Majority Voting for Code Generation

Tim Launer, Jonas Hübotter, Marco Bagatella, Ido Hakimi, Andreas Krause

ETH zürich | Learning & Adaptive Systems

Functional majority voting is an effective test-time scaling strategy for code generation.

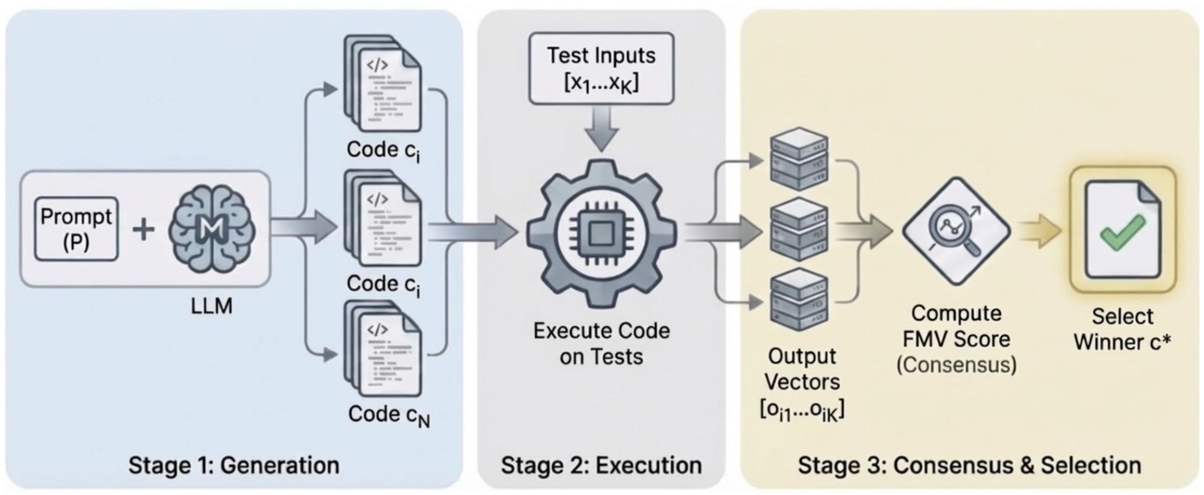
Background

In the absence of a verifier, how can we select the correct attempt among many generations?

Main idea: **Majority voting**

- Sample multiple responses, select the most common response.
- Known to work in single-answer settings like math.

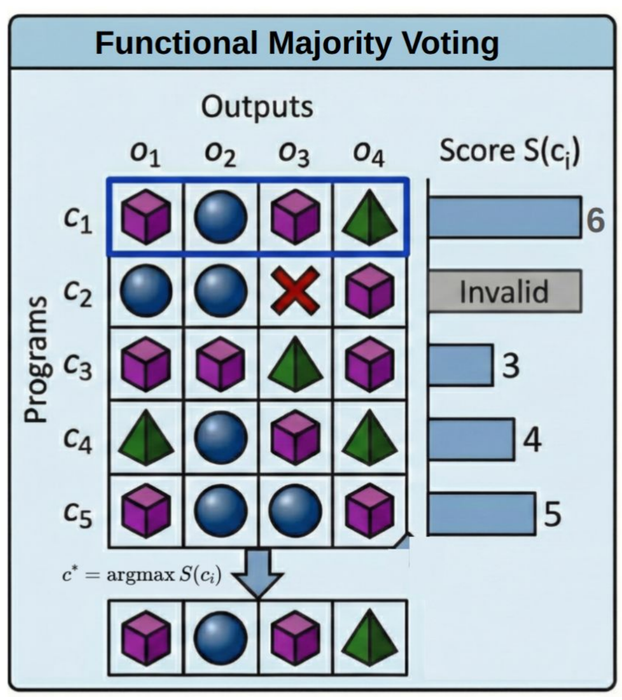
Question: Does majority voting also work in multi-answer settings like code generation?



