KernelCraft

Benchmarking for Agentic Close-to-Metal Kernel Generation on Emerging Hardware

Jiayi Nie, Haoran Wu, Yao Lai, Zeyu Cao, Cheng Zhang, Binglei Lou, Erwei Wang, Jianyi Cheng, Timothy M. Jones, Robert Mullins, Rika Antonova, Yiren Zhao

Background

- New AI accelerators come with custom instruction sets (ISAs)
- Kernels: low-level programs that run one computation on the hardware
- Programmability wall
- LLM kernel generation works in mature ecosystems

Limitations of prior work

- LLM kernel generation works in mature ecosystems
- Existing benchmarks do not test new hardware
- Can an agent write correct, fast kernels for an unseen ISA from docs and feedback alone?

Overview

- Benchmark for agent-written assembly kernels
- Inputs to the agent
  - Task (e.g. "implement attention").
  - ISA documentation.
  - Hardware config (memory, array sizes).
- Agent: an LLM calling tools natively
- Output: an assembly kernel, checked for correctness and cycles
- 3 accelerators + 1 CPU, 33 tasks × 5 configs

- Agent gets a task, ISA manual and hardware configuration
- KernelCraft provides tools the LLM agent can call
- Agent outputs assembly
- Output checked for correctness

Tools

Tool	Input	Output	Description
`write_code`	assembly code, workload	success, line count	Save assembly code as a file
`check_syntax`	workload	success, errors, instruction count	Validate syntax and compile
`run_evaluation`	workload	match rate, latency	Evaluate correctness and performance
`view_output`	workload	value comparison statistics	Compare actual vs expected output
`get_instruction_size`	workload	instruction counts by category	Count instructions by category
`grep_docs`	query	matched documentation	Search ISA and hardware documentation

Tasks

ID	Category	Task	Description
		Level 1: Primitive Operations	
1	Activation	SiLU	$x \cdot \sigma(x)$
2		ReLU	$\max(0, x)$
3		GELU	$x \cdot \Phi(x)$
4	Normalization	Softmax	$e^{x_i} / \sum_j e^{x_j}$
5		LayerNorm	$(x - \mu)/\sqrt{\sigma^2 + \epsilon}$
6		RMSNorm	$x/\text{RMS}(x)$
7	Matrix	GEMV	$y = Ax$
8		GEMM	$Y = XW$
9		BatchMatMul	$Y_i = X_i W_i$ for batch i
10		Linear	$Y = XW + b$
11	Spatial	Conv2D	2D convolution
12		DepthwiseConv	Per-channel 2D convolution
		Level 2: Composite Operations	
13	Encoding	RoPE	Rotary position encoding
14	MLP	FFN	Linear, SiLU, Linear
15		SwiGLU	(Linear $\odot$ SiLU(Linear)), Linear
16	Attention (core)	ScaledDotProduct	$\text{softmax}(QK^\top/\sqrt{d})V$
17		FlashAttention	Tiled attention with online softmax
18	Attention (+proj)	MHA	Multi-head attention with projections
19		GQA	Grouped K/V heads
20		MQA	Single shared K/V head
		Level 3: End-to-End System	
21	CNN	ConvBlock	Conv2D, BatchNorm, ReLU
22	Transformer	DecoderBlock(LLaMA-style)	RMSNorm, self-attn, SwiGLU
23		DecoderBlock(T5-style)	RMSNorm, self-attn, cross-attn, FFN

