NEURAL INFORMATION PROCESSING SYSTEMS



PSD: Pushing the Pareto Frontier of Diffusion LLMs via Parallel Speculative Decoding

Shengyin Sun[1], Yiming Li[1]*, Renxi Liu[1], Xinqi Li[1], Hui-Ling Zhen[1], Weizhe Lin[1],
Chen Chen[1], Xianzhi Yu[1], Mingxuan Yuan[1], Chen Ma[2]

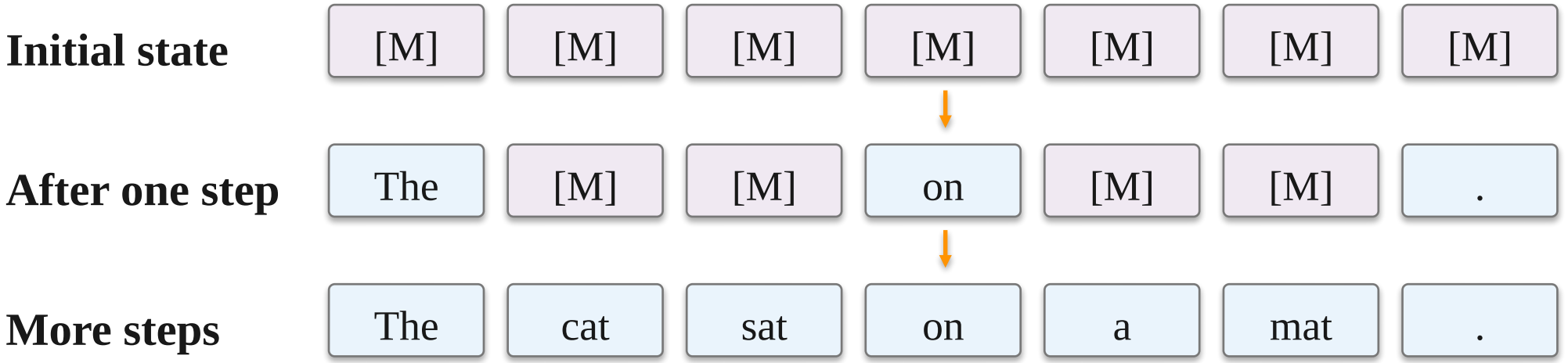
[1] Huawei Technologies [2] City University of Hong Kong * Corresponding author

HUAWEI

CityU 香港城市大學 City University of Hong Kong

Background — Parallel Predictions, Sequential Denoising

- **A dLLM predicts all masked positions in one forward pass.**
- A transfer policy decides which predictions become committed tokens.



One token per step: **N tokens ≈ N forward passes**.

Predicting in parallel is only the first step toward decoding faster.

Figure 1. Iterative denoising of a masked token sequence.

Motivation — Two Complementary Axes of Acceleration

Spatial parallelism

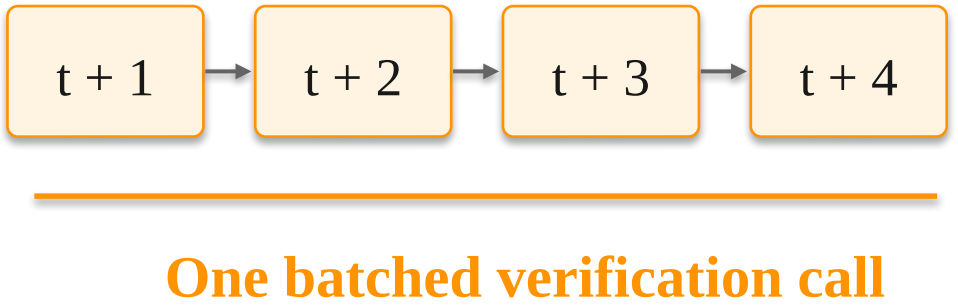
Reveal more positions per step



More tokens per step can force uncertain predictions to commit.

Temporal speculation

Verify several future steps together



Temporal-only methods still reveal one token per denoising step.

