How Efficient is LLM-Generated Code? A Rigorous & High-Standard Benchmark

Ruizhong Qiu† Weiliang Will Zeng‡ James Ezick‡ Christopher Lott‡ Hanghang Tong†
† ILLINOIS ‡ Qualcomm
{rq5,htong}@illinois.edu {wzeng,jezick,clott}@qti.qualcomm.com
https://github.com/q-rz/enamel

ICLR

PROPOSED BENCHMARK: ENAMEL

HumanEval Canonical: $2^{\Theta(n)}$ recursions

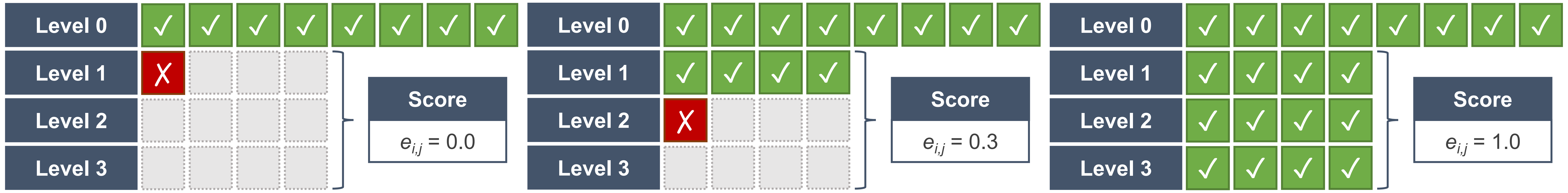
```
def fib(n):
  if n == 0:
    return 0
  if n == 1:
    return 1
  return fib(n - 1) + fib(n - 2)
```

GPT-4 Turbo: $\Theta(n)$ iterations

```
def fib(n):
  a, b = 0, 1
  for _ in range(n):
    a, b = b, a + b
  return a
```

Our Expert: $\Theta(\log n)$ iterations

```
def fib(n):
  if n == 0: return 0
  a, b = 0, 1
  for n in bin(n)[3 :]:
    a, b = a * a + b * b, b * (a * 2 + b)
    if n == '1': a, b = b, a + b
  return b
```



HumanEval Canonical:
Level 0 ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓
Level 1 ✗ ☐ ☐ ☐
Level 2 ☐ ☐ ☐ ☐
Level 3 ☐ ☐ ☐ ☐
Score $e_{i,j}$ = 0.0

GPT-4 Turbo:
Level 0 ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓
Level 1 ✓ ✓ ✓ ✓
Level 2 ✗ ☐ ☐ ☐
Level 3 ☐ ☐ ☐ ☐
Score $e_{i,j}$ = 0.3

Our Expert:
Level 0 ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓
Level 1 ✓ ✓ ✓ ✓
Level 2 ✓ ✓ ✓ ✓
Level 3 ✓ ✓ ✓ ✓
Score $e_{i,j}$ = 1.0

✓ Test case passed ✗ Time limit exceeded ☐ Test case skipped

Example (*n*-th Fibonacci): · **Level 0:** $n \leq 10$ · **Level 1:** $n \leq 30$ · **Level 3:** $n \leq 9\,000$ · **Level 3:** $n \leq 10\,000$

PROPOSED METRIC: eff@*k*

- Proposed **metric** $\mathrm{eff}_i@k$ (a generalization of $\mathrm{pass}_i@k$):

$$\mathrm{eff}_i@k \coloneqq \mathbb{E}_{c_{i,1},\dots,c_{i,k}}\left[\max_{j=1}^{k} e_{i,j}\right];$$

  - $c_{i,j}$: the j-th LLM-generated code sample for problem i;
  - $e_{i,j}$: the efficiency score of code $c_{i,j}$ compared with our human expert.
- An **estimator** $\widehat{\mathrm{eff}}_i@k$ for $\mathrm{eff}_i@k$ using $n \geq k$ code samples:
  - Let $e_{i,(r)}$ be the r-th smallest score among $e_{i,1}, \dots, e_{i,n}$. **Estimator**:

$$\widehat{\mathrm{eff}}_i@k \coloneqq \sum_{r=k}^{n} \binom{r-1}{k-1} e_{i,(r)} \Big/ \binom{n}{k}.$$

  - ✓ **Unbiasedness**: for any $n \geq k$,

$$\mathbb{E}_{c_{i,1},\dots,c_{i,n}}\left[\sum_{r=k}^{n} \binom{r-1}{k-1} e_{i,(r)} \Big/ \binom{n}{k}\right] = \mathbb{E}_{c_{i,1},\dots,c_{i,k}}\left[\max_{j=1}^{k} e_{i,j}\right].$$

  - ✓ **Variance reduction**: for any $n \geq k$,

$$\mathrm{Var}_{c_{i,1},\dots,c_{i,n}}\left[\sum_{r=k}^{n} \binom{r-1}{k-1} e_{i,(r)} \Big/ \binom{n}{k}\right] \leq \frac{k}{n} \mathrm{Var}_{c_{i,1},\dots,c_{i,k}}\left[\max_{j=1}^{k} e_{i,j}\right].$$

  - A **numerically stable** implementation: See our paper for detail…

EXPERT-WRITTEN SOLUTIONS

- **Problemset**: 142 problems selected from HumanEval.
- **Our expert solutions**: much more **efficient** than HumanEval+'s.

