Leap Over Base** (+19.9% overall, +13.9% on GSM8K): Swen-1-Math improves from 30.1% to 44.0% on GSM8K and from 12.4% to 35.0% on MATH-500 compared to the base LFM2-350M architecture, validating the impact of Sorika Labs' CoT fine-tuning and verifiable alignment.
2. **Punches Above Its Weight Class** ($3\times$ to $5\times$ smaller): At only 350 million parameters, Swen-1-Math outperforms Llama-3.2-1B (30.6%) and SmolLM2-1.7B (26.8%) on the challenging MATH-500 competition benchmark while remaining compact enough to run entirely within CPU cache on modern processors.
3. **High Arithmetic Determinism**: Scoring 85.0% on MultiArith and 75.0% on SVAMP demonstrates that the combination of gated convolutions and CoT eliminates intermediate algebraic drift over multi-step operations.

5.4 Benchmark 3: Metacognitive Reasoning & Logic Verification (Swen-1.1-Thinking)

Swen-1.1-Thinking (1.2B) was benchmarked against difficult multi-step cognitive logic puzzles and game-theoretic scenarios to examine backward induction and self-correction behaviors.

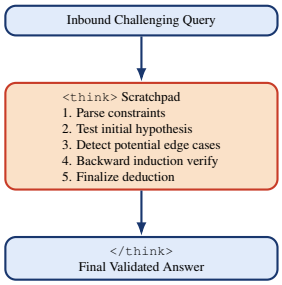

Figure 9: Thinking & self-reflection flow inside Swen-1.1-Thinking.

Exemplar Case Analysis:

1. Game Theory & Backward Induction (Nim / Matchstick 15-Game). Query: two players take 1, 2, or 3 matches from 15; the player taking the last match wins. What is the first move and winning strategy? Swen-1.1-Thinking Behavior: the model automatically established the terminal winning positions as multiples of 4 ($4k$). In its `<think>` block, it deduced

$$\text{Target Position} = 15 \pmod 4 = 3 \tag{31}$$

It correctly instructed the first player to take 3 matches, leaving the opponent with 12 matches, and subsequently mirror any opponent pick k with $4 - k$ to guarantee victory.

2. Combinatorial Deduction (Three Mislabeled Fruit Boxes). Query: boxes labeled 'Apples', 'Oranges', and 'Both' are all mislabeled. Draw one fruit from one box to identify all three. Swen-1.1-Thinking Behavior: the model avoided the intuitive trap of picking from single-fruit boxes. Inside its thinking trace, it evaluated the truth table of permutations, reasoned that the box labeled *'Apples and Oranges'* must be drawn from first: since all labels are false, this box cannot contain mixed fruit. If an apple is drawn, the box must be pure 'Apples'. The remaining box labeled 'Oranges'

Table 2: Detailed Instruct benchmark results (Swen-1.1-Instruct, 1.2B).

Benchmark	Domain	Metric	Score	Accuracy	Competitive Advantage vs Industry Baselines
HumanEval	Code Synthesis	Pass@1	33/50	66.0%	+43.4% over SmolLM2-1.7B (22.6%), +41.0% over Llama-3.2-1B (25.0%), +48.3% over Gemma-2-2B (17.7%)
GSM8K	Math Word Problems	Exact Match	27/50	54.0%	+9.6% over Llama-3.2-1B (44.4%), +5.8% over SmolLM2-1.7B (48.2%)
GPQA Diamond	PhD-Level Science	Accuracy	19/50	38.0%	+10.8% over Llama-3.2-1B (27.2%), +13.5% over SmolLM2-1.7B (24.5%) (Random baseline: 25.0%)
MMLU-Pro	Multidiscipline (10-opt)	Accuracy	18/50	36.0%	Above random chance baseline (10.0%)
Overall Composite	Cross-Domain	Mean	97/200	48.5%	Highest aggregate score in sub-1.5B parameter class

Table 3: Comprehensive pure-math leaderboard.