PSD combines more tokens per step with more progress per verification.

Figure 2. Spatial parallelism and temporal speculation.

Key Insight — Today's Confidence Predicts Future Reveals

- **High-confidence positions remain informative across future steps.**
- **Reuse cached predictions** to construct drafts without extra model calls.

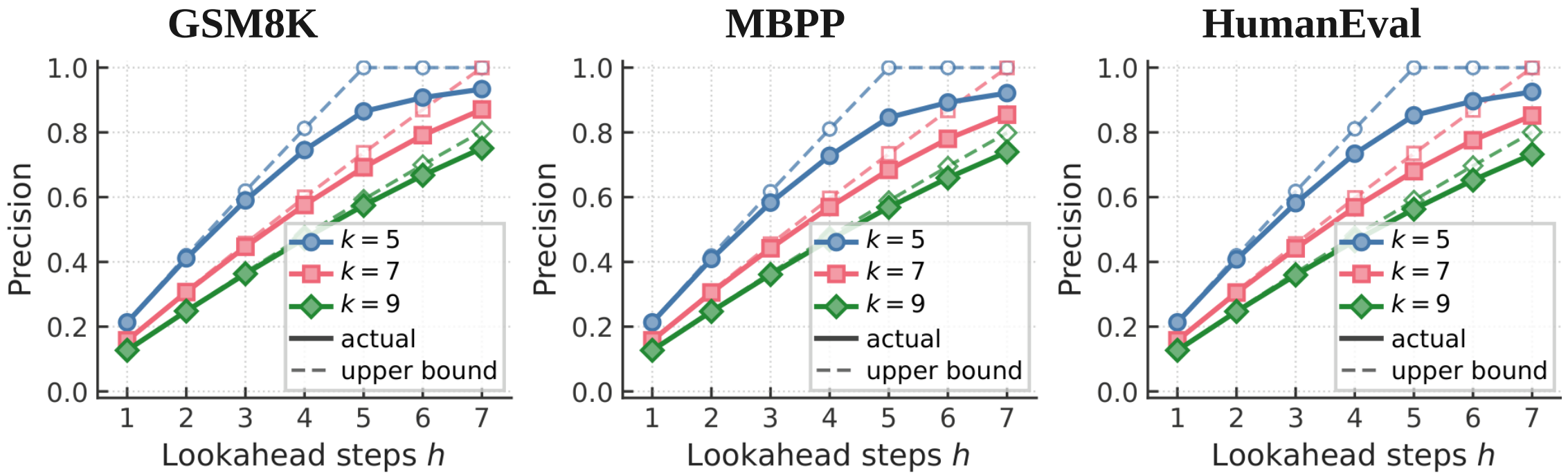


Figure 3. Precision@K of future reveal positions across three tasks.

