

ICoder-27B: Recursive AI-Led Development of Frontier Industrial Coding Model

A Technical Review and Analysis

PRATYUSH SINGH¹

¹C-9, 9796, VASANT KUNJ, NEW DELHI, 110070

Abstract

Recursive AI, the prospect of artificial intelligence taking an increasingly complete role in building and improving artificial intelligence, has so far been demonstrated mostly on small models, bounded tasks, and fixed time budgets. A recent technical report by Yang and colleagues [1] asks a more demanding question: how little human involvement is sufficient for an agent to lead the development of a release-ready, frontier-competitive model? Their answer is an agent-led pipeline that produced iCoder-27B, a 27-billion-parameter model for register-transfer-level (RTL) hardware design and GPU kernel optimization. This paper is an independent technical review and analysis of that report; it does not present new training results. We reconstruct the design principle of high-density human priors with low-frequency intervention, the executable research skills that carry those priors, and the four-stage pipeline of data evolution, supervised fine-tuning, on-policy self-distillation, and reinforcement learning with verifiable rewards. We summarize the reported evidence, including a score of 68.0 on RTLLM, 61% correctness on the first level of Kernel Bench, a tie for the best Triton Bench-G result at 20.1, and exploratory search case studies that approach larger models at a fraction of the output tokens. We then analyze what the evidence does and does not establish, discuss how the verifier-contract idea may extend to other industrial software such as process monitoring, illustrated with two anomaly detection examples, and identify open questions about attribution of human and agent contributions, controls, and statistical reporting.

Keywords: Recursive AI, Agentic Model Development, Industrial Coding, RTL Generation, GPU Kernel Optimization, Reinforcement Learning with Verifiable Rewards, On-Policy Self-Distillation, Anomaly Detection.

1. Introduction

Artificial intelligence agents are moving from assisting isolated research tasks to carrying out increasingly complete scientific workflows. Recent systems can take a project from hypothesis through experiment to manuscript [26], search executable solution spaces under quantitative feedback, and sustain long investigations through persistent memory. Yet the strongest evidence to date comes from bounded loops in which the data, the evaluator, and the action space are fixed in advance, and in which success is measured on a single target.

A technical report by Yang et al., posted in August 2026 as arXiv:2609.29626, pushes on this boundary [1]. It asks whether an AI agent can coordinate the development of a genuinely competitive frontier model, and it frames the practical version of that question as follows: how little human involvement is sufficient for an agent to lead the development of a release-ready model? The authors do not claim fully autonomous recursive self-improvement. Instead they concentrate human input into what they call a high-density, low-frequency interface. Experts write down objectives, stage scaffolds, permission boundaries, and operating procedures once, as reusable research skills, and a coding agent then chooses experiments, diagnoses failures, and revises the training strategy. The domain is industrial coding, meaning code whose acceptance is decided by specialized executable toolchains: RTL design for hardware, and GPU kernel optimization for accelerators. The resulting model, iCoder-27B, is built on a 27-billion-parameter base model and, according to the report, matches or exceeds much larger open and proprietary models on several benchmarks.

This paper provides an independent technical review of that report. It reconstructs the method, collects the reported evidence in one place, and analyzes strengths, limitations, and implications. It reports no new experiments, and all numerical results in Sections V and VI are those reported by Yang et al. [1]. We add three elements of our own. We read the reported search trajectories for GPU operators, shown in Fig. 1, and relate them to the claims made about cost and quality. We discuss how the central design idea, trusted executable verifiers wrapped in explicit contracts, might carry over to other kinds of industrial software, using two examples of anomaly detection in process monitoring shown in Fig. 2 and Fig. 3. Those two figures are not part of the iCoder report; they are included only as illustrations of the kind of oracle a monitoring application can provide. Finally, we collect open questions that future work should address.

The contributions of this review are as follows:

- A structured reconstruction of the iCoder development pipeline, including the human prior, the shared executable task pool, and the four training stages.
- A consolidated summary of the reported benchmark results, ablations, training dynamics, and case studies, with attention to what each experiment can support.
- A reading of the kernel search trajectories in Fig. 1, showing where the compact model leads, where it trails, and why the reported cost comparison must be interpreted cautiously.
- A discussion of how verifier contracts could generalize to industrial monitoring software, illustrated by Fig. 2 and Fig. 3, and an inventory of open methodological questions.

The remainder of the paper is organized as follows. Section II reviews background literature. Section III discusses related work. Section IV describes the methodology of the reviewed report. Section V describes its experimental setup. Section VI summarizes results. Section VII discusses implications, limitations, and extensions, and Section VIII concludes.

2. LITERATURE REVIEW

A. Levels of Recursive Self-Improvement

Self-improving AI systems are best distinguished by what is revised across iterations. At the first level, a fixed model proposes candidate solutions that are tested against an evaluator, and improvement accumulates in the resulting artifact rather than in the model that produced it. Evolutionary coding agents for algorithmic discovery are of this kind [18], [19]. At the second level, the recursion moves into the agent harness, so that execution traces change how a frozen model behaves on later tasks, for example through evolving prompts, skills, or tools [20]. At the third level, experience is written back into the model parameters by training on self-generated trajectories and verifiable outcomes. Only this level makes the acquired capability persist independently of any harness, and it is the level most directly related to autonomous model development.

B. Industrial Coding: RTL Design and GPU Kernels

The term industrial coding in the reviewed report denotes code generation problems whose acceptance is determined by specialized executable toolchains and deployment quality rather than by surface similarity. Two regimes are covered. In hardware design, RTL generation turns a behavioral specification into a clocked circuit description that must elaborate, synthesize, and behave as intended. Benchmarks have progressed from short isolated modules, as in VerilogEval and RTLLM [7], [8], toward realistic specifications and hierarchical designs with simulation-based validation, as in CVDP, ArchXBench, and RealBench [9], [10], [11]. In accelerator programming, a kernel must compile, preserve numerical semantics, use the machine efficiently, and deliver measured speedup, as assessed by KernelBench and TritonBench [5], [6].

C. Post-Training with Verifiable Signals

Modern post-training of language models for reasoning combines several techniques. Supervised fine-tuning on verified teacher trajectories provides a cold start. On-policy distillation evaluates teacher supervision on student-generated trajectories, which reduces the mismatch between training and inference distributions [14]. A related idea, self-distillation, uses the same model as both teacher and student, with the teacher given additional context such as feedback or a plan [21], [22].

Reinforcement learning with verifiable rewards then lets the policy explore and receive rewards computed from executable outcomes. Group-relative policy optimization estimates advantages by comparing responses to the same prompt [12], and sequence-level variants adjust the importance ratio and clipping for stability [13], building on the clipped policy gradient family [15].

D. Human Oversight in Agentic Research

Delegating research to agents raises questions of governance, and the report frames its approach through the bitter lesson that general methods scale with computation [16]; one benchmark of agentic post-training studies the same ambition on small models with short runs [17]. Which decisions must remain with humans, how often must humans intervene, and how can the contribution of each be attributed? The reviewed report answers with a deliberate boundary: humans supply the objective, permissions, verifier semantics, and the process prior, while the agent makes the empirical choices.

3. RELATED WORK

Related work relevant to the reviewed report can be grouped into bounded solution-search systems, harness-level self-improvement, agentic post-training benchmarks, and domain-specific coding models.

Bounded solution search. Systems such as AlphaEvolve and open-source evolutionary coding agents use a fixed model to propose programs and an automatic evaluator to select among them [18], [19].

Harness-level self-improvement. Recent work refines prompts, skills, and tools from execution traces, which can raise system performance without changing weights [20].

Agentic post-training. PostTrainBench evaluates whether language model agents can automate post-training of small models under tight budgets [17]. The reviewed report operates at a different scale: a multi-stage process on a 27-billion-parameter model with substantial compute, whose outcome is a release-ready checkpoint.

Domain coding models and search frameworks. For hardware, RTLScout studies joint optimization of code and synthesis results [23], and for broader scientific discovery, SimpleTES provides a propose–evaluate–refine interface that the report adopts for its case studies [24].