ID	Problem Description	HumanEval+ Solution	Our Expert Solution
#10	Find the shortest palindrome that begins with a given string S	$O(\lvert S\rvert^2)$: Enumerate suffixes and check palindromicity	$\Theta(\lvert S\rvert)$: Use Knuth–Morris–Pratt w.r.t. reversed S plus S
#36	Count digit 7's in positive integers $< n$ that are divisible by 11 or 13	$\Theta(n \log n)$: Enumerate integers $< n$ and count the digits	$\Theta(\log n)$: Design a dynamic programming over digits
#40	Check if a list l has three distinct elements that sum to 0	$O(\lvert l\rvert^3)$: Enumerate triples in l and check their sums	$O(\lvert l\rvert^2)$: Use a hash set and enumerate pairs in l
#109	Check if a list a can be made non-decreasing using only rotations	$O(\lvert a\rvert^2)$: Enumerate the rotations of a and check	$O(\lvert a\rvert)$: Check if the list a has at most one inversion
#154	Check if any rotation of a string b is a substring of a string a	$O(\lvert b\rvert^2 \lvert a\rvert)$: Enumerate rotations and run string matching	$O(\lvert a\rvert + \lvert b\rvert)$: Run the suffix automaton of a w.r.t. $b + b$

TAKEAWAYS

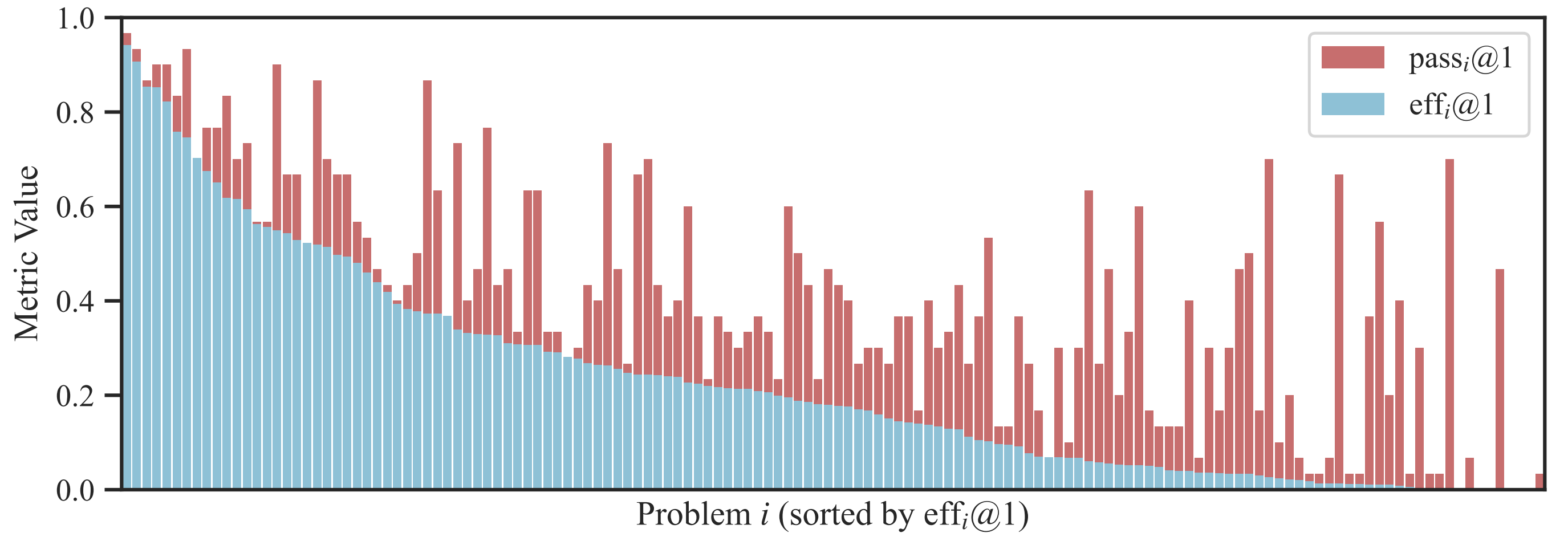
- **Overall evaluation results:** (*table truncated*)
  - Even strong LLMs **fall short** of generating **expert-level** efficient code.