Methodology [1/4] — One Framework, Three Stages

Unmask in parallel → **Draft from cached predictions** → **Verify and accept**

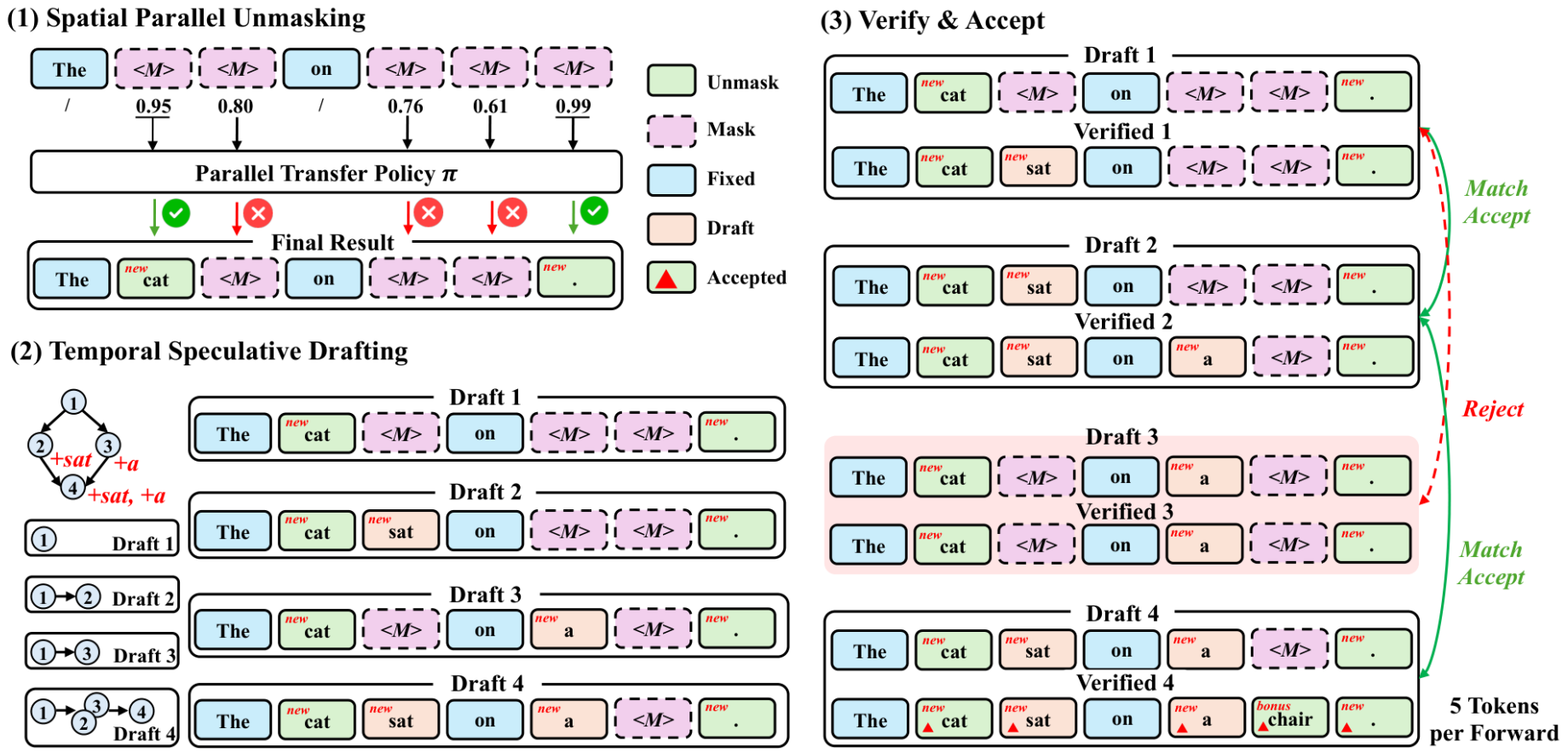
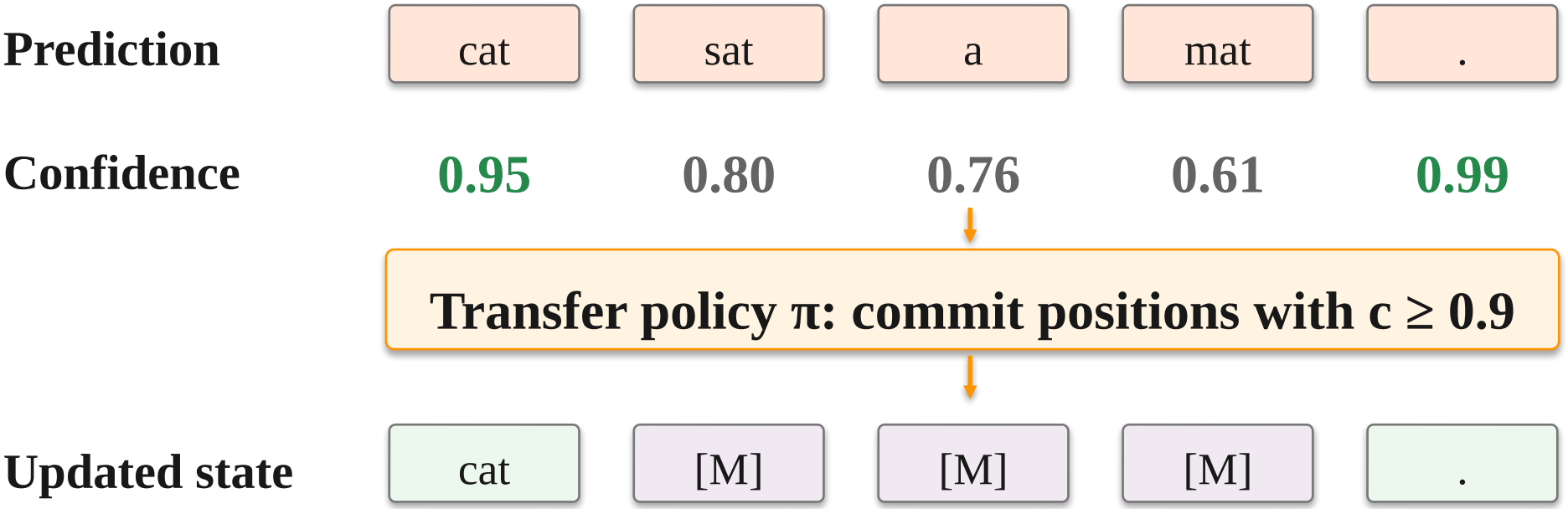