TABLE I. Positioning of the Reviewed Report Relative to Related Approaches

Approach	What is revised	Evaluator	Scale	Release-ready model
Evolutionary coding agents [19]	Program artifacts	External, fixed	Single problems	No
Harness refinement [20]	Prompts, skills, tools	External, fixed	Task suites	No
Agentic post-training benchmark [17]	Small model weights	Single target benchmark	Small models, short runs	No
Search frameworks [23], [24]	Candidate designs or kernels	Testbench or latency	Per design or operator	No
iCoder report [1]	Data, objectives, rewards, weights	Task-native verifiers	27B model, multi-stage	Yes (claimed)

Table I shows the niche the report occupies: it is the only entry that revises model weights through a multi-stage pipeline at 27-billion-parameter scale with a release-oriented outcome.

4. METHODOLOGY OF THE REVIEWED REPORT

A. Design Principle: High-Density Prior, Low-Frequency Intervention

The central design principle of the report is to encode expert model-development experience once and then step back. Humans write a versioned collection of research skills: concise operational procedures backed by reusable code that make established practice executable on a

GPU cluster. These skills describe how to access approved resources, construct auditable training artifacts, launch distributed jobs, diagnose failures, and consume executable verification. They also fix invariants that should not be discovered by trial and error, such as separation of held-out data, verifier integrity, permission boundaries, and the requirement that every reported checkpoint be traceable to its data, code, configuration, and parent model. The agent is then free to choose data transformations, privileged context, optimization objectives, reward adapters, and checkpoints.

The agent that carried out the development was a commercial coding agent, reported as Codex GPT-5.6-Sol operating at its highest reasoning effort. Starting from the base checkpoint and the available compute, it constructed an executable task pool, cold-started the base model with verified trajectories, used on-policy self-distillation as a bridge, and finished with reinforcement learning. The report stresses that the final recipe was not a fixed human-authored procedure but the result of failures and revisions along the way.

B. Executable Research Skills and the Attribution Boundary

Table II summarizes the layers of the human prior as described in the report. Each layer pairs human-specified content with an explicit limit on what the agent may do. The agent may select experiments and transitions but may not silently redefine the deliverable; it may use registered capabilities within budget, but new identities or permissions require a human gate; it may integrate and audit verifiers but not weaken the approved correctness criterion; and it may formulate hypotheses but must cite registered evidence for conclusions. A task queue and an experiment log record research questions, budgets, expected artifacts, and decisions, and the released human prior stays immutable during a run. This structure is what allows the authors to claim an attribution boundary between initial human expertise and agent-led development.

TABLE II. Layers of the Human Prior and Boundaries on Agent Action

Prior layer	Human-specified content	Boundary on agent action
Objective and scaffold	Target capability, completion criteria, and the data, SFT, OPSD, RLVR stage roles	May select experiments and transitions; may not silently redefine the deliverable
Resource access	Approved repositories, models, data, launchers, compute, storage, side-effect permissions	May use registered capabilities within budget; new permissions need a human gate
Verification	Task-contract schema, official harness, integrity checks, pass, fail, and unjudged semantics	May integrate and audit verifiers; may not weaken the approved correctness criterion
Research method	Controlled pilots, explicit baselines, artifact lineage, stop rules, regression-aware selection	May formulate hypotheses; conclusions must cite registered evidence
Governance and memory	Task queue, experiment log, decision records, approval boundaries	May record findings and negative results; the released prior stays immutable

C. Shared Executable Task Pool

Every later stage requires a trustworthy signal about the correctness of generated code, so the pipeline begins with data construction. The agent normalized heterogeneous seeds from the two domains into a common contract consisting of a natural-language specification, a boundary for the generated artifact, an interface, a reference status, and a trusted verifier. For RTL this records ports, clock and reset behavior, timing requirements, and the behavioral oracle. For kernels it records the model interface, argument shapes, numerical semantics, and the permitted implementation boundary.

Seeds were routed among three evolution families according to which contract elements a transformation affects. Solution-first reconstruction derives a new specification from a verified implementation while preserving its behavior and verifier. Instruction-first evolution changes the specification and regenerates the affected executable components before validation. Contract-bound mutation applies a declared behavioral change to the reference and then derives

a matching instruction and test; the evolved reference must pass the evolved test while the original reference must fail it. Candidates then pass ordered admission gates covering interface preservation, leakage checks, semantic-change tests, solvability, compilation, and execution, with cheap checks first. The resulting shared pool contains 55,462 verifier-backed tasks, of which 34,972 (63.1%) are RTL and 20,490 (36.9%) are kernels.

D. Supervised Fine-Tuning from Verified Teacher Trajectories

Supervised fine-tuning provides a reasoning cold start on tasks where the base model has no verified success. The agent selected tasks for which the base model failed all four sampled attempts and at least one of four trajectories from a stronger teacher passed the verifier. The teacher in this stage was DeepSeek-V4-Pro, and one complete passing trajectory, including its reasoning and final answer, was retained per task. Training integrity rules required that training pairs never be truncated and that length distributions be audited. The resulting corpus contains 28,952 trajectories, 63.0% RTL and 37.0% kernel, and full-parameter fine-tuning on it produced the cold-start checkpoint.

E. On-Policy Self-Distillation: Discovery of a Verifier-Anchored Objective

On-policy self-distillation is the most distinctive stage. Its purpose is to convert observed self-correction into persistent capability: tasks that the model can solve only after receiving guidance or feedback are used to teach it to solve them directly. The agent first identified a recoverable capability frontier by retaining tasks that failed all four attempts, failed the first bare-prompt attempt, and were solved in a later round of an execution-feedback loop. This yielded 1,874 tasks, 72.0% RTL and 28.0% kernel.

For these tasks the agent constructed a privileged context for the teacher view, which is the same model given more information than the student. Controlled ablations, summarized in Section VI, led it to include a direction-only repair plan and the failed responses with coarse verifier feedback, and to exclude generic domain hints and fine-grained counterexamples. The plan had to pass three leakage controls: an instruction-level prohibition on solution content, a rule-based scan for code and answer-specific overlap, and an audit by a model different from the plan generator.

The first objective used the gap between how likely each token was under the privileged view and under the bare view as the direction of an online update. Because the privileged view was scored by the current model, its distribution moved with the student, creating a self-reinforcing loop. Early checkpoints improved some short tasks but regressed harder ones, matched trajectories on one benchmark became roughly 2.9 times longer, some generations repeated an intermediate reasoning segment without emitting code, and continued training ended in degenerate repetition. The agent diagnosed this as a conflation of contextual preference with environmental correctness, and the report summarizes the lesson as follows: privilege can allocate credit but not authority.

The replacement is an offline five-stage pipeline: roll out from the bare prompt, score the rollouts with a frozen privileged teacher, verify them by execution, build signed advantages, and apply a clipped importance-weighted update. Freezing and precomputing the teacher scores removed the moving-teacher feedback path. Execution outcomes determine the sign and size of each response's advantage through a hierarchical scheme in which compilation failures receive the largest penalty, other incorrect responses a smaller one, and correct responses receive a performance-based term, after which advantages are centered within the group of rollouts for the same prompt. The teacher view is used only to modulate token-level credit within bounds, never to set the direction of learning. Section VI compares this design with two alternatives.

F. Reinforcement Learning with Verifiable Rewards

The final stage starts from the self-distilled checkpoint and optimizes against rewards obtained by compiling and executing generated code. The agent first measured per-task success rates under the current policy and divided the pool into four strata: frontier (0 to 25% success), exploration (25 to 50%), near-mastery (50 to 75%), and exploitation (75 to 100%). Early training emphasized near-mastery and exploitation tasks to keep the reward signal reliable, and sampling shifted toward exploration and frontier tasks as success rates rose, with strata refreshed periodically. The training set contains 13,212 tasks, 45.8% RTL and 54.2% kernel.

