Computer Science

MINT group

STAR AI RESEARCH LAB UCLA

Accelerating Diffusion LLMs via Adaptive Parallel Decoding

Daniel Israel, Guy Van den Broeck, Aditya Grover

Punchline

MINT group

STAR AI RESEARCH LAB UCLA

Prompt

Toulouse has twice as many sheep as Charleston. Charleston has 4 times as many sheep as Seattle. How many sheep do Toulouse, Charleston, and Seattle have together if Seattle has 20 sheep?

Autoregressive Decoding

Let's start by defining the number of sheep each city has in terms of variables and then use the given information to find the exact numbers.
1. Let \(S \) represent the number of sheep in Seattle.
2. According to the problem, Seattle has 20 sheep, so \(S = 20 \).
3. Charleston has 4 times as many sheep as Seattle. So, let \(C \) represent the number of sheep in Charleston:
\[C = 4S = 4 \times 20 = 80 \]
4. Toulouse has twice as many sheep as Charleston. Let \(T \) represent the number of sheep in Toulouse:
\[T = 2C = 2 \times 80 = 160 \]
5. To find the total number of sheep in Toulouse, Charleston, and Seattle together, we add the number of sheep in each city:
\[T + C + S = 160 + 80 + 20 = 260 \]
Therefore, the total number of sheep in Toulouse, Charleston, and Seattle together is 260.

Latency: 7.08 seconds Throughput: 37 tokens per second

Adaptive Parallel Decoding

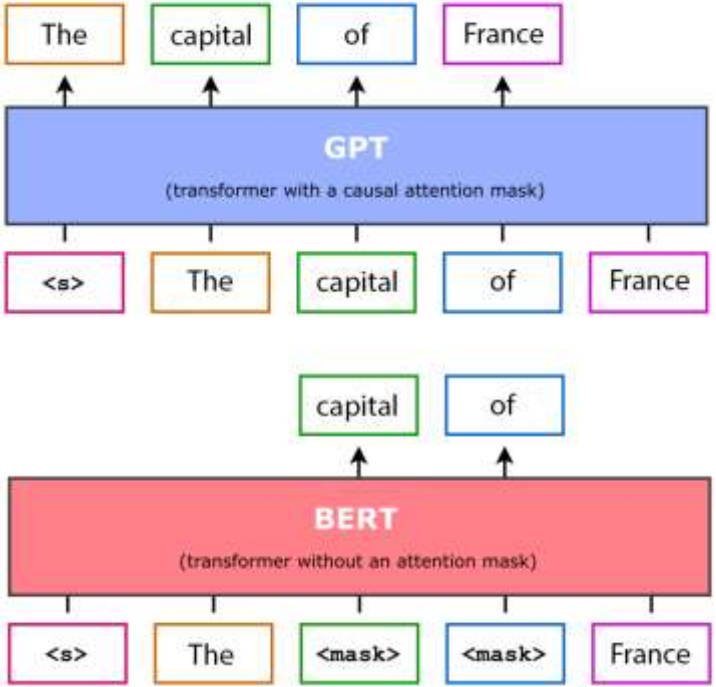
Let's break down the information:
* Seattle has 20 sheep.
* Charleston has 4 times as many sheep as Seattle, so Charleston has 4 * 20 = 80 sheep.
* Toulouse has twice as many sheep as Charleston, so Toulouse has 2 * 80 = 160 sheep.
Now, let's add up the number of sheep each city has:
Seattle: 20 sheep
Charleston: 80 sheep
Toulouse: 160 sheep
Total number of sheep: 20 + 80 + 160 = 100 + 160 = 260
So, Toulouse, Charleston, and Seattle have a total of \boxed{260} sheep together

Latency: 2.75 seconds Throughput: 59 tokens per second

What is a dLLM?

MINT group

	Autoregressive	Diffusion
Architecture	GPT	BERT
Masking	Causal Attention Masking	Masked Token Input
Training	Exact NLL	Denoising ELBO
Inference	Sequential	Parallel



What is a dLLM?

