Can LLMs Compress (and Decompress)? Evaluating Code Understanding and Execution via Invertibility

Nickil Maveli, Antonio Vergari, Shay B. Cohen

University of Edinburgh

Dataset and Code

We present RoundTripCodeEval (*RTCE*) to evaluate whether LLMs maintain consistent reasoning across *forward* and *backward* code execution. Despite testing SOTA models with zero-shot prompting, fine-tuning, and self-reflection, all approaches yield only modest improvements

Contributions

NEW BENCHMARK 🗂️

- 250 inputs per algorithm across four lossless compression algorithms, assessed via exact-match bijection fidelity
- No execution required

FOUR TASKS 🔄

- Forward and inverse variants of output & input prediction form a closed loop
- Test bidirectional code reasoning in a single coherent framework

THREE PARADIGMS 🧪

- Zero-shot prompting, SFT on traces & iterative self-reflection
- None closes the round-trip gap, revealing a fundamental reasoning deficit

Motivation

🔁 **Passing forward ≠ understanding** High accuracy in one direction can be achieved through pattern matching alone, with no genuine understanding of what the code actually does

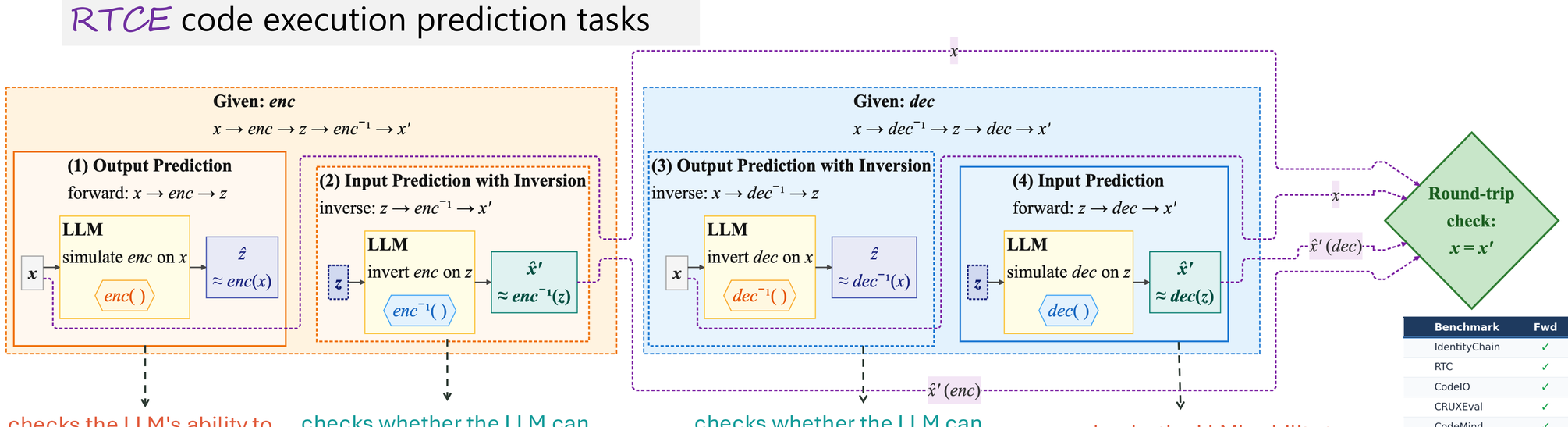
⚠️ **Existing benchmarks are blind to this failure** LLMs can pass both forward or backward directions independently while holding mutually contradictory internal representations, a critical flaw no existing benchmark can detect

🫧 **The round-trip test reveals the truth** If a model truly understands a function, it should be able to run it both ways

RTCE statistics

Data Source	# Cat.	Key Characteristics
Patterned String Data	15	• Repeated chars, alternating/block patterns, palindromes • Pangrams, keyboard sequences, pseudo-random token insertions • Short NL sentences (single & repeated) • Mix of highly compressible patterns and high-entropy noise
Structured Log Data	11	• Apache-style logs: timestamp, level, HTTP method, endpoint • Synthetic user ID, request duration, IP address • Slow requests, auth events, DB errors, severity levels • Metadata provided for downstream analysis
YAML-Like Config Data	10	• App configs, Kubernetes, Docker Compose, Helm values • Ansible, Prometheus, GitHub Actions, CircleCI • CloudFormation & Terraform snippets • Preserves indentation, key–value syntax, multi-doc formats
Tabular Data	10	• CSV/TSV tables (5×5): numeric, alphanumeric, mixed types • Sparse & repeated-header variants • Variable delimiter, value entropy, and sparsity • Captures real-world spreadsheet/log regularity

RTCE code execution prediction tasks



checks the LLM's ability to simulate the compression process

checks whether the LLM can internally reinterpret the decoder as its inverse encoder to reconstruct the input