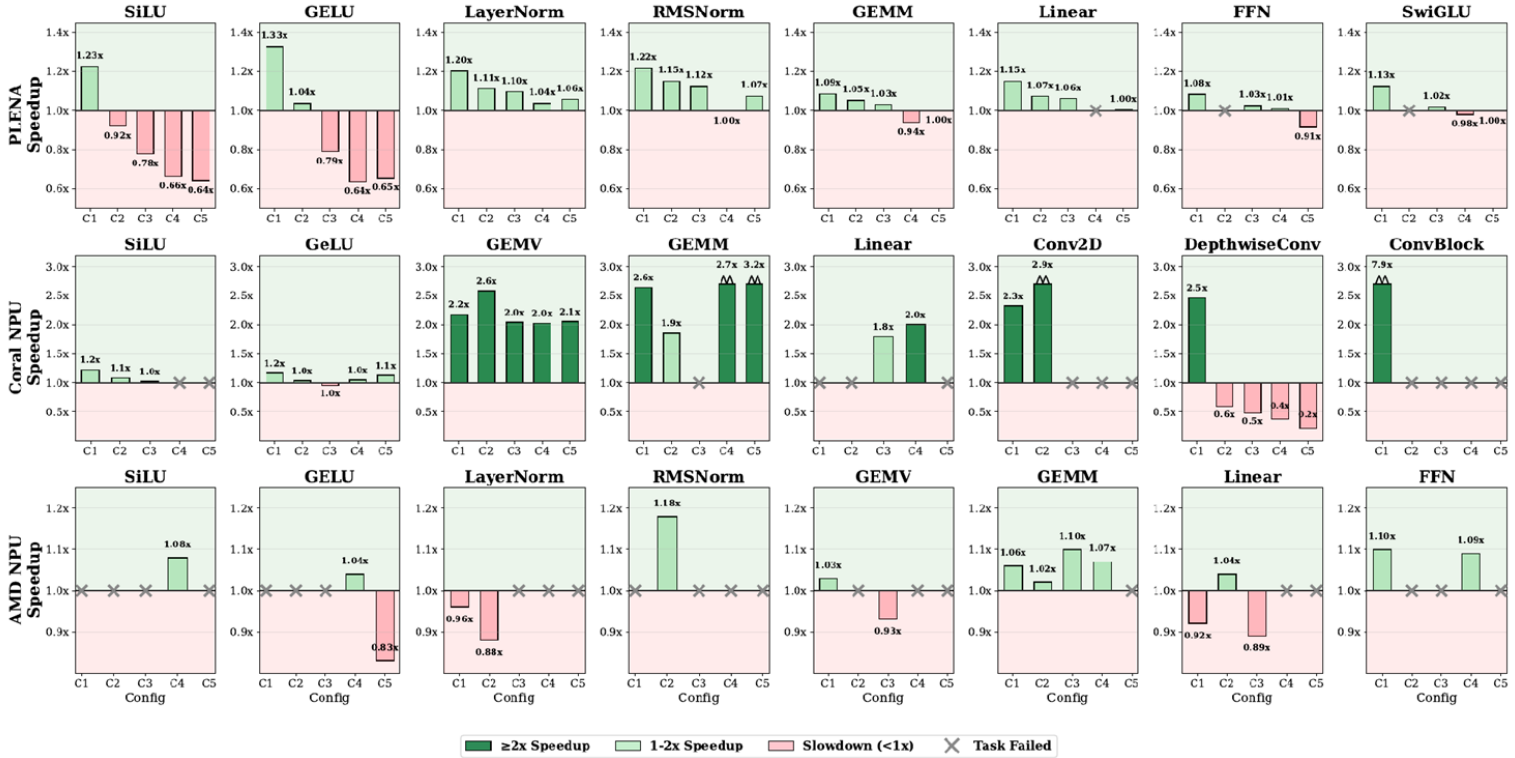
Task	Description	Succ.	Iter.	Speedu
	Level 1: Primitive Operations (max 10 iterations)			
multiply	Shift-and-add multiplication	✓	3	1.04›
vvadd	Vector-vector add	✓	5	1.10›
copy	Copy memory block	✓	8	1.50›
median	3-element median filter	✓	10	1.03›
dotprod	Dot product	✓	8	1.01›
Level 1 Subtotal		*5/5*		
	Level 2: Algorithmic Operations (max 10 iterations)			
qsort	Quicksort algorithm	✓	9	1.63›
rsort	Radix sort algorithm	✓	6	0.12›
towers	Towers of Hanoi	✓	8	1.18›
Level 2 Subtotal		*3/3*		
	Level 3: End-to-End System (max 15 iterations)			
dhrystone	Mixed integer operations	✓	12	0.93›
pmp	Physical Memory Protection	✓	8	1.68›
Level 3 Subtotal		*2/2*		