MINT group

- dLLMs can compute the product of marginals given observed context at arbitrary positions

- Define a forward noise process that convert tokens to [MASK] (and never change back)

- dLLMs are trained to denoise by maximizing a variational lower bound

(1) $$p_{\mathrm{D}}\left(x_{\mathcal{Q}} \middle| x_{\mathcal{O}};\theta\right) = \prod_{i\in\mathcal{Q}} p_\theta(x_i|x_{\mathcal{O}})$$

(2) $$q_{t|0}\left(x_i^t \middle| x_i^0\right) = \begin{cases} t, & \text{if } x_i^t = [\mathrm{MASK}] \\ 1-t, & \text{if } x_i^t = x_i^0 \\ 0, & \text{otherwise} \end{cases}$$

(3) $$q_{t|0}(x^t|x^0) = \prod_i q_{t|0}(x_i^t|x_i^0)$$

(4) $$\log p_\theta(x^0) \geq \mathbb{E}_{t\sim U(0,1),x^t\sim q\left(x^t \middle| x^0\right)}[\log p_{\mathrm{D}}(x_{\mathbb{1}(x^t=[\mathrm{MASK}])}|x_{\mathbb{1}(x^t\neq[\mathrm{MASK}])};\theta)]$$

Generation Example

MINT group

STAR AI RESEARCH LAB UCLA

```
Janet's ducks lay 16 eggs per day. She eats three for breakfast
every morning and bakes muffins for her friends every day with
four. She sells the remainder at the farmers' market daily for $2
per fresh duck egg. How much in dollars does she make every day at
the farmers' market?
```

Dream 7B [2]

Open Source dLLMs Fall Short

MINT group

- Not really "diffusion"
  - Random unmask order does not perform well
  - Entropy/Confidence ordered decoding in practice

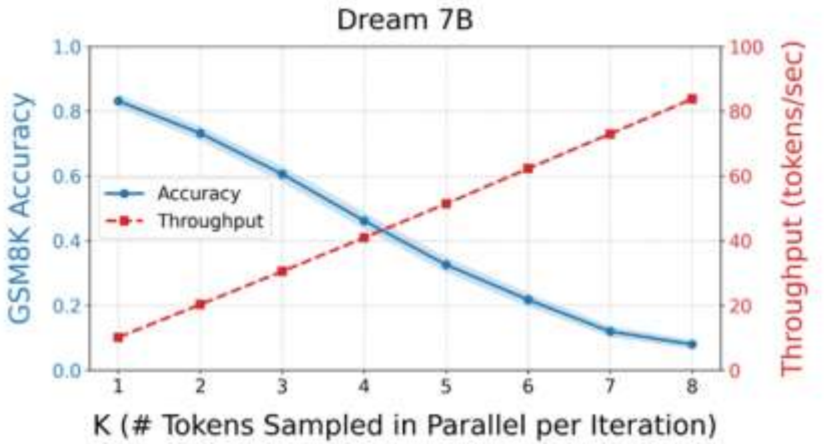
- Not really "parallel"
  - Need as many steps as tokens generated for good performance

Table 1: dLLM Quality and Throughput with Different Decoding Approaches

Model	**GMS8K Accuracy**	**Throughput (tokens/sec)**
Dream 7B (Random, 256 Steps)	0.404 ± 0.021	3.31 ± 0.068
Dream 7B (Entropy, 128 Steps)	0.708 ± 0.020	7.57 ± 0.157
Dream 7B (Entropy, 256 Steps)	0.804 ± 0.017	4.28 ± 0.080
Dream 7B (Left to Right, 256 Steps)	0.832 ± 0.016	10.1 ± 0.015
Llada 8B (Random, 256 Steps)	0.456 ± 0.022	5.07 ± 0.168
Llada 8B (Confidence, 128 Steps)	0.526 ± 0.022	13.6 ± 0.284
Llada 8B (Confidence, 256 Steps)	0.534 ± 0.022	6.63 ± 0.143
Llada 8B (Left to Right, 256 Steps)	0.712 ± 0.020	9.33 ± 0.016
Qwen2.5 7B (Autoregressive)	$\mathbf{0.854 \pm 0.015}$	$\mathbf{38.6 \pm 0.004}$

Curse of Parallel Decoding

MINT group



Central Question

MINT group

STAR AI RESEARCH LAB UCLA

How do we increase dLLM throughput without dramatically hurting quality?