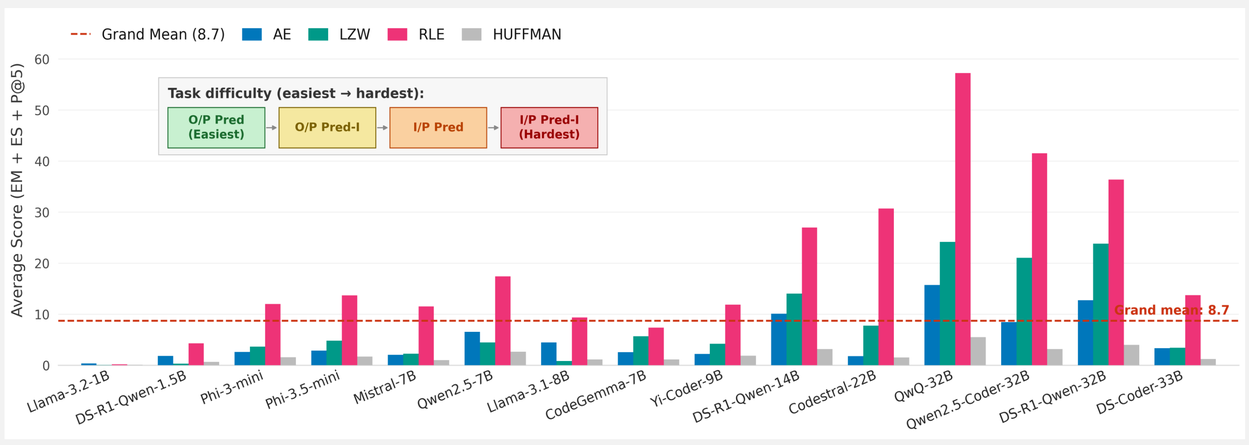
checks whether the LLM can internally reinterpret the encoder as its inverse decoder to obtain the compressed output