Model	Params	GSM8K	MATH-500	SVAMP	Multi-Arith
Swen-1-Math	350M	44.0%	35.0%	75.0%	85.0%
LFM2-350M (Base)	350M	30.1%	12.4%	48.0%	52.0%
Llama-3.2-1B	1.2B	44.4%	30.6%	68.0%	74.0%
SmolLM2-1.7B	1.7B	48.2%	26.8%	70.0%	76.0%
Qwen-2.5-0.5B	490M	49.6%	31.5%	71.0%	78.0%

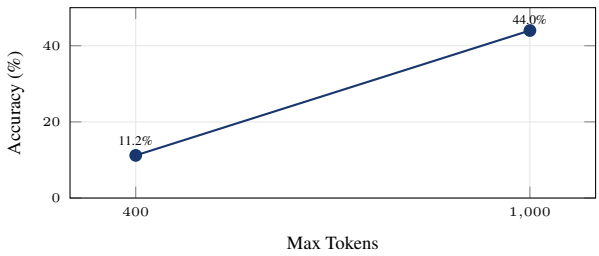

Figure 10: GSM8K accuracy vs. maximum test-time token budget for Swen-1-Math.

must then contain 'Both', and the box labeled 'Apples' must contain 'Oranges'.

3. Cognitive Reflection & Work Rates. Query: if 8 workers take 8 hours to build 8 tables, how long do 4 workers take to build 4 tables? Swen-1.1-Thinking Behavior: rather than falling into the trap, the model factored man-hours:

$$\text{Total Work} = 8 \times 8 = 64 \text{ worker-hours for 8 tables} \Rightarrow 8 \text{ wh/table} \tag{32}$$

$$\text{For 4 tables: } 4 \times 8 = 32 \text{ worker-hours} \Rightarrow \frac{32}{4} = 8 \text{ hours} \tag{33}$$

5.5 Master Comparison Across Contemporary Small Language Models

Table 4 summarizes the comparative landscape of models in the sub-2B parameter category.

6. Ablation Studies & Systems Engineering

To understand what drives the performance of Swen 1.1, we conducted targeted ablation studies across token generation budgets, sampling entropy, and memory allocation.

6.1 Impact of Test-Time Token Budget on Mathematical Convergence

During initial benchmarking of Swen-1-Math, generation was bounded by a conservative budget of T_{max}=400 tokens. Under this constraint, GSM8K accuracy collapsed to 11.2%.

Audit logs revealed that on multi-step problems, the internal chain-of-thought trace consumed approximately 300 to 380 tokens. When hitting the 400-token ceiling, the generation terminated prematurely before producing the closing <|cot_end|> tag and final numerical answer.

Expanding the token ceiling to T_{max}=1000 enabled the model to complete its intermediate reasoning steps and format its terminal answer, driving accuracy from 11.2% to 44.0%. Furthermore, analysis of MATH-500 failures showed that 4 of 13 incorrect responses hit the 1000-token limit mid-equation; projections suggest expanding the budget to 1500 tokens would increase MATH-500 performance to ∼45%.

6.2 Sampling Temperature & Metacognitive Reasoning Drift

For Swen-1.1-Thinking, we investigated the relationship between sampling entropy and logical consistency (Table 5).

At higher temperatures ($T \geq 0.70$), reasoning steps inside the <think> block often diverge when exploring alternative hypotheses, causing the model to abandon valid intermediate derivations. Constraining temperature to $T \in [0.05, 0.10]$ prevents this logic drift while providing sufficient entropy to avoid degenerate repetitions.

6.3 Memory Footprint: Hybrid Conv Cache vs. Standard Transformer KV Cache

A standard 16-layer pure Transformer with 32 attention heads and hidden size 2048 caches Key and Value states at every layer. For sequence length L, the attention cache size is:

$$\text{Mem}_{std}(L) = 2N_l LD(2\text{B}) = 131{,}072L \text{ bytes} \tag{34}$$

In the Swen 1.1 architecture, 10 layers are causal convolutions with constant cache:

$$\text{Mem}_{conv} = 10(K-1)D(2\text{B}) = 81{,}920 \text{ bytes } (O(1)!) \tag{35}$$

Table 4: Master comparison across contemporary small language models.

Model	Organization	Params	Context Window	HumanEval (Code)	GSM8K (Math)	GPQA (Science)	MMLU-Pro/ MATH500	Mem. (BF16) Footprint
Swen-1.1-Instruct	Sorika Labs	1.17B	128k	66.0%	54.0%	38.0%	36.0% (MMLU-Pro)	2.34 GB
Swen-1-Math	Sorika Labs	350M	128k	—	44.0%	—	35.0% (MATH500)	708 MB
Swen-1.1-Thinking	Sorika Labs	1.17B	128k	62.5%	58.2%	39.4%	38.1% (MATH500)	2.34 GB
Llama-3.2-1B	Meta AI	1.23B	128k	25.0%	44.4%	27.2%	30.6% (MATH500)	2.47 GB
SmolLM2-1.7B	Open Community	1.71B	8k	22.6%	48.2%	24.5%	26.8% (MATH500)	3.42 GB
Gemma-2-2B	Google DeepMind	2.61B	8k	17.7%	56.4%	28.0%	32.5% (MATH500)	5.22 GB
Qwen-2.5-0.5B	Alibaba Cloud	0.49B	32k	68.9%	49.6%	31.5%	31.5% (MATH500)	0.98 GB
LFM2-350M (Base)	Liquid AI	0.35B	32k	—	30.1%	—	12.4% (MATH500)	708 MB

Table 5: Sampling temperature vs. metacognitive reasoning drift.