Figure 4. PSD: parallel unmasking, speculative drafting, and verification.

Methodology [2/4] — Plug in Any Spatial Policy

- One forward pass gives the top token and confidence at every masked position.



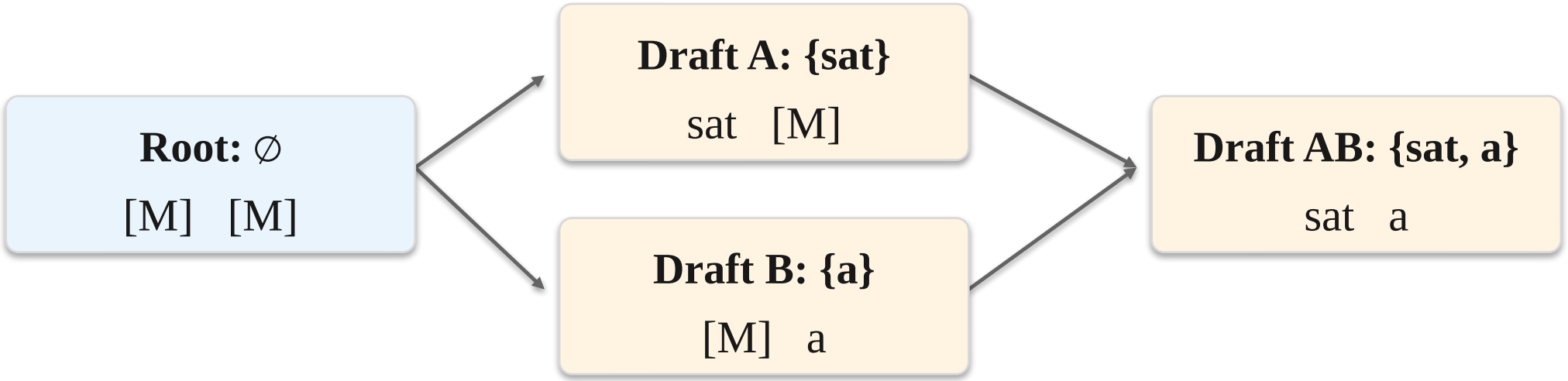
Instantiations: **Confidence (τ = 0.9)** or **LocalLeap**.

PSD keeps the spatial policy modular; the speculation module stays the same.

Figure 5. Confidence-based spatial unmasking with threshold 0.9.