Reward design was refined through several failures. In the first kernel pilot, correctness plus

the presence of a Triton kernel was rewarded, and the policy found cheaper paths: some responses delegated the computation to the reference framework behind a nominal wrapper, and others launched an identity kernel that copied inputs to outputs. The agent introduced hard eligibility gates based on runtime launch evidence, identity-computation detection, and framework-delegation detection, with a detected exploit receiving zero reward. A uniform gate proved too strict, because KernelBench legitimately allows partial replacement of operators, so the delegation gate was restricted to tasks whose definitions prohibit delegation. Eligible kernel responses are rewarded in tiers: full reward for a correct kernel with a speedup above 1.05 times, half reward for a correct kernel, a small reward for code that compiles but is incorrect, a smaller one for code that fails to compile, and zero for exploits.

For RTL, no calibrated analogue of speedup exists, so rewards are binary: full reward only when the official harness explicitly validates behavior, zero when the candidate is judged and fails. Two false-verdict problems drove a fail-closed rule. A self-checking testbench could exit without producing its explicit positive verdict, and treating the absence of a failure message as success rewarded executions that ended before completing their checks. Separately, malformed payloads and mismatched toolchains could fail before the candidate was evaluated, which says nothing about policy quality. Such events are routed to a bounded retry and, if unresolved, to a sentinel value outside the reward range that marks missing supervision.

The sentinel required a change in the learner. Replacing a missing reward with the group mean would shrink the estimated variance and inflate the advantages of the remaining samples, while keeping it as an ordinary reward would create a spurious negative gradient. The agent therefore masked unjudgeable trajectories out of the group statistics and the loss normalization, so that infrastructure failures cannot become gradients. A final set of experiments exposed trajectory length as a failure mode: mean reward fell from about 0.27 in the shortest length bucket to about 0.03 in the middle and 0.02 near the cap, the longest tail contained no full-reward responses, and one very long sequence caused a memory failure that reducing batch size could not fix, so the agent replaced separate prompt and response caps with a single whole-trajectory budget.

TABLE III. Post-Training Data by Stage

Stage	Training unit	Total	RTL	Kernel
SFT	Verified teacher trajectory	28,952	18,256 (63.0%)	10,696 (37.0%)
OPSD	Recoverable executable task	1,874	1,349 (72.0%)	525 (28.0%)
RLVR	Stratified verified task	13,212	6,058 (45.8%)	7,154 (54.2%)

5. EXPERIMENTAL SETUP OF THE REVIEWED REPORT

A. Model, Hardware, and Software

The base model is Qwen3.6-27B [2]. Training used the open-source reinforcement learning framework HybridFlow [25]. Evaluation ran on six nodes with eight NVIDIA H20 GPUs per node; during generation each model occupied one node and evaluation was sharded across 40 GPUs. Generation used a single-GPU backend in bfloat16 precision with a serving engine configured for a default maximum model length of 65,536 tokens. Hardware verification used the simulator version specified by each official harness, because versions differ materially in SystemVerilog support: Icarus Verilog 11.0 for ArchXBench, 12.0 for VerilogEval and RTLLM, and 13.0 for CVDP, and Verilator 5.049 for RealBench. Kernel benchmarks ran with PyTorch 2.9.1, Triton 3.5.1, and CUDA 13.0. A kernel counts as correct when every returned tensor matches the reference in shape and within a tolerance of one hundredth, and as fast when it is correct and at least 1.05 times faster than the reference.

B. Benchmarks and Metrics

Seven benchmarks were used, summarized in Table IV. Five cover RTL: VerilogEval in two formulations, RTLLM, CVDP, ArchXBench, and RealBench-Module. Two cover kernels: KernelBench, whose three levels contain 100 single-operator tasks, 100 operator-fusion tasks, and 50 full-network tasks, and TritonBench-G, which contains 184 repository-derived Triton tasks. Average-at-k reports the mean of binary success over k samples, while pass-at-k reports whether at least one of k samples succeeds. Syntax measures compilation or elaboration success,

and functional correctness requires every testbench or numerical comparison to pass. All benchmarks follow their official task definitions and harnesses.

TABLE IV. Benchmarks, Sampling Settings, and Verification Backends

Benchmark	Tasks	Sampling	Reported metric	Verification
VerilogEval (two formulations)	156	4 samples, temperature 0.8	Average at 4	Icarus Verilog simulation
RTLML	50	4 samples, temperature 0.8	Functional average at 4	Icarus Verilog simulation
CVDP	78	5 samples, temperature 0.8	Functional average at 5	Icarus Verilog with cocotb
ArchXBench v1.5	71	1 sample, temperature 0	Functional pass at 1	Self-checking testbenches
RealBench-Module	60	5 samples, temperature 0.8	Syntax and functional pass at 5	Verilator build and simulation
KernelBench	250	1 sample, temperature 0	Compiled, correct, fast	Numerical comparison
TritonBench-G	184	1 sample, temperature 0	Correctness pass at 1	Numerical comparison

C. Comparison Models and Case-Study Protocols

The main table compares iCoder against its same-size base model, a larger mixture-of-experts model, DeepSeek-V4-Pro [3], and several proprietary systems including GPT-5.5 [4], Claude Opus 4.8, and Gemini 3.5 Flash. Two exploratory case studies go beyond single-shot generation. In the RTL study, eight designs adapted from RTLScout [23] are optimized in a propose-evaluate-refine loop built on SimpleTES [24], in which a model proposes three revisions per iteration for twenty iterations. Candidates must pass the design's self-checking testbench and synthesize successfully, and are ranked by post-synthesis cell count, with lower being better. In the kernel study, three models are run in the same loop on archived GPU operator tasks, and each candidate enters the trace only if it compiled, was correct, and had a measured latency. The report states plainly that these case studies are exploratory and not controlled for token budget; iCoder used an output cap of 32,768 tokens while the comparison models used 131,072.

6. RESULTS REPORTED IN THE REVIEWED WORK

A. Main Benchmark Results

Table V reproduces a subset of the main comparison, selecting the base model, DeepSeek-V4-Pro, and the three proprietary systems named in the report. The headline gains are against the same-size base: iCoder improves RTLML functional average from 49.6 to 68.0, VerilogEval spec-to-RTL from 70.1 to 86.3, and KernelBench level 2 correctness from 28 to 74 percent, a factor of 2.64. It also edges DeepSeek-V4-Pro on RTLML (68.0 against 67.5) and nearly doubles its KernelBench level 1 correctness (61 against 32). Against proprietary systems, iCoder leads on RTLML, where GPT-5.5, Claude Opus 4.8, and Gemini 3.5 Flash score 66.0, 64.7, and 63.5, and on KernelBench level 1 correctness, where they score 43, 55, and 45. It ties Claude Opus 4.8 for the best TritonBench-G correctness at 20.1.

TABLE V. Selected Benchmark Results (Higher Is Better)

Benchmark and metric	iCoder 27B	Qwen3.6 27B	DeepSeek V4-Pro	GPT 5.5	Claude Opus 4.8	Gemi ni 3.5 Flash
VerilogEval spec-to-RTL	86.3	70.1	69.9	90.1	82.7	89.1
VerilogEval code-complete	86.0	70.8	79.8	91.4	81.9	83.8
RTL2M functional	68.0	49.6	67.5	66.0	64.7	63.5
CVDP functional	44.1	33.9	38.5	39.5	47.7	29.7
ArchXBench functional	49.3	35.2	50.7	56.3	54.9	50.7
KernelBench L1 correct (%)	61	32	32	43	55	45
KernelBench L2 correct (%)	74	28	40	41	70	78
KernelBench L2 fast (%)	40	17	25	24	37	47
KernelBench L3 correct (%)	34	12	4	38	40	58
TritonBench-G correct (%)	20.1	11.4	19.0	19.5	20.1	14.9

The table also shows where iCoder does not lead. On VerilogEval, GPT-5.5 and Gemini 3.5 Flash score well above it, on ArchXBench several larger models are ahead, and on KernelBench level 3 full-network tasks it trails the strongest systems, with Gemini 3.5 Flash reaching 58 percent against its 34.