(Have our cake and eat it too)

dLLMs Can Sample Autoregressively

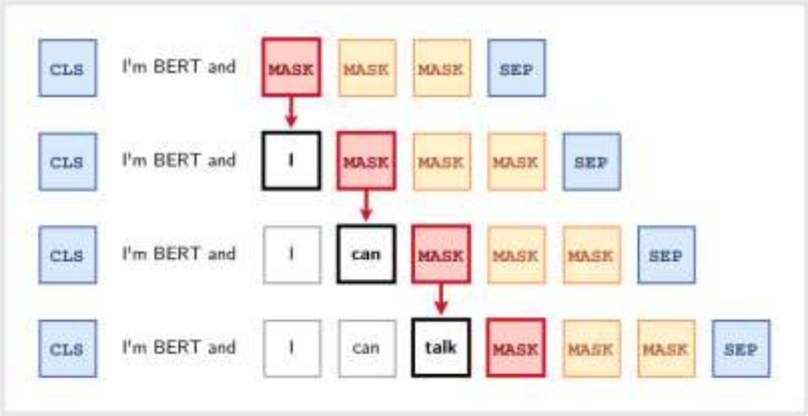
MINT group

$$p_{\mathrm{D}}(x_Q|x_O;\theta) = \prod_{i \in Q} p_\theta(x_i|x_O)$$

$$p_{\mathrm{AR}}(x;\theta) = \prod_{i=1}^{n} p_D(x_i|x_{<i};\theta)$$

- Use the diffusion model autoregressively
- Advantages
  - High quality
  - Use as gold standard distribution
- Disadvantages
  - One token at a time sequentially is slow
  - No KV caching

Text generation



BERTs are generative in-context learners [1]

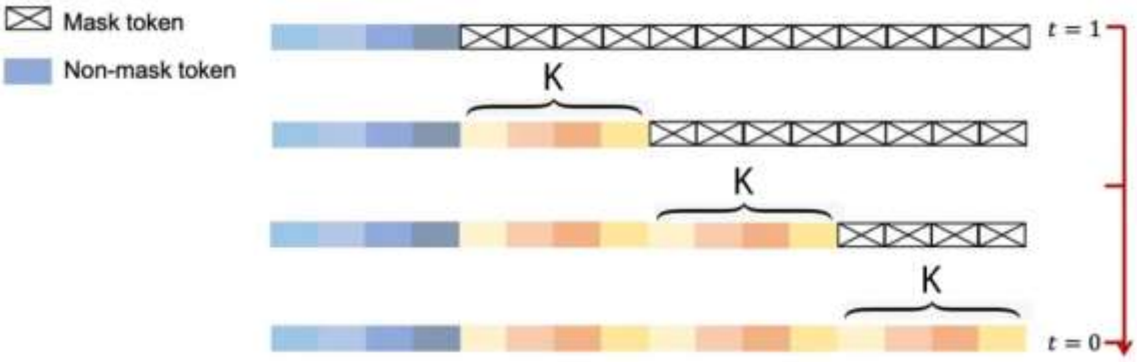
From Sequential to Parallel Sampling

MINT group

- Instead of sampling autoregressively 1 token at a time, we can sample in parallel k tokens per iteration