Methodology [3/4] — Draft a DAG Without Extra Calls

- Rank the remaining masked positions using the cached confidence scores.
- **Fill candidate subsets** with cached top-1 tokens; no new model call is needed.



A deeper state can have multiple parents—and multiple valid acceptance paths.

Figure 6. A draft DAG with multiple paths to the same state.



Methodology [4/4] — Verify Once, Accept Hierarchically

- **Batch all candidate states** in one verification forward pass.
- Check newly added tokens against an accepted parent, using the spatial policy's criterion.

Table 1. Hierarchical acceptance on a two-position draft DAG.

Parent → child	New position	Passes criterion?	Decision
Root → A	sat	Yes	Accept A
Root → B	a	No	Reject this edge
A → AB	**a**	**Yes**	**Accept AB via A**

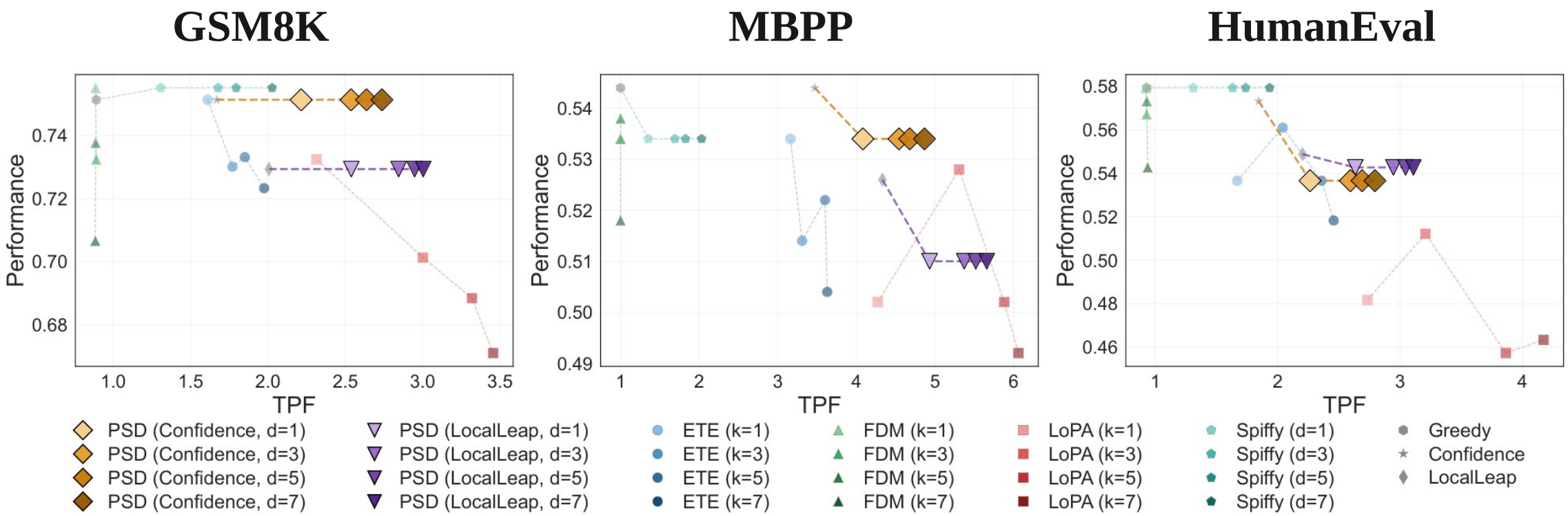
Choose the accepted state with **the most revealed positions**.

Then commit eligible bonus tokens from the chosen state's verifier output.

Experiments [1/4] — Dream: Quality–Efficiency Frontier

- Quality–efficiency comparison on Dream-7B across math and code tasks.

  TPF = tokens per forward pass; GSM8K reports accuracy, and code tasks report pass@1.