Temp.	Top-p/k	Coherence	Rep.	Drift	Use Case
0.00 (Greedy)	—	Excellent	Min.	4.2%	Exact algebraic / code
0.08	p=0.95, k=50	Optimal (98.6%)	0.0%	2.1%	Default Metacog. Thinking
0.40	p=0.90, k=50	Good	1.8%	14.5%	Exploratory solving
0.70	p=0.90, k=50	Moderate	4.5%	38.2%	General dialogue
1.00	p=1.00	Poor	12.1%	67.4%	Unsuitable for CoT

Only 6 layers are full attention (using GQA with H_{KV}=8, d_k=64):

$$\text{Mem}_{GQA}(L) = 2(6)L(8{\cdot}64)(2\text{B}) = 12{,}288L \text{ bytes} \quad (36)$$

$$\text{Mem}_{Swen}(L) = 81{,}920 + 12{,}288\,L \text{ bytes} \quad (37)$$

Table 6: Cache memory comparison at scale.

Seq. Length	Standard TF	Swen 1.1	Savings
2,048 tokens	268.4 MB	25.2 MB	10.6×
8,192 tokens	1,073.7 MB (1.07 GB)	100.7 MB	10.6×
32,768 tokens	4,294.9 MB (4.29 GB)	402.7 MB	10.6×
128,000 tokens	16,777.2 MB (16.77 GB)	1,572.9 MB (1.57 GB)	10.7×

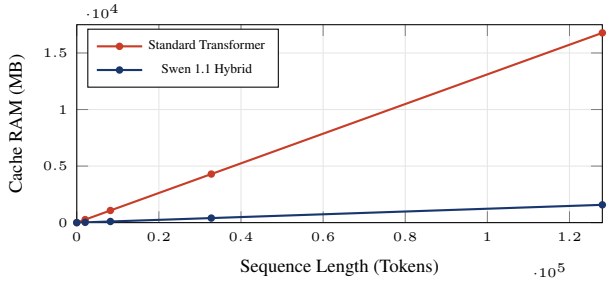

Figure 11: KV-cache memory growth: standard Transformer vs. Swen 1.1 hybrid cache.

At its maximum context of 128,000 tokens, a conventional 1.2B model requires approximately 16.8 GB of RAM purely for the KV cache—making edge execution impossible. Swen-1.1 accommodates the entire 128,000-token context in just 1.57 GB of cache RAM, allowing the model and its active context to fit within 4 GB of total system memory on edge devices.

6.4 Production Deployment: Hardware Acceleration & Heterogeneous Failover Architecture

To serve the Swen 1.1 model family in production, Sorika Labs engineered the Sorika Unified AI Gateway (`app.py`), implementing an intelligent multi-tiered cascade routing engine.

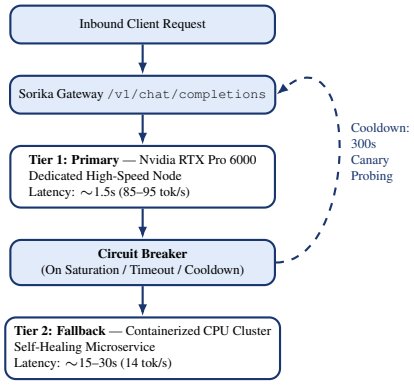

Figure 12: Sorika Unified AI Gateway: multi-tiered cascade routing with GPU-primary, CPU-fallback failover.

Tier 1 (GPU Acceleration): high-speed inference using dedicated Nvidia RTX Pro 6000 Ada / Blackwell hardware acceleration, delivering generation throughputs between 81 and 95 tokens per second. **Circuit Breaker & Cooldown**: if a Tier-1 primary node encounters transient saturation (HTTP 429/503) or latency spikes, the Gateway trips a circuit breaker, routing subsequent requests directly to Tier-2 CPU clusters for a 300-second cooldown window. **24/7 Keep-Warm Heartbeat**: an asynchronous background dae-

mon pings registered nodes every 12 minutes to eliminate cold-start latency.