- For high k, quality goes down

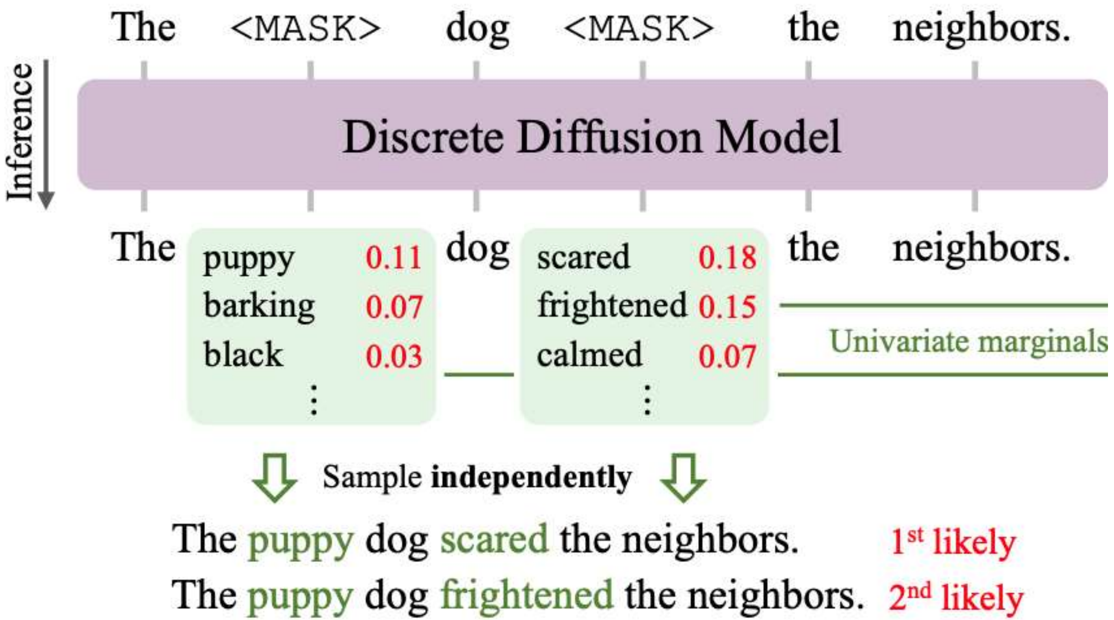
$$p_{\mathrm{SAR}}(x;\theta;k) = \prod_{i=1}^{\lfloor \frac{n-1}{k} \rfloor + 1} p_{\mathrm{D}}(x_{(ik-k+1:ik)} | x_{<(ik-k+1)}; \theta)$$



Failure Mode

MINT group

- Modelling the marginals $p(x_1), p(x_2)$ will not capture the joint $p(x_1, x_2)$
- Fail to capture dependence: $p(x_1|x_2)$
- We do not want to draw samples that are marginally likely but jointly unlikely



Discrete Copula Diffusion [3]

Goals

MINT group

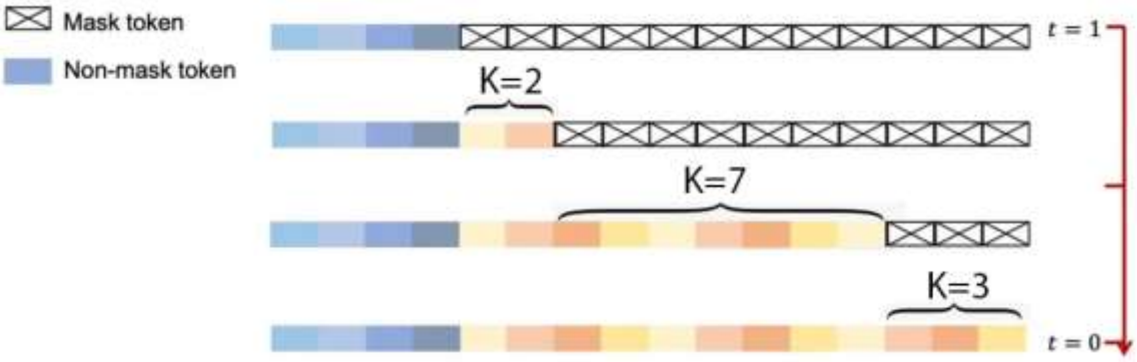
1. Speed: Minimize $|\mathcal{G}|$

   $\mathcal{G} = \{(s_1, e_1), (s_2, e_2), \ldots, (s_l, e_l)\}$

2. Quality: Minimize the distance between p_{APD} and p_{AR}

$$p_{\mathrm{AR}}(x;\theta) = \prod_{i=1}^{n} p_D(x_i | x_{<i}; \theta)$$

$$p_{\mathrm{APD}}(x;\theta,\mathcal{G}) = \prod_{(s,e)\in\mathcal{G}} p_{\mathrm{D}}(x_{s:e} | x_{<s})$$