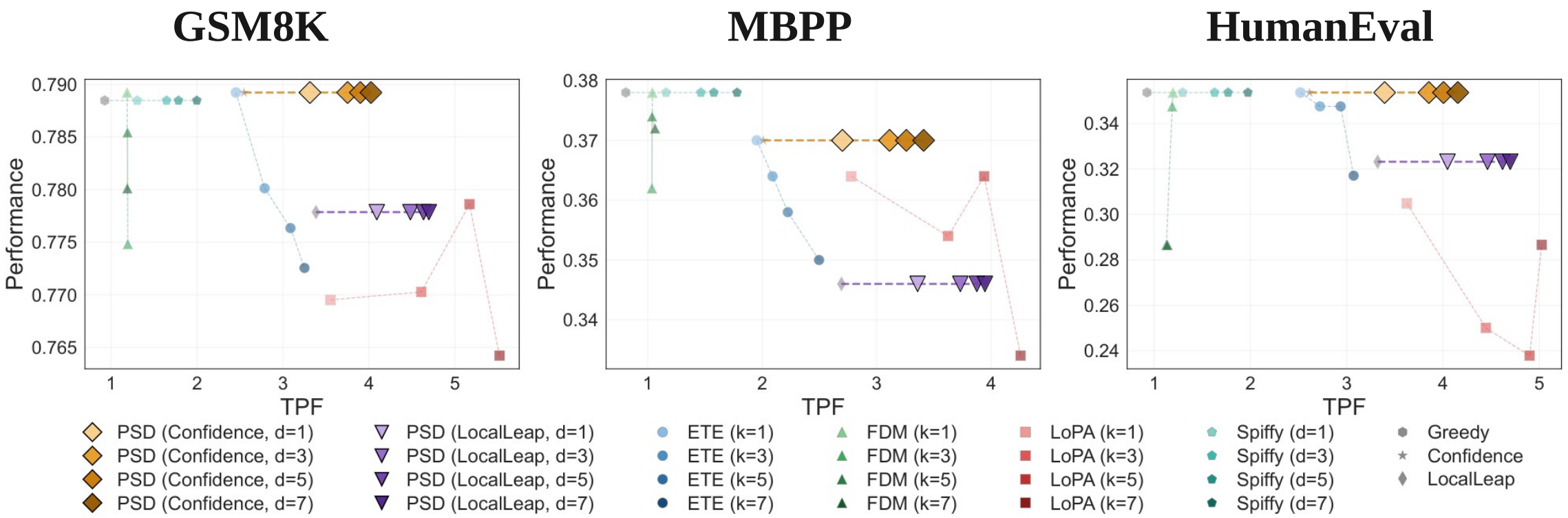


MBPP: 4.9 TPF at 53.4% pass@1; Spiffy reaches 2.0 at the same quality.

Figure 7. Dream: task performance versus tokens per forward pass.

Experiments [2/4] — LLaDA-1.5: Quality–Efficiency Frontier

- Quality–efficiency comparison on LLaDA-1.5 across math and code tasks.