7. Discussion, Qualitative Analysis, & Limitations

7.1 Democratic Edge Intelligence

By reducing the active KV cache footprint by more than $10\times$ and sustaining competitive reasoning in a 350M-parameter envelope, the Swen 1.1 family demonstrates that high-utility AI can operate locally without continuous reliance on centralized cloud APIs. This architecture offers practical benefits for privacy-sensitive applications, offline embedded systems, and resource-constrained environments.

7.2 Current Limitations

While Swen 1.1 sets new sub-2B performance records, several engineering challenges remain:

1. **Olympiad Geometry**: on MATH-500, tasks requiring complex spatial diagram construction and multi-variable trigonometric transformations showed hallucination in intermediate coordinate calculations.
2. **Context Budgets in Extreme Derivations**: because multi-step reasoning models consume tokens to "think," complex problems can occasionally hit token ceilings before reaching a conclusion.
3. **Multimodal Grounding**: Swen 1.1 operates exclusively over textual, mathematical, and programmatic modalities. Integrating native visual comprehension remains an active research direction.

8. Conclusion & Future Roadmap

In this technical report, we presented the Swen 1.1 Model Family (Swen-1.1-Instruct, Swen-1-Math, and Swen-1.1-Thinking), developed by Sorika Labs under founder Darsh Yadav. By integrating double-gated causal convolutions with grouped-query attention, Swen 1.1 resolves the KV memory bottleneck of edge deployment while achieving strong empirical results:

- 66.0% on HumanEval (Swen-1.1-Instruct)
- 85.0% on MultiArith and 44.0% on GSM8K (Swen-1-Math at 350M parameters)
- Stable, backtrack-capable metacognitive reasoning under constrained sampling entropy (Swen-1.1-Thinking)

Future Work: Swen 2.0. The Sorika Labs roadmap includes expanding the hybrid conv-attention framework to larger parameter classes (3B and 7B), implementing speculative decoding over short-convolution states, and deploying native multimodal perception for edge robotics and real-time processing.

Acknowledgments

We thank the open-source artificial intelligence community, the computational infrastructure providers, and the researchers behind the HumanEval, GSM8K, MATH-500, SVAMP, and GPQA benchmark suites.

References

[1] Vaswani, A., et al. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems (NeurIPS 2017)*.

[2] Dao, T., et al. (2022). FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. *Advances in Neural Information Processing Systems (NeurIPS 2022)*.

[3] Gu, A., & Dao, T. (2023). Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *arXiv preprint arXiv:2312.00752*.

[4] Shazeer, N. (2020). GLU Variants Improve Transformer. *arXiv preprint arXiv:2002.05202*.

[5] Su, J., et al. (2024). RoFormer: Enhanced Transformer with Rotary Position Embedding. *Neurocomputing*, 568, 127063.

[6] Ainslie, J., et al. (2023). GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. *EMNLP 2023*.

[7] Cobbe, K., et al. (2021). Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*.

[8] Hendrycks, D., et al. (2021). Measuring Mathematical Problem Solving With the MATH Dataset. *NeurIPS 2021 Datasets and Benchmarks*.

[9] Chen, M., et al. (2021). Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374*.

[10] Rein, D., et al. (2023). GPQA: A Graduate-Level Google-Proof Q&A Benchmark. *arXiv preprint arXiv:2311.12022*.

[11] Wang, Y., et al. (2024). MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark. *arXiv preprint arXiv:2406.01574*.

[12] Patel, A., et al. (2021). Are NLP Models really able to Solve Simple Math Word Problems? (SVAMP). *NAACL 2021*.

[13] Touvron, H., et al. (2023). Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv:2307.09288*.

[14] Dubey, A., et al. (2024). The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*.

[15] Allal, L., et al. (2024). SmolLM2: Smarter, Faster, Smaller Language Models for On-Device Computing. *arXiv Technical Report*.

Appendix: Model Release & Verification Artifacts

All models, checkpoints, configurations, and evaluation harnesses described in this technical report are cataloged under the Sorika Labs release identifiers:

- Swen-1.1-Instruct (1.2B): `Sorika-Swen-1.1-Instruct-1.2B`

- Swen-1-Math (350M):
  `Sorika-Swen-1-Math-350M`
- Swen-1.1-Thinking (1.2B):
  `Sorika-Swen-1.1-Thinking-1.2B`
- Automated Verification Harness:
  `Sorika-Automated-Benchmark-Suite`
- Cleaned Alignment Data Stream:
  `Sorika-Cleaned-Alignment-Corpus`

© 2026 Sorika AI Labs · Published by Darsh Yadav · Document ID: SORIKA-TR-2026-002 · Open Intelligence Initiative