Constraints

- We have a strong diffusion model Dream 7B: p_{D}
  - Samples in parallel
  - Computes likelihood sequentially
- We have a weak AR model Qwen2.5 0.5B: $\hat{p}_{\mathrm{AR}}$
  - Computes likelihood in parallel
  - Samples sequentially
- This is not the same setting as speculative decoding!
  - We do not access to fast likelihood queries from our diffusion model
  - The big model samples quickly, the small model verifies quickly
  - The cost of obtaining target distribution p_{AR} directly is too high

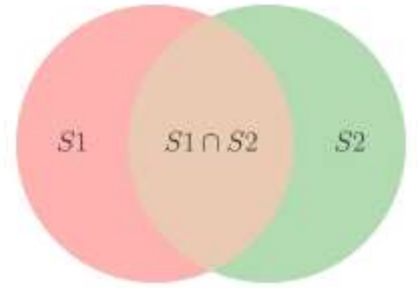
$$p_{\mathrm{D}}(x_{\mathcal{Q}}|x_{\mathcal{O}};\theta) = \prod_{i\in\mathcal{Q}} p_{\theta}(x_i|x_{\mathcal{O}})$$

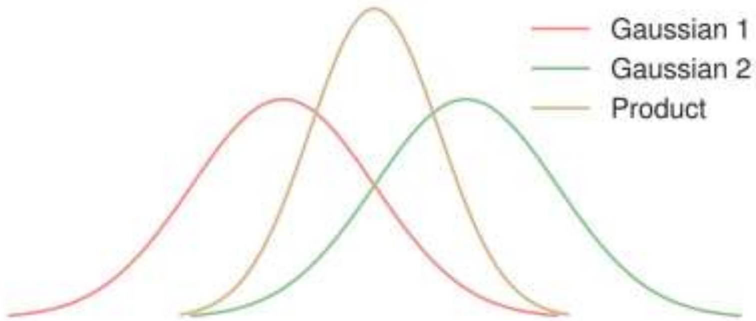
$$p_{\mathrm{AR}}(x;\theta) = \prod_{i=1}^{n} p_{D}(x_i|x_{<i};\theta)$$

Multiplicative Mixture

MINT group

- Product of experts [9]
- Intuitively, it corresponds to the intersection of distributions instead of the union
- Implemented by adding logits and renormalizing with softmax
- We define our target distribution as the multiplicative mixture between p_{D} and $\hat{p}_{\mathrm{AR}}$, weighted by R
- Will capture marginal distributions and the joint dependencies to approximate p_{AR}





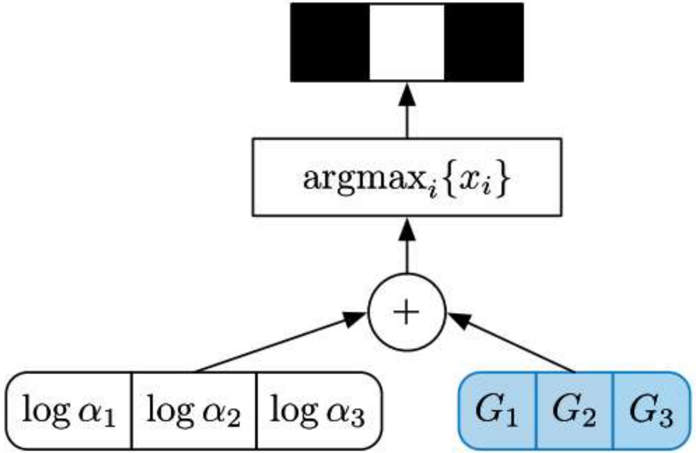
$$p_{\mathrm{T}}(x) = \frac{1}{Z} p_{\mathrm{D}}(x)^{R} \hat{p}_{\mathrm{AR}}(x)^{1-R}$$