Hardware	ISA	Compiler	Backend
PLENA	Custom ISA	PLENA compiler	PLENA Simulator
AMD NPU	Custom ISA	Peano	NPU Hardware
Coral NPU	RISC-V + RVV	RISC-V Compiler	Verilator
BOOM	RISC-V	RISC-V Compiler	Verilator

Evaluation: success rate

ID	Task	PLENA				AMD NPU				Coral NPU			
		GPT-5.2	Gemini-3-flash	Sonnet 4	DeepSeek R1	GPT-5.2	Gemini-3-flash	Sonnet 4	DeepSeek R1	GPT-5.2	Gemini-3-flash	Sonnet 4	DeepSeek R1
Level 1: Primitive Operations (Max 15 iterations)													
1	SiLU	**5/5**	**5/5**	**2/5**	0/5	**1/5**	**1/5**	0/5	0/5	**3/5**	**3/5**	0/5	0/5
2	ReLU	**2/5**	0/5	**1/5**	0/5	**2/5**	**1/5**	0/5	0/5	**5/5**	**4/5**	**1/5**	0/5
3	GELU	**4/5**	**4/5**	**1/5**	0/5	**1/5**	**2/5**	0/5	0/5	**5/5**	**5/5**	0/5	0/5
4	Softmax	**5/5**	**3/5**	**4/5**	0/5	0/5	0/5	0/5	0/5	**4/5**	**2/5**	0/5	0/5
5	LayerNorm	**3/5**	**5/5**	**2/5**	0/5	**2/5**	**1/5**	0/5	0/5	**1/5**	0/5	0/5	0/5
6	RMSNorm	**3/5**	**5/5**	**1/5**	**1/5**	**1/5**	0/5	0/5	0/5	**1/5**	**1/5**	0/5	0/5
7	GEMV	**5/5**	**2/5**	**1/5**	0/5	**2/5**	**1/5**	0/5	0/5	**4/5**	**5/5**	0/5	0/5
8	GEMM	**4/5**	**2/5**	0/5	0/5	**4/5**	**3/5**	**2/5**	**1/5**	**2/5**	**4/5**	**1/5**	**1/5**
9	BatchMatMul	**2/5**	**2/5**	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5
10	Linear	**4/5**	**2/5**	0/5	0/5	**3/5**	**2/5**	**1/5**	0/5	**2/5**	0/5	0/5	0/5
11	Conv2D			–‡		0/5	0/5	0/5	0/5	**2/5**	**1/5**	0/5	0/5
12	DepthwiseConv			–‡		0/5	0/5	0/5	0/5	**5/5**	**3/5**	0/5	0/5
Level 1 Subtotal		*37/50*	*30/50*	*12/50*	*1/50*	*16/60*	*11/60*	*3/60*	*1/60*	*34/60*	*28/60*	*2/60*	*1/60*
Level 2: Composite Operations (Max 20 iterations)													
13	RoPE	**1/5**	0/5	0/5	0/5	0/5	0/5	0/5	0/5			–†	
14	FFN	**3/5**	**2/5**	0/5	0/5	**2/5**	**1/5**	0/5	0/5	**1/5**	0/5	0/5	0/5
15	SwiGLU	**4/5**	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5
16	ScaledDotProduct	**3/5**	**2/5**	0/5	0/5	**1/5**	0/5	0/5	0/5			–†	
17	FlashAttention	**3/5**	**1/5**	0/5	0/5			–§				–†	
18	MHA	**3/5**	0/5	0/5	0/5	0/5	0/5	0/5	0/5			–†	
19	GQA	**1/5**	0/5	0/5	0/5	0/5	0/5	0/5	0/5			–†	
20	MQA	**1/5**	0/5	0/5	0/5	0/5	0/5	0/5	0/5			–†	
Level 2 Subtotal		*19/40*	*5/40*	*0/40*	*0/40*	*3/35*	*1/35*	*0/35*	*0/35*	*1/10*	*0/10*	*0/10*	*0/10*
Level 3: End-to-End System (Max 25 iterations)													
21	ConvBlock			–‡		0/5	0/5	0/5	0/5	0/5	**1/5**	0/5	0/5
22	DecoderBlock (LLaMA)	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5			–†	
23	DecoderBlock (T5)	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5			–†	
Level 3 Subtotal		*0/10*	*0/10*	*0/10*	*0/10*	*0/15*	*0/15*	*0/15*	*0/15*	*0/5*	*1/5*	*0/5*	*0/5*
Total		**56/100**	**35/100**	**12/100**	**1/100**	**19/110**	**12/110**	**3/110**	**1/110**	**35/75**	**29/75**	**2/75**	**1/75**