B. Ablations of the Self-Distillation Stage

Two sets of controlled experiments support the design of the self-distillation stage. Table VI summarizes the context ablations. A task-specific plan improved average success by 6.1 percentage points over the bare prompt on the same 194 tasks, with larger gains on tasks that had failed initially. Removing verifier diagnostics from the failure history reduced the number of recovered tasks by six of 92. Fine-grained counterexamples did not improve on coarse feedback: both solved 49 of 58 paired tasks, and the difference was not statistically distinguishable. A shuffled-context control showed that the matched context raised the likelihood of the responses by about 0.18 nats, ruling out context length and formatting as sufficient explanations for the teacher advantage.

TABLE VI. Context Ablations for the Privileged View

Question	Comparison	Evaluation	Primary result
Plan value	Plan versus bare prompt	Same 194 tasks, five samples	Average at 5: 47.4% versus 41.3% (+6.1 points)
Diagnostic value	History with versus without diagnostics	92 round-0 failures	Six additional tasks solved (+6.5 points)
Feedback detail	Fine versus coarse feedback	58 paired tasks	49 of 58 solved in both conditions
Task match	Matched versus shuffled context	Same responses rescored	Mean log-likelihood about +0.18 nats

Table VII compares three objectives from the same fine-tuned checkpoint under matched data and budget: the failed gap-directed objective, an outcome-only variant, and the final variant that adds bounded teacher credit. The gap-directed objective falls below the fine-tuned starting point on every metric with an available baseline, which confirms the collapse described in

Section IV. The final variant improves on the outcome-only variant on ten of eleven reported metrics. The largest gains are on RealBench, where functional correctness rises from 6.00 to 26.70 and syntax from 14.00 to 50.00, and on ArchXBench functional pass at 1, which rises from 23.94 to 36.60. KernelBench level 1 is the one regression, from 44 to 40 percent. The authors selected the final variant on an internal validation set before these benchmark numbers were produced.

TABLE VII. Comparison of Self-Distillation Objectives from the Same Checkpoint

Benchmark and metric	SFT start	Gap-directed	Outcome-only	Outcome plus bounded credit
VerilogEval spec-to-RTL	73.88	57.37	79.33	84.70
RTL2RTL functional	60.50	40.50	59.00	61.20
ArchXBench functional	22.54	11.27	23.94	36.60
RealBench functional	9.00	4.67	6.00	26.70
KernelBench L1 correct (%)	40	29	44	40
KernelBench L2 correct (%)	53	43	56	60
KernelBench L3 correct (%)	8	4	9	14

C. Reinforcement Learning Dynamics

The report tracks validation performance across the reinforcement learning stage. Mean reward rises from 0.295 at step 20 to 0.446 at step 160 and 0.473 at step 200, and the unweighted mean success across six verification backends rises from 45.8 to 54.3 percent between steps 20 and 160. Roughly the first hundred steps account for 59 percent of the aggregate gain. Gains are largest on the most complex tasks: KernelBench rises 26.0 points and ArchXBench 12.7, while VerilogEval and RTL2RTL, which are shorter and where the model already scored 82.7 and 64.0 percent, gain 2.9 and 4.0. All six backends improve under a single reward interface with a kernel-to-RTL ratio near one, which the authors read as evidence of cross-domain compatibility.

D. Iterative Search Case Studies

In the RTL search study, mean cell-count reductions relative to each design's seed, averaged with equal weight over eight designs, are 51.1 percent for iCoder, 52.2 percent for HY3, and 55.6 percent for DeepSeek-V4-Pro. Total reported output tokens are 10.03 million, 19.75 million, and 30.45 million, so iCoder comes within 1.1 points of the first model using 51 percent of its tokens and within 4.5 points of the second using 33 percent. Against HY3 per design, iCoder records one win, five ties, and two losses.

The kernel search study is shown in Fig. 1, which plots the cumulative best speedup over the archive-designated reference implementation against search generation for eight operators. Several features are visible. For Vector Sum and Cumulative Sum, the iCoder curve ends at the top of the group, at 2.31 and 2.96 times the reference. On Grayscale Conversion all three models converge to a speedup close to 3.8 times within a few generations, and on SwiGLU every model remains near the reference, with the reported best iCoder endpoint within 1.7 percent of the fastest. On Causal Conv1D, KL Divergence, and Grayscale Conversion the reported gaps are 9.1, 4.4, and 4.8 percent. The two clearest shortfalls are Asymmetric MatMul, where the best iCoder endpoint trails the fastest by 21.8 percent, and Fused Cross-Entropy, where DeepSeek-V4-Pro reaches a speedup around 3.3 and iCoder ends well below it, a gap of 29.4 percent.

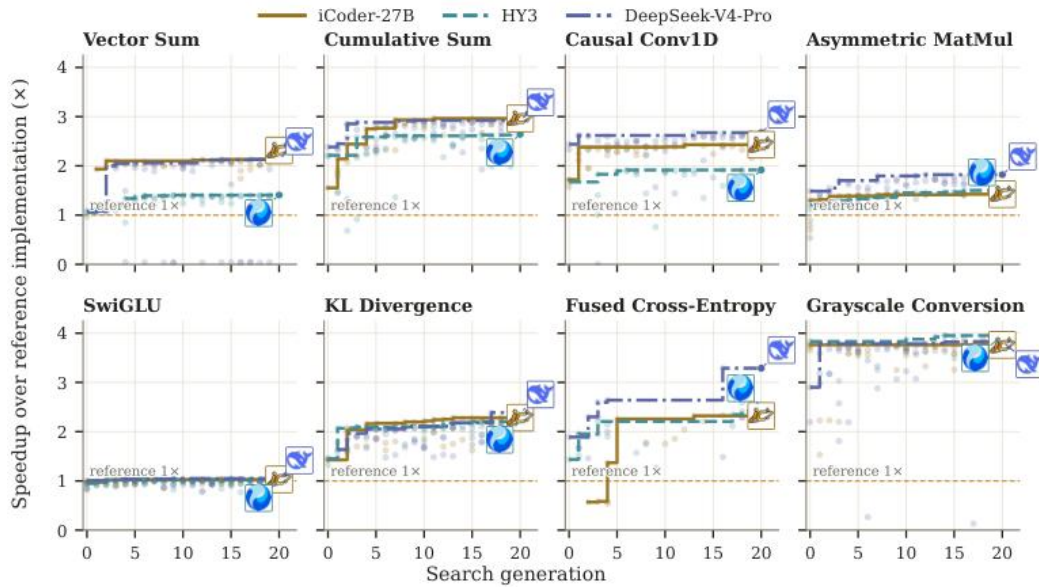


Fig. 1. Evaluation-driven search trajectories for eight GPU operators, showing the cumulative best speedup over the reference implementation against search generation for iCoder-27B, HY3, and DeepSeek-V4-Pro.

7. DISCUSSION

A. What the Evidence Establishes

The reported results support several conclusions with reasonable confidence. First, a 27-billion-parameter model post-trained with executable verification can reach or exceed the performance of much larger post systems on specific industrial coding benchmarks, notably RTLLM and KernelBench level 1, and it improves substantially over its same-size base on every reported benchmark. Second, the engineering lessons documented in the failure episodes are credible and valuable regardless of how one assigns credit: an objective that lets privileged context set the direction of learning can collapse; rewards computed from execution invite exploitation unless eligibility is scoped to the task definition; a missing verdict must be distinguished from a failed attempt and masked out of group statistics; and trajectory length should be budgeted as a whole. Third, the staged recipe, with self-distillation placed between supervised cold start and reinforcement learning, produces measurable gains in the ablation, especially on complex RTL benchmarks.

B. Reviewer Observations and Open Questions

Several aspects limit what can be inferred. The most important concerns attribution. The central question, how little human involvement suffices, is not measured directly: the report documents what humans supplied but does not vary it, and it does not run a fixed-policy control on the same scaffold, such as a human-designed recipe or a weaker agent. Without such controls it is hard to know how much of the final quality is due to the agent's decisions as opposed to the prior and to the training data. The report is candid that the model is jointly contributed by humans and the agent, and that the human prior included a research bet that self-distillation could serve as a bridge to reinforcement learning, so the agent's contribution is better described as operationalizing and repairing that bet than as inventing it.