Functional Majority Voting (FMV)

- Generate *N* candidate solutions
- Evaluate against *K* test inputs
- Remove candidates with runtime errors
- Select candidate "most in agreement" with other candidates:

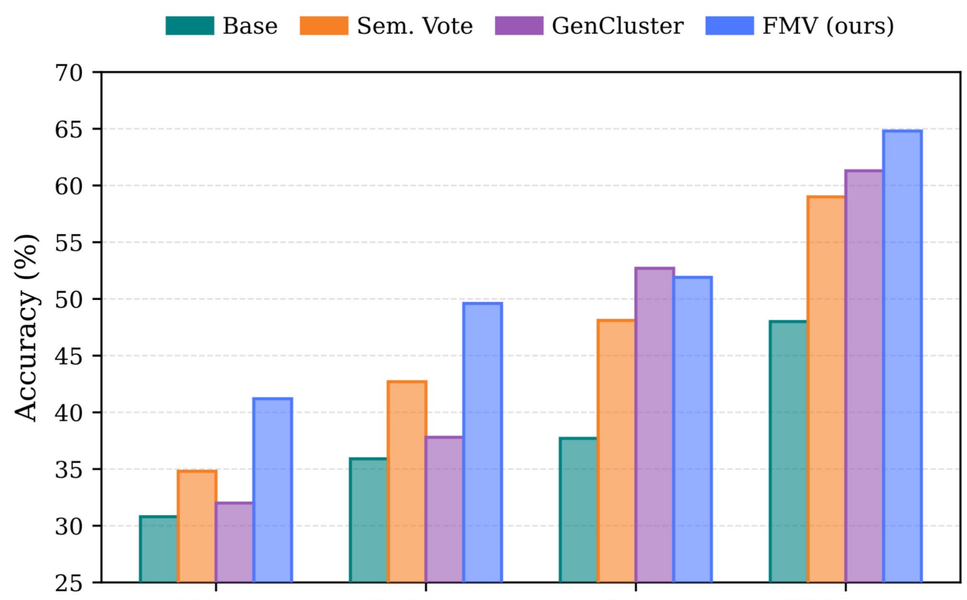
$$S(c_i) = \sum_{j \neq i} \sum_{k=1}^{K} \mathbb{I}(o_{i,k} = o_{j,k})$$



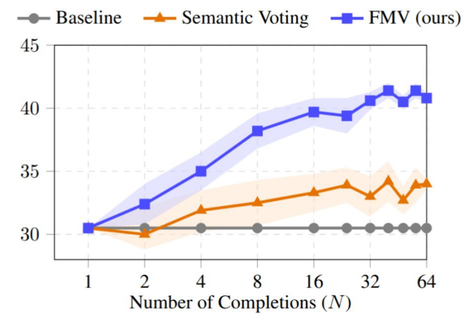
FMV as Test-Time Scaling

We evaluate Qwen3-2507 on LiveCodeBench v6. Baselines:

- **Semantic voting (Jiang et al., 2026)**: Each candidate is embedded with a sentence embedding model. The candidate with the highest average similarity is selected.
- **GenCluster (Samadi et al., 2026):** Judge-based selection in a round-robin tournament. This is significantly more expensive than FMV.



Takeaway: Functional majority voting outperforms semantic voting and performs on-par or better than the more expensive GenCluster.



FMV for Test-Time RL

FMV can be used as synthetic labels for TTRL (Zuo et al., 2025). Two options:

- FMV: Output labels correspond to output of candidate selected by FMV.
- Pointwise-FMV: Output labels correspond to most common output per test input.

Method	Train (mean@64)	Hold-Out (mean@64)
Base (Qwen3-4B-Instruct)	30.8 ± 3.2	31.6 ± 3.1
+ FMV TTRL	36.9 ± 3.8	34.3 ± 3.2
+ Pointwise-FMV TTRL	36.6 ± 3.8	34.5 ± 3.2

Takeaway: FMV + RL can improve held-out accuracy, but does not raise test-time scaling performance.

Training Method	mean@64 (%)	FMV (%)	best@64 (%)
Base (Qwen3-4B-Instruct)	30.8 ± 3.2	$\mathbf{41.2} \pm 4.3$	$\mathbf{48.9} \pm 1.0$
Test-Time Reinforcement Learning			
+ Joint ($N = 32$)	34.4 ± 3.8	38.9 ± 4.1	44.2 ± 4.2
+ Pointwise ($N = 32$)	36.6 ± 3.8	40.5 ± 4.3	46.6 ± 4.2
+ Joint ($N = 128$)	$\mathbf{36.9} \pm 3.8$	40.5 ± 4.2	45.0 ± 4.2
+ Pointwise ($N = 128$)	36.8 ± 4.2	38.9 ± 3.9	46.6 ± 4.2