Model	Greedy		Sampling					
	eff@1	pass@1	eff@1	pass@1	eff@10	pass@10	eff@100	pass@100
GPT-4 Turbo	0.470	0.796	—	—	—	—	—	—
GPT-4	0.454	0.831	—	—	—	—	—	—
Llama 3 70B Instruct	0.421	0.746	0.438	0.747	0.526	0.836	0.575	0.880
Llama 3 8B Instruct	0.344	0.592	0.345	0.564	0.500	0.770	0.595	0.874
Mixtral 8x22B Instruct	0.408	0.746	0.407	0.721	0.575	0.870	0.704	0.923
Mixtral 8x7B Instruct	0.266	0.444	0.279	0.456	0.436	0.689	0.542	0.810
Claude 3 Opus	0.401	0.789	—	—	—	—	—	—
Claude 3 Sonnet	0.345	0.662	0.365	0.677	0.498	0.814	0.594	0.887
Claude 3 Haiku	0.386	0.739	0.382	0.730	0.478	0.831	0.529	0.861
Phind Code Llama V2	0.394	0.683	0.372	0.638	0.584	0.862	0.723	0.935
ChatGPT	0.364	0.683	0.374	0.673	0.557	0.847	0.690	0.937
Code Llama 70B Python	0.264	0.500	0.082	0.177	0.326	0.610	0.614	0.908
Code Llama 34B Python	0.268	0.458	0.226	0.405	0.511	0.786	0.711	0.934
Code Llama 13B Python	0.216	0.408	0.204	0.372	0.487	0.732	0.714	0.899
Code Llama 7B Python	0.247	0.373	0.180	0.320	0.432	0.663	0.643	0.837
StarCoder	0.195	0.352	0.134	0.236	0.355	0.557	0.542	0.787
CodeGen 16B	0.169	0.310	0.122	0.219	0.326	0.512	0.536	0.761
CodeGen 6B	0.193	0.296	0.111	0.188	0.298	0.455	0.491	0.694
CodeGen 2B	0.153	0.254	0.098	0.168	0.264	0.389	0.421	0.602
CodeT5+ 16B	0.160	0.317	0.130	0.250	0.343	0.551	0.551	0.785

- **Evaluation on two subsets:** (*table truncated*)
  - LLMs **struggle** in designing advanced algorithms.
  - LLMs are largely **unaware** of implementation optimization.

Model	Algorithm Design Subset						Implementation Optimization Subset					
	eff@1	pass@1	eff@10	pass@10	eff@100	pass@100	eff@1	pass@1	eff@10	pass@10	eff@100	pass@100
Llama 3 70B Instruct	0.246	0.660	0.306	0.749	0.359	0.750	0.404	0.791	0.497	0.869	0.551	0.920
Llama 3 8B Instruct	0.201	0.518	0.303	0.724	0.367	0.849	0.313	0.582	0.468	0.806	0.571	0.906
Mixtral 8x22B Instruct	0.225	0.635	0.363	0.837	0.470	0.900	0.376	0.783	0.556	0.914	0.686	0.947
Mixtral 8x7B Instruct	0.124	0.391	0.244	0.681	0.344	0.850	0.248	0.473	0.411	0.699	0.515	0.827
Claude 3 Sonnet	0.184	0.577	0.328	0.804	0.450	0.950	0.358	0.723	0.475	0.846	0.548	0.893
Claude 3 Haiku	0.149	0.692	0.208	0.752	0.266	0.775	0.360	0.772	0.465	0.889	0.513	0.923
Phind Code Llama V2	0.185	0.554	0.353	0.789	0.401	0.849	0.351	0.712	0.567	0.901	0.732	0.968
ChatGPT	0.120	0.488	0.304	0.799	0.483	0.950	0.337	0.715	0.508	0.864	0.633	0.949
Code Llama 70B Python	0.018	0.100	0.129	0.519	0.402	0.950	0.076	0.181	0.294	0.627	0.589	0.920
Code Llama 34B Python	0.071	0.293	0.271	0.713	0.425	0.881	0.197	0.415	0.473	0.804	0.687	0.949
Code Llama 13B Python	0.058	0.212	0.276	0.665	0.478	0.844	0.176	0.405	0.476	0.784	0.715	0.928
Code Llama 7B Python	0.068	0.202	0.231	0.589	0.393	0.761	0.165	0.349	0.417	0.703	0.620	0.863

- **Distribution of problem difficulties:**
  - High $\mathrm{pass}_i@1$, low $\mathrm{eff}_i@1$: *seemingly* easy task, **non-trivial** algorithm.



SCAN TO LEARN MORE

ACKNOWLEDGEMENTS NSF USDA NIFA IBM