A second observation concerns the role of the teacher. The supervised stage relies on trajectories from DeepSeek-V4-Pro, and on RTLLM iCoder's lead over that teacher is half a point, 68.0 against 67.5. That is a strong result for a compact model but not obviously a surpassing of the teacher, and it suggests that distillation from a stronger model accounts for a large part of the headline gains over the base. The ablations isolate later stages but not the supervised stage, so the relative contributions of data evolution, distillation, self-distillation, and reinforcement learning to the final scores are not fully separated in the reported experiments.

Third, statistical reporting is limited. Benchmarks such as RTLLM contain only fifty tasks, so a difference of half a point is well inside plausible sampling noise, and the main results appear to come from single runs without confidence intervals. The kernel and RTL case studies are, as acknowledged, not token-matched or hardware-controlled, use a lower output cap for iCoder,

and use a post hoc selection of operators. They are best read as evidence that the model can operate within an iterative search loop at modest token cost, not as a controlled efficiency comparison.

These observations do not diminish the contribution of documenting a full agent-led trajectory. They indicate what the next round of experiments should provide: ablations that remove the self-distillation stage and the agent-discovered objective, controlled comparisons against human-designed recipes of equal compute, repeated runs with uncertainty estimates, and graded reductions of the human prior to find the point where performance degrades.

C. Generalizing Verifier Contracts: An Industrial Monitoring Illustration

The most transferable idea in the report is the verifier contract: a specification of what a generated artifact may be, what trusted harness judges it, what counts as positive evidence of success, and how infrastructure faults are separated from model failures. The report applies the idea to RTL and kernels, but industrial software spans many other tasks. Process monitoring is one example. A typical monitoring application tracks a controlled variable, such as a temperature, against its reference and flags deviations. Fig. 2 and Fig. 3 are provided here as illustrations of oracles that such applications offer; they are not results of iCoder, they do not come from the reviewed report, and their original source should be credited by the author of this review where they are used.

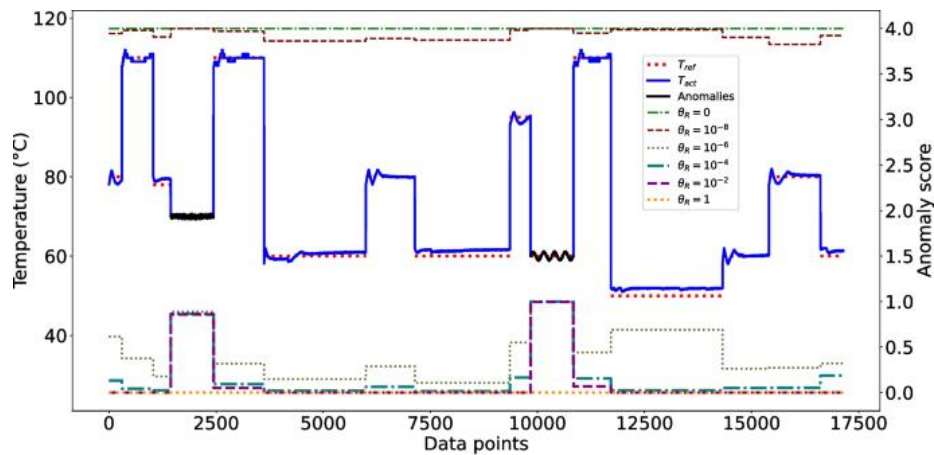


Fig. 2. Illustrative industrial monitoring example: reference and actual temperature over time with marked anomalous periods, and anomaly scores for several tolerance settings (not a result of the reviewed report).

Fig. 2 shows a temperature trace in which the actual temperature follows a stepwise reference between roughly 50 and 110 degrees Celsius over about 17,500 data points. Two periods are marked as anomalies: one in which the actual temperature stays flat at about 70 degrees while the reference moves, and one in which it oscillates around 60 degrees. The anomaly score curves on the right axis correspond to different tolerance settings. The curves for large tolerance values rise sharply to about one during the anomalous periods and stay close to zero elsewhere, whereas the curves for the zero and very small settings sit near the top of the axis nearly all the time and therefore do not discriminate, and the largest tolerance suppresses the signal entirely. The figure thus illustrates that the usefulness of an oracle depends on a calibrated threshold, which is analogous to the calibration problems the report faces when deciding whether a verdict is positive evidence or merely the absence of failure.

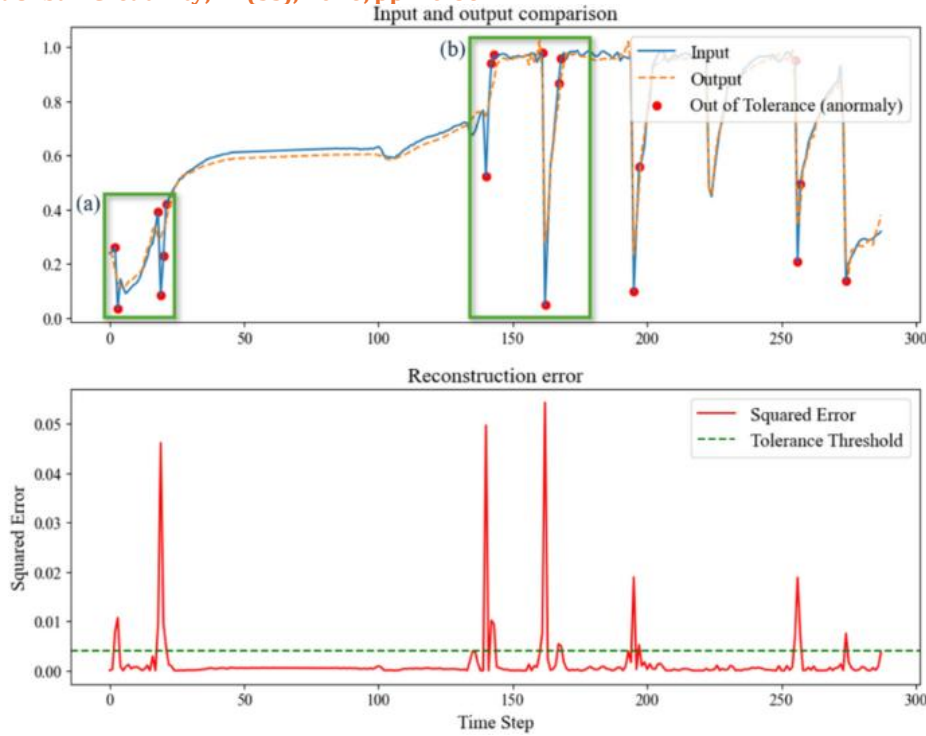


Fig. 3. Illustrative reconstruction-based anomaly detection: comparison of a normalized input with its reconstruction, with out-of-tolerance points marked, and the squared reconstruction error against a tolerance threshold (not a result of the reviewed report).

Fig. 3 shows a complementary mechanism, reconstruction-based detection. An autoencoder-style model reconstructs a normalized input signal over about 290 time steps, and points whose squared reconstruction error exceeds a tolerance threshold, here about 0.004, are marked as anomalies. The lower panel shows that the error stays near zero for most of the sequence and spikes at a handful of locations, with the largest peaks, between roughly 0.045 and 0.055, near time steps 18, 140, and 162. The highlighted regions correspond to sharp dips in the input that the reconstruction does not follow. Reconstruction models of this kind are a standard approach to anomaly detection [27], [28]. For a coding model, such an application provides an executable oracle: a generated detector can be run on labeled traces, and its detection of the known anomalous windows can be checked mechanically.

If a coding model were asked to generate monitoring code of this kind, a verifier contract in the spirit of the report would specify the interface of the generated detector, the labeled traces on which it is run, the precision and recall that count as success, and a separate state for runs that fail because of data or infrastructure problems. This is a conjecture about transfer rather than a result, and the report itself does not test it; but it indicates why the contract-and-eligibility pattern is likely to be useful beyond the two domains studied.