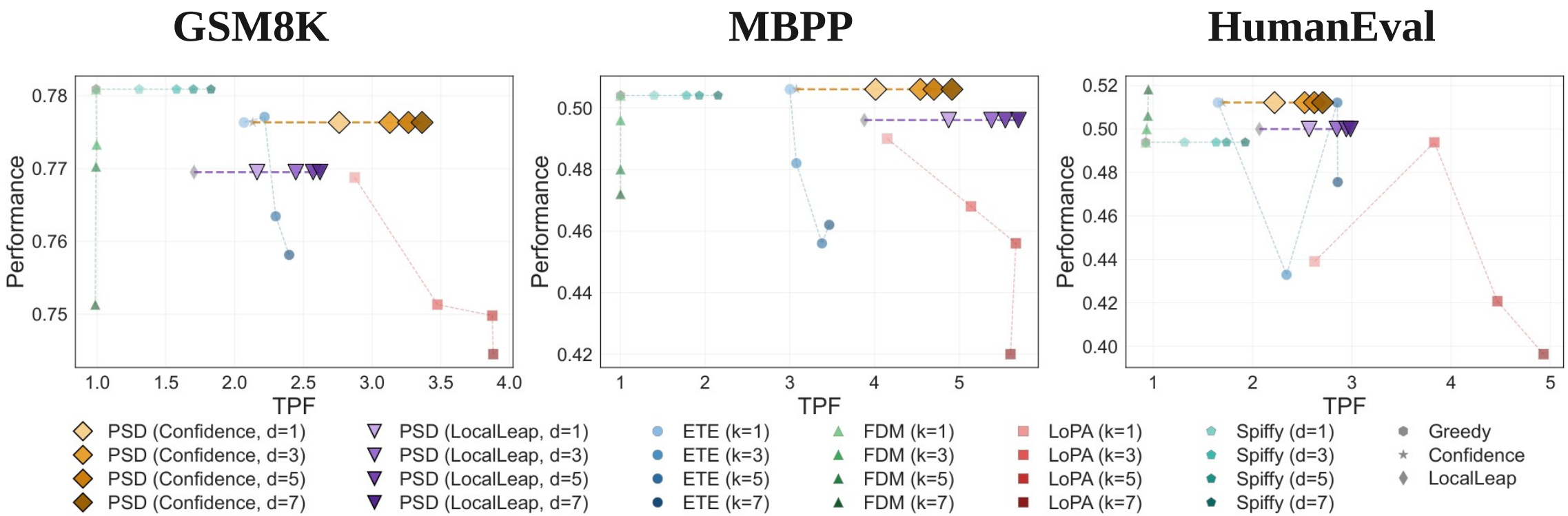


GSM8K: 4.0 TPF and 78.9% accuracy, versus Greedy's 78.8%.

Figure 8. LLaDA-1.5: task performance versus tokens per forward pass.

Experiments [3/4] — openPangu: Quality–Efficiency Frontier

- Quality–efficiency comparison on openPangu-7B across math and code tasks.



MBPP: 4.9 TPF while retaining greedy-level pass@1 at 50.6%.

Figure 9. openPangu: task performance versus tokens per forward pass.



Experiments [4/4] — Verification Protects Code Quality

- Code generation is sensitive to premature token commitments.

Table 2. LLaDA-1.5 on HumanEval: quality and decoding efficiency.

Method	TPF	Pass@1	Change vs. Greedy
Greedy	1.0	35.4%	–
LoPA (k = 5)	4.9	23.8%	−11.6 pp
PSD (Confidence)	**4.2**	**35.4%**	**0.0 pp**

Comparable parallelism can have very different quality costs.



Extension — Multi-News Summarization

- Additional TPF gains also appear in longer-form summarization.

Table 3. Multi-News summarization: ROUGE scores and TPF speedup.

Model / method	R-1	R-2	R-L	TPF speedup
Dream / Greedy	31.0	13.1	15.6	1.00×
Dream / Spiffy	31.3	13.2	15.9	2.37×
Dream / PSD (LocalLeap)	**32.9**	**13.5**	**16.3**	**2.61×**
openPangu / Greedy	41.0	15.4	18.3	1.00×
openPangu / Spiffy	39.2	15.4	17.8	2.05×
openPangu / PSD (LocalLeap)	**39.1**	**16.1**	**17.7**	**2.73×**

Up to 2.73× TPF; openPangu's ROUGE-1 / ROUGE-L are lower than Greedy.



Takeaways — A General Recipe for Faster dLLM Decoding

1 **Combine spatial and temporal progress**

Parallel unmasking and speculative verification improve different dimensions.

2 **Reuse what the model already predicts**

Training-free drafting; configurable spatial policy; hierarchical acceptance.

3 **Improve the quality–efficiency trade-off**

Up to 5.5× TPF; accuracy comparable to Greedy in the reported settings.

Next: measure deployment latency and adapt speculation to the workload.

Thank you for watching!

Code

Paper

Project page: https://github.com/sunshy-1/PSD
Correspondence: shengysun4-c@my.cityu.edu.hk