checks the LLM's ability to simulate the decompression process

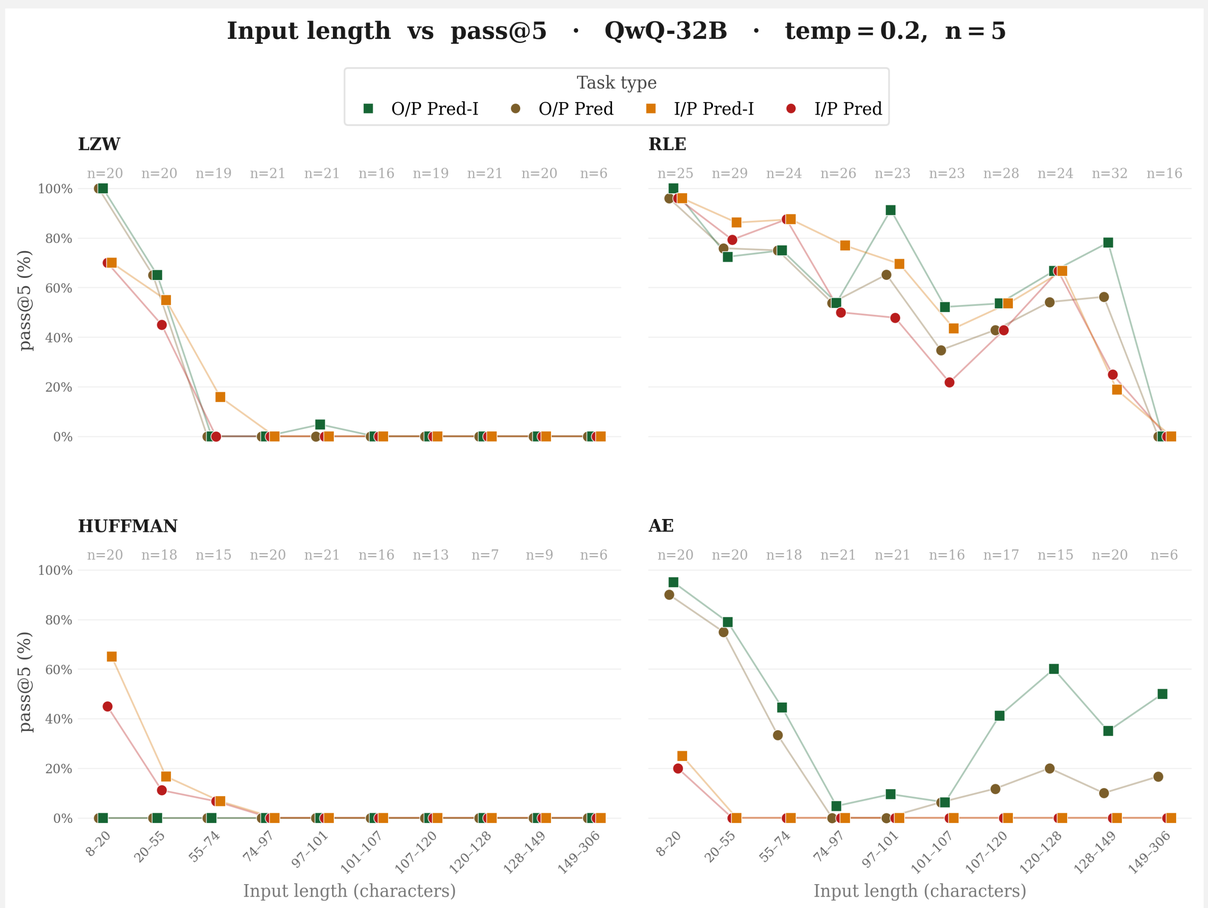
RTCE v/s other benchmarks

Benchmark	Fwd	Bwd	RT	Invert?	Exec?	Comparison Level
IdentityChain	✓	✓	Partial	✗	✗	Spec ↔ Code
RTC	✓	✓	Strict	✗	✗	Code ↔ NL descriptions
CodeIO	✓	✓	No	✗	✓	Function I/O
CRUXEval	✓	✓	No	✗	✓	Function exec I/O
CodeMind	✓	✗	No	✗	✓	Task-level exec reasoning
ExeRScope	✓	✗	No	✗	✓	Feature-level exec reasoning
REVAL	✓	✗	No	✗	✓	Runtime behavior
CACP	✓	✗	No	✗	✓	Concept-level traces
RTCE (ours)	✓	✓	Strict	✓	✓	**Exact enc/dec I/O bijection**

(1) How well can LLMs invert code in a zero-shot setting, relying solely on their parametric knowledge?



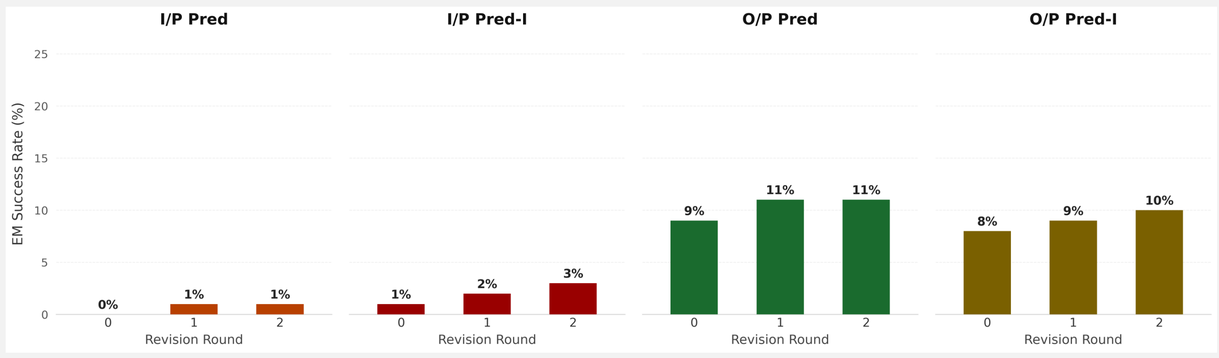
Input length vs. pass@5: Longer inputs consistently degrade Pass@5, confirming input complexity as a primary difficulty driver



- Models succeed on isolated forward and backward tasks, but fail the combined round-trip
- Reasoning training doubles round-trip accuracy: QwQ-32B (15.71) vs Qwen2.5-Coder-32B (8.45), same size, same tokeniser

(2) Can iterative self-reflection recover LLM performance beyond what zero-shot prompting achieves?

- Reflection helps once and multi-turn revision plateaus after a single round
- Failures reflect broken code logic, not arithmetic slips



(3) Can fine-tuning on execution traces give LLMs the ability to invert code?

- Fine-tuning on step-by-step execution traces still fails to close the round-trip loop
- Failure reflects an inability to internalise bijective logic, not a lack of memorised steps

Algo	Temp	I/P Pred	I/P Pred-I	O/P Pred	O/P Pred-I
AE	0.2	30.77	23.08	**78.57**	**84.62**
	0.8	15.00	20.00	70.00	**84.21**
HUFFMAN	0.2	35.00	50.00	0.00	0.00
	0.8	36.36	50.00	0.00	0.00
LZW	0.2	62.50	62.50	**87.50**	**87.50**
	0.8	**77.42**	**75.00**	**78.13**	**80.65**
RLE	0.2	**76.47**	**86.00**	**80.00**	**86.00**
	0.8	**80.65**	**86.89**	**80.33**	**86.89**