D. Limitations of This Review

This review has its own limitations. It relies on a single report that appeared recently and has not, to our knowledge, been independently replicated, and its numbers are taken as reported. The illustrative monitoring figures are not connected to the report's experiments, and any claims about transfer are conjectural. Readers should consult the original report for full details and should cite it, rather than this review, for the results.

8. CONCLUSION

This paper reviewed iCoder-27B, an industrial coding model whose development was led by an AI agent working inside a prior written by human experts. The report's central idea, high-density priors with low-frequency human intervention, is realized through executable research skills, a shared pool of verifier-backed RTL and kernel tasks, and a four-stage pipeline of supervised fine-tuning, on-policy self-distillation, and reinforcement learning with verifiable rewards. The reported results are strong for a compact model: 68.0 on RTLLM, 61 percent correctness on KernelBench level 1, a tie for the best TritonBench-G correctness at 20.1, and

exploratory search studies that approach larger models at a fraction of the output tokens.

The most durable contributions are probably the documented failure episodes and the design patterns they motivated: keep the direction of learning anchored in executable outcomes, scope reward eligibility to the task definition, separate failed attempts from missing verdicts, and budget whole trajectories. The strongest claim, that an agent led the development with minimal human involvement, remains only partly tested, because the report does not vary human input or compare against fixed recipes, and because several comparisons are single runs on small benchmarks or are not budget-matched. Future work should supply those controls, report uncertainty, and test whether verifier contracts transfer to other industrial software, such as monitoring and control applications, where executable oracles are available but must be calibrated with care. If those tests succeed, the path sketched by the report, in which humans distill the principles of model building and agents operationalize them through evidence-driven experimentation, will have stronger support.

References

1. Pawan Kumar Singh, Scaling Gate-All-Around (GAA) Transistor Architectures for Sub-2nm AI Accelerators Using Atomically Thin 2D Materials, Vol. 33 No. 08 (2024): JOCAAA, Journal Name: Journal of Computational Analysis and Applications, Journal's ISSN: 1521-1398 (Paper),1572-9206 (Online), <https://eudoxuspress.com/index.php/pub/article/view/5713>
2. Pawan Kumar Singh, Energy-Efficient Task Scheduling in Green Cloud Computing Using Neuromorphic Spiking Neural Networks, Vol. 33 No. 08 (2024): JOCAAA, Journal Name: Journal of Computational Analysis and Applications, Journal's ISSN: 1521-1398 (Paper),1572-9206 (Online), <https://eudoxuspress.com/index.php/pub/article/view/5712>
3. Rangavinay Teja Royal Jagga, 2026, DOI: 10.1109/GCWCN66157.2025.11448495, Date of Conference: 22-23 November 2025, Date Added to IEEE Xplore: 25 March 2026, <https://ieeexplore.ieee.org/document/11448495>
4. Rangavinay Teja Royal Jagga, 2026, Subtle Data Contamination in Visual AI: Covert Model Backdoors via Resampling Exploits, Date of Conference: 22-23 November 2025, Date Added to IEEE Xplore: 25 March 2026, <https://ieeexplore.ieee.org/document/11448514>
5. Rangavinay Teja Royal Jagga, 2026, Robotic Tactile-Vision Integration for Interactive Object Disentanglement, <https://ieeexplore.ieee.org/document/11448337>
6. Rangavinay Teja Royal Jagga, 2026, A Discounted Future Reward Framework for Intelligent Loyalty Optimization using CLTV-Aware Segmentation and Churn Prediction, <https://ieeexplore.ieee.org/document/11497560>
7. Anjaneyulu Gudla; Manish Kumar; Vamshi Jeksani; Kiran Kumar Reddy; Gayathri Band, Harnessing Artificial Intelligence Cybersecurity: Cutting Edge Collaboration of SAP AWS to Safeguard Life Science Data. 2026 <https://doi.org/10.1109/aiei69164.2026.11497833>
8. G. Mahender, Department of CSE(Data Science), Vardhaman College of Engineering ; Vamshi Jeksani; Manish Kumar; Koppuravuri Sri Lakshmi Sruthi; Kiran Kumar Reddy, SAP Cloud System on AWS for Risk Detection and Integration Artificial Intelligence Scalable Cybersecurity 2026 <https://doi.org/10.1109/aiei69164.2026.11497435>
9. Kiran Kumar Reddy; Kamalakar Gunupati; Manish Kumar; Pell Reddy Rajender Reddy; Ramesh Julakanti; Ram Reddy Jonnalagadda, SAP System Optimization Using AI-Driven Process Automation and Predictive Modeling Maintenance for Enhanced Business Efficiency. 2025 <https://doi.org/10.1109/computingcon64838.2025.11376613>
10. Amilineni, M.V. 2025. Audit-Safe Agentic RAG Framework for Regulated Enterprise Systems: A Trustworthy AI Architecture for SAP, Finance, Healthcare, and Government Workflows. INTERNATIONAL JOURNAL OF ADVANCES IN SIGNAL AND IMAGE SCIENCES. (Jan. 2025), 34–46. DOI: <https://doi.org/10.29284/7fjzk883>
11. Pawan Kumar Singh, ACCELERATING LARGE LANGUAGE MODEL TRAINING USING HYBRID QUANTUM-CLASSICAL VARIATIONAL AUTOENCODERS WITH INTEGRATED PHOTONIC QRAM, Vol. 54 No. 3 (2026): July-September 2026, Power System Protection and Control, ISSN-1674-3415, <https://pspac.info/index.php/dlbh/article/view/501>, DOI: <https://doi.org/10.46121/pspc.54.3.45>
12. Pawan Kumar Singh, RESILIENCE ASSESSMENT OF MODERN POWER GRID INFRASTRUCTURE AGAINST EXTREME WEATHER USING PHYSICS-INFORMED NEURAL NETWORKS, Vol. 54 No. 1 (2026): January-March 2026, Power System Protection and Control, ISSN-1674-