Evaluation: kernel performance



What does an agent need to succeed?

- Extended reasoning is essential
- In-context learning helps
- Documentation is important
- Agents utilizing the native tool calls do better

	No Thinking			Med. Thinking		
Task	Avg Iter	Avg Tok	Succ	Avg Iter	Avg Tok	Succ
Attention	5.6	817	0/5	19.3	2135	3/5
MHA	6.8	611	0/5	18.4	2671	3/5
FFN	11.8	867	0/5	18.8	2270	3/5
SwiGLU	9.4	997	0/5	18.0	2157	4/5

		GPT-5.2		Gemini 3 Flash	
Hardware	**Task**	Zero-shot	One-shot	Zero-shot	One-shot
PLENA	MHA	3/5	4/5	0/5	1/5
	MQA	1/5	4/5	0/5	0/5
AMD NPU	FFN	0/5	2/5	0/5	1/5
	SDPA	0/5	1/5	0/5	0/5

KernelCraft abilities

- Applicability to general CPU workloads
- Comparison with human-expert tuned kernels

Config	Human Expert	KernelCraft	Compiler
C1	33327 (20×)	269239 (2.5×)	673098
C2	75732 (18.2×)	2300040 (0.6×)	1380024
C3	87470 (31.5×)	5510952 (0.5×)	2755476

- Co-designing ISA for emerging ML workloads

Task	Description	Succ.	Iter.	Speedup
Level 1: Primitive Operations (max 10 iterations)				
multiply	Shift-and-add multiplication	✓	3	1.04×
vvadd	Vector-vector add	✓	5	1.10×
copy	Copy memory block	✓	8	1.50×
median	3-element median filter	✓	10	1.03×
dotprod	Dot product	✓	8	1.01×
Level 1 Subtotal		*5/5*		
Level 2: Algorithmic Operations (max 10 iterations)				
qsort	Quicksort algorithm	✓	9	1.63×
rsort	Radix sort algorithm	✓	6	0.12×
towers	Towers of Hanoi	✓	8	1.18×
Level 2 Subtotal		*3/3*		
Level 3: End-to-End System (max 15 iterations)				
dhrystone	Mixed integer operations	✓	12	0.93×
pmp	Physical Memory Protection	✓	8	1.68×
Level 3 Subtotal		*2/2*		
Total		**10/10**		

Personal opinions

Strengths:

- Addresses a real gap
- Useful analysis of benefits of providing appropriate tools/data to models
- Decent results

Weaknesses:

- Still significantly worse than human experts.
- Baseline is weak immature compilers.
- Uneven model choice (Sonnet 4 vs GPT-5.2); 5 runs per cell, no error bars.
- Approximate floating point equality might be weak
- Potential reward hacking.