- 3415, <https://pspac.info/index.php/dlbh/article/view/502>, DOI: <https://doi.org/10.46121/pspc.54.1.64>
13. Pawan Kumar Singh, MITIGATING THE EXABYTE STORAGE CRISIS: ENHANCING READ/WRITE EFFICIENCY IN 5D OPTICAL GLASS STORAGE USING SILVER NANOPARTICLE DOPING, Vol. 38 No. 11s (2025), International Journal of Applied Mathematics (IJAM) , pISSN 1311-1728, eISSN 1314-8060, <https://ijamjournal.org/ijam/publication/index.php/ijam/article/view/2153>
 14. Mr. Pawan Kumar Singh (Author), Miss. Pari Nidhi Singh (Author) ,Intelligent Bot, ASIN : B0DBGCQY8X, Publisher : Namya Press, Publication date : November 25, 2022, Language : English, Print length : 105 pages, ISBN-10 : 9390124867, ISBN-13 : 978-9390124862, Item Weight : 7.5 ounces, Dimensions : 6.24 x 0.43 x 9.24 inches, https://www.amazon.com/Intelligent-Bot-Pawan-Kumar-Singh/dp/B0DBGCQY8X/ref=sr_1_5?crid=2VAXN5BRN5SMS&dib=eyJ2IjoiMSJ9.v3OCQvHBpy2ORlJfMWh0qbOI-RLOIUJIC8Xa0Pjeqilvo2sbZ_suN3LbSCw_OA_ZD_4lgjRLMGlBwZQQ1ehZEfDpt_MjiOUfVuqqzMa v0I5_mb6pN5DQog0U7SA826n1DQQ3QAtWwopHy4lC5IfhZXttKj9faiyATwhdtDATSSV1-r7FmzM2VQyIA0jxcfq6E6hp9ECfH1fby0li9kUfGpLxMq4GktlzUIJ05ggrhsQ.G1AjQiVqU4HFsnc zVz-X6NXCNs3EFSn-RI8mcWWG0c&dib_tag=se&keywords=Pawan+Kumar+Singh%2C+book&qid=1787815741&s prefix=pawan+kumar+singh%2C+bo%2Caps%2C450&sr=8-5
 15. Mr. Pawan Kumar Singh (Author), STORAGE CRISES DUE TO RISE OF AI: A Comprehensive Guide to Managing the Storage Challenges in the Age of Artificial Intelligence, ASIN : B0HFK12ZCT Publisher : EB1 Profile, Accessibility : Learn more, Publication date : August 17, 2026, Edition : 1st
Language : English, File size : 1.7 MB, Screen Reader : Supported, Enhanced typesetting : Enabled
X-Ray : Not Enabled, Word Wise : Not Enabled ,Print length : 261 pages ,ISBN-13 : 979-8951479112, Page Flip : Enabled, https://www.amazon.com/STORAGE-CRISES-DUE-RISE-Comprehensive-ebook/dp/B0HFK12ZCT/ref=sr_1_1?crid=9MF6C4WDVJR5&dib=eyJ2IjoiMSJ9.DQ8fesBch7ELmRnildt-YA.NBSCHLmetvnbhkwIpOoj3s0IVsCV8y6-ZCfCP6U1c&dib_tag=se&keywords=Pawan+Kumar+Singh%2C+book%2C+eb1+Profile&qid=1787816040&sprex=pawan+kumar+singh%2C+book%2C+eb1+profile%2Caps%2C420&sr=8-1
 16. Parag Bhartkumar Parikh, FRAMEWORKS FOR MITIGATING SUPPLY CHAIN DISRUPTIONS AND MATERIAL PRICE VOLATILITY IN SUSTAINABLE BUILDING CONSTRUCTION, Vol. 53 No. 2 (2025): April-June 2025 , , Power System Protection and Control, ISSN-1674-3415, <https://pspac.info/index.php/dlbh/article/view/447>, DOI: <https://doi.org/10.46121/pspc.53.2.34>
 17. Parag Bhartkumar Parikh, IMPLEMENTING DIGITAL INSPECTION PROTOCOLS FOR ENHANCED QUALITY CONTROL MANAGEMENT IN HIGH-RISE CONSTRUCTION, Vol. 54 No. 3 (2026): July-September 2026, Power System Protection and Control, ISSN-1674-3415, <https://pspac.info/index.php/dlbh/article/view/439>, DOI: <https://doi.org/10.46121/pspc.54.3.12>
 18. Parag Bhartkumar Parikh , EVALUATING THE FEASIBILITY AND EFFICIENCY OF 3D-PRINTED CONCRETE TECHNOLOGY IN SUSTAINABLE MODULAR CONSTRUCTION, Power System Protection and Control, ISSN-1674-3415, <https://pspac.info/index.php/dlbh/issue/view/30>, DOI: <https://doi.org/10.46121/pspc.54.1.61>
 19. Parag Bhartkumar Parikh, EVALUATING THE FEASIBILITY AND EFFICIENCY OF 3D-PRINTED CONCRETE TECHNOLOGY IN SUSTAINABLE MODULAR CONSTRUCTION, Vol. 54 No. 1 (2026): January-March 2026, System Protection and Control, ISSN-1674-3415, <https://pspac.info/index.php/dlbh/article/view/440>, DOI: <https://doi.org/10.46121/pspc.54.1.61>
 20. Ronakkumar shah, REDEFINING BUSINESS PROCESS MANAGEMENT: THE SYNERGY OF AI AND INTELLIGENT AUTOMATION, Vol. 54 No. 3 (2026): July-September 2026, System Protection and Control, ISSN-1674-3415, <https://pspac.info/index.php/dlbh/article/view/448>, DOI: <https://doi.org/10.46121/pspc.54.3.14>

21. Ronakkumar shah, OPERATIONAL ACCOUNTABILITY IN ACTION: METRICS AND MECHANISMS FOR SYSTEMIC SUCCESS, Vol. 53 No. 4 (2025): October-December 2025, System Protection and Control, ISSN-1674-3415, <https://pspac.info/index.php/dlbh/article/view/452>, DOI: <https://doi.org/10.46121/pspc.53.4.40>
22. Ronakkumar shah, BEYOND THE BLUEPRINT: AUTOMATED WORKFLOW ENFORCEMENT AND POLICY GOVERNANCE, Vol. 53 No. 2 (2025): April-June 2025 , System Protection and Control, ISSN-1674-3415 ,<https://pspac.info/index.php/dlbh/article/view/451>, DOI: <https://doi.org/10.46121/pspc.53.2.35>
23. Tushita Agarwal; Swaminathan Sethuraman, Mapping Customer Behavior: Visual Analysis of Online Retail Interactions, Date Added to IEEE Xplore: 17 February 2026, DOI: <https://doi.org/10.1109/IC2NC67409.2025.11376261>
24. Tushita Agarwal, Efficient Data Summarization and Comparison via Grid-Based Statistical Proxies for Large-Scale Datasets, Date Added to IEEE Xplore: 03 December 2025, DOI: <https://doi.org/10.1109/ICBATS66542.2025.11258564>
25. Dip Bharatbhai Patel, Building scalable business intelligence systems in the cloud: A technical approach, International Journal of Science and Research Archive, 2021, 02(01), 212-215., Received on 03 February 2021; revised on 17 April 2021; accepted on 19 April 2021,DOI: <https://doi.org/10.30574/ijrsra.2021.2.1.0047>
26. Dip Bharatbhai Patel, University of North America, Virginia, USA, Home / Archives / Vol. 5 No. 02 (2025): Volume 05 Issue 02 / Articles, <https://www.academicpublishers.org/journals/index.php/ijdsml/article/view/5958>
27. Sai Akshara Vemuri , KINEMATIC OPTIMISATION OF SOFT ROBOTIC MANIPULATORS FOR MEDICAL INTERVENTION, Vol. 54 No. 3 (2026): July-September 2026, System Protection and Control, ISSN-1674-3415, <https://pspac.info/index.php/dlbh/article/view/450>
28. Sai Akshara Vemuri, <https://www.periodicos.ulbra.org/index.php/acta/article/view/528>
29. Aziz, A., Chandusha, K., Naga Kiran, B., Prasad, M. L. M., Golla, K., Rajeswari, S., & Katta, S. K. R. (2026). Exploring Machine Learning Applications for Genomic Data Analysis in Personalized Medicine. International Journal of Drug Delivery Technology, 16(38s), 1166-1173. <https://doi.org/10.25258/ijddt.16.38s.127>,<https://www.periodicos.ulbra.org/index.php/acta/article/view/687>
30. Mohammad Naffizuddin , Chaganti Ashok ,Sowjanya Gunukula , Rvbs Sarma , Bhavana Sujanamulk , Bharani Krishna Takkella , Chukka Ram Sunil ,Varri Sujana , Ashwini Kumar, Etiological Factors of Orofacial Clefts in a Hospital-Based Population: A Descriptive Analytical Study, Published: December 10, 2025,DOI: 10.7759/cureus.98868,DOI : <https://doi.org/10.7759/cureus.98868>
31. Manoj Kumar Kagitha, Automated Security Compliance in Ci/Cd: A Natural Language Processing (NLP) Approach to Detecting Vulnerabilities in Infrastructure-As-Code (Terraform/Bicep), Publication year 2021, Volume 2021, Issue 12, <https://computerfraudsecurity.com/index.php/journal/article/view/970>
32. Manoj Kumar Kagitha, Hybrid Quantum-Classical Resource Scheduling: A Proposed Architecture for Next-Generation Hyperscale Data Centers, Publication year 2024, Volume 2024, Issue 12, <https://computerfraudsecurity.com/index.php/journal/article/view/969>
33. Manoj Kumar Kagitha, Self-Healing Cloud Infrastructure: A Hybrid Reinforcement Learning Framework for Predicting Microservice Failures in Azure Kubernetes Service (AKS),Publication year 2023, Volume 2023, Issue 12,<https://computerfraudsecurity.com/index.php/journal/article/view/971>
34. Manoj Kumar Kagitha, GREENOPS IN THE CLOUD: AN AI-DRIVEN TELEMETRY ANALYSIS MODEL FOR REDUCING CARBON FOOTPRINT IN HIGH-AVAILABILITY CLUSTERS, Vol. 28 No. 6 (2020): JOCAAA, <https://www.eudoxuspress.com/index.php/pub/article/view/5054>
35. Lakshmi Rahul Reddy Mareddy. (2026). Security and Performance Assessment of Encryption Techniques for Cloud Platforms. International Journal of Communication Networks and Information Security (IJCNIS), 18(1), 68–80. Retrieved from <https://ijcnis.org/index.php/ijcnis/article/view/8735>
36. Lakshmi RahulReddy Mareddy, Context-Aware Learning Approaches for Improving Prediction Accuracy in Dynamic Systems, International Journal of Computational Mathematical Ideas (IJCMI), Vol.17 No.3(2025): IJCM I Vol17 NO3 2025 <https://www.ijcml.com/index.php/journal/article/view/15>
37. Lakshmi Rahul Reddy , Mareddy. (2026). Revisiting Autonomy in Artificial Intelligence through Agentic System Design. Journal of Computational Analysis and Applications (JoCAAA), 35(1),

- 707–713. <https://eudoxuspress.com/index.php/pub/article/view/4780>
38. Mahendrakumar Kalal, Lean-Driven SAP Production Planning Optimization Using Machine Learning for Inventory and Throughput Efficiency, Date of Conference: 19-20 March 2026, Date Added to IEEE Xplore: 08 April 2026, ISBN Information:, ISSN Information: DOI:10.1109/ISDFS69419.2026.11459029, <https://ieeexplore.ieee.org/abstract/document/11459029>
39. Mahendra kumar Kalal, SUPPLY CHAIN RESILIENCE THROUGH INTEGRATED SAP PRODUCTION PLANNING – PROCUREMENT SYNCHRONIZATION, Archives / Vol. 54 No. 1 (2026): January-March 2026 / Articles, <https://pspac.info/index.php/dlbh/article/view/259>
40. Dr Manoj Goel (December 2018). THE ATTITUDE OF DIFFERENT STATES STUDENTS TOWARDS PHYSICAL EDUCATION; STUDYING IN PHYSICAL EDUCATION COLLEGES OF MAHARSHTRA STATE, <https://ijeponline.com/index.php/journal/article/view/965/847>
41. Ilakiya Ulaganathan, Tagore Engg. College, Anna University, Chennai, India, ; FNU Antara; Santhosh Chitraju Gopal Varma; Zokir Mamadiyarov; Ziyadulla Turayev, Published in: 2025 International Conference on Sustainability, Innovation & Technology (ICSIT), Date of Conference: 22-23 August 2025, Date Added to IEEE Xplore: 19 December 2025, ISBN Information: <https://ieeexplore.ieee.org/document/11294748/references#references>
42. Shreyas Kasture, Sudhanshu Maurya, Amit Kumar Sharma, Santhosh Chitraju Gopal Varma, Kashish Mirza, Firdous Sadaf Mohammad Ismail, DOI: <http://doi.org/10.11591/ijai.v15.i2.pp1746-1761>, IAES International Journal of Artificial Intelligence (IJ-AI), ISSN/e-ISSN 2089-4872/2252-8938 <https://ijai.iaescore.com/index.php/IJAI/article/view/26932>
43. Shiv Shankar Dwivedi, Mr. Santosh Kumar Sharma, Future-Proofing Data Security with Quantum-Resistant Cryptography, Vol. 11 No. 3 (2025): IJCESEN, DOI: <https://doi.org/10.22399/ijcesen.3447>, <https://ijcesen.com/index.php/ijcesen/article/view/3447>
44. Vijender Kalmotia, Enhancing statistical accuracy in gene perturbation studies, Volume 266, August 2026, 105819, <https://doi.org/10.1016/j.biosystems.2026.105819>, <https://www.sciencedirect.com/science/article/pii/S0303264726001292>
45. Naveen Undrathi, Design and implementation of generative AI-driven data transformation pipelines using spring boot for scalable financial applications, eISSN: 2581-9615 || CODEN: WJARAI || Impact Factor 8.2 || CrossRef DOI, September 2026 (Volume 31, Issue 3), DOI url: <https://doi.org/10.30574/wjarr.2026.30.3.1629>,
46. Naveen Undrathi, Agentic generative AI-enabled microservices framework for intelligent financial data processing in java-based enterprise systems, September 2026 (Volume 31, Issue 3), eISSN: 2581-9615 || CODEN: WJARAI || Impact Factor 8.2 || <https://doi.org/10.30574/wjarr.2026.30.3.1628>
47. Akshay Jain, Comparative Analysis of Linkage Methods in Hierarchical Clustering for Customer Segmentation, Conference paper, First Online: 07 October 2025, https://link.springer.com/chapter/10.1007/978-981-96-7289-9_44
48. Ismail Hossain, Department of Computer Science, George Mason University, Fairfax, VA, USA, Mridul Banik, Department of Computer Science, Colorado State University, Fort Collins, CO, USA, <https://ieeexplore.ieee.org/abstract/document/11446534> „DOI: 10.1109/FMLDS67896.2025.00106
49. Ismail Hossain, Mridul Banik , Fahmid Al Farid , Jia Uddin , Hezerul bin Abdul Karim , Advancing Large Language Models for Low-Resource Languages: A Systematic Review of Pretraining, Adaptation, and Ethical Challenges, 10.32604/cmcs.2026.075507, Date: 2026 <https://www.sciencedirect.com/org/science/article/pii/S152614922600322X>
50. Ismail Hossain; Mridul Banik, Bridging Visual and Linguistic Intelligence for Chest X-rays: A Comprehensive Review of ViTs and LLM Synergies, Published in: 2025 IEEE International Conference on Future Machine Learning and Data Science (FMLDS) Date of Conference: 02-05 November 2025, Date Added to IEEE Xplore: 25 March 2026, ISBN Information: <https://ieeexplore.ieee.org/abstract/document/11446534>
51. Rifat Sarker Aoyon¹, Ismail Hossain², M. Abdullah-Al-Wadud³, Jia Uddin⁴, A Secured and Continuously Developing Methodology for Breast Cancer Image Segmentation via U-Net Based Architecture and Distributed Data Training, <https://www.sciopen.com/article/10.32604/cmcs.2025.060917>

52. M. B. Vibha, M. Chandrika, Samitha Khaiyum & P. Rakshitha Kiran, Predicting sedentary behavior in adults using stacked LSTM modeling, published: 27 November 2024
Volume 16, pages 346–355 (2025), <https://link.springer.com/article/10.1007/s13198-024-02622-2>
53. Bellary Chiterki Anil, Jayasimha Sondekoppa Rajkumar, T. L. Divya, Samitha khaiyum Rakshitha Kiran P., Balakrishnan Ramadoss, A Radiomics-based Framework for Liver Cancer Analysis using Explainable Artificial Intelligence (XAI) Methods, Vol. 15 No. 3 (2025): June, 2025, <https://etasr.com/index.php/ETASR/article/view/10377>
54. Jayminkumar Patel, Sanjaykumar patel, Birju Patel, Rajan shah, Development and Validation of a Stability-Indicating Analytical Method for Quantitative Determination of Selected Anticancer Drugs in Pharmaceutical Dosage Form, Archives / Vol. 35 No. S2 (2026): Apr-jun 2026, <https://welltestingjournal.com/index.php/WT/article/view/288>
55. Jayminkumar Patel, Sanjaykumar patel, Birju Patel, Rajan shah, Artificial Intelligence-Guided Optimization of Chromatographic Methods for Pharmaceutical Analytical Development, Archives / Vol. 35 No. S2 (2026): Apr-jun 2026, <https://welltestingjournal.com/index.php/WT/article/view/294>
56. Sharoon Sharif, Inikpi Egbunu, Yujian Fu, and Zhigang Xiao. 2026. Hybrid Multi-Modal Learning for Extreme Long-Tail Fungi Classification. In 2026 ACM Southeast Conference (ACMSE 2026), April 23–25, 2026, Troy, AL, USA. ACM, New York, NY, USA, 38–45. <https://doi.org/10.1145/3746467.3801502>