Gumbel Argmax Sampling Trick

MINT group

ST★R AI RESEARCH LAB UCLA

- Given the output logits from a model for K categories: $L = [l_1, l_2, \ldots, l_k]$
- Sampling procedure
  1. Draw independent samples $r_1, \ldots, r_k \sim \text{Gumbel}(0,1)$
  2. Compute $L' = [l_1 + r_1, l_2 + r_2, \ldots, l_k + r_k]$
  3. Sample $x = \text{argmax}(L')$
- Mathematically equivalent to sampling x directly from categorical distribution
- Gumbel trick is a universal coupler [8]



Algorithm

MINT group

STAR AI RESEARCH LAB UCLA

x_1 x_2 x_3 x_4 x_5 x_6 x_7

Same Gumbel Noise

Algorithm

MINT group

Algorithm 1 Adaptive Parallel Decoding

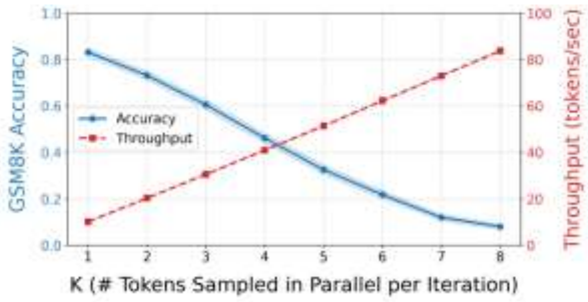
Input: Diffusion model p_{D}, Autoregressive model $\hat{p}_{\mathrm{AR}}$, Mixture Weight Parameter R, Maximum sequence length n
Output: Generated token sequence x
$x \leftarrow ()$ ▷ Stores the accepted tokens
$t \leftarrow 1$ ▷ Index of token to generate
while $t \leq n$ **do**
  $\texttt{marginal\_logits}_{t:n} \leftarrow p_{\mathrm{D}}(x_{t:n} \mid x_{<t})$
  $r \leftarrow \mathrm{Gumbel}(0, 1)$
  $\hat{x}_{t:n} \leftarrow \texttt{sample\_gumbel}(\texttt{marginal\_logits}_{t:n}, r)$
  $\texttt{joint\_logits}_{t:n} \leftarrow \hat{p}_{\mathrm{AR}}(\hat{x}_{t:n} \mid x_{<t})$
  $\texttt{product\_logits}_{t:n} \leftarrow \texttt{softmax}(R * \texttt{marginal\_logits}_{t:n} + (1-R) * \texttt{joint\_logits}_{t:n})$
  $\hat{y}_{t:n} \leftarrow \texttt{sample\_gumbel}(\texttt{product\_logits}_{t:n}, r)$
  $k \leftarrow \mathrm{sum}(\mathrm{cumprod}(\hat{x}_{t+1:n} = \hat{y}_{t+1:n})) + 1$
  $x \leftarrow \mathrm{concat}(x, \hat{x}_{t:t+k-1})$ ▷ Append accepted tokens
  $t \leftarrow t + k$
end while
return x

Results

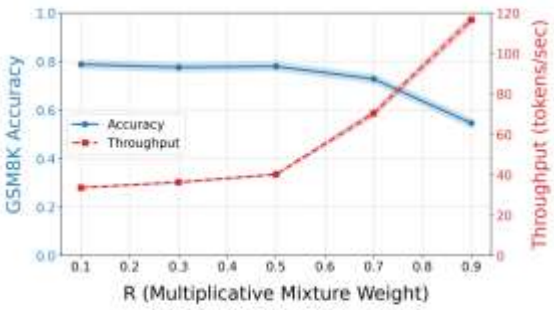
MINT group

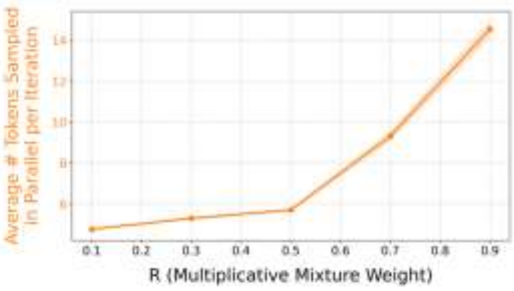
- ADP bends the throughput quality curve
- We can now achieve much higher throughput without large drop in quality
- Hyperparameter R (multiplicative mixture weight) allows control over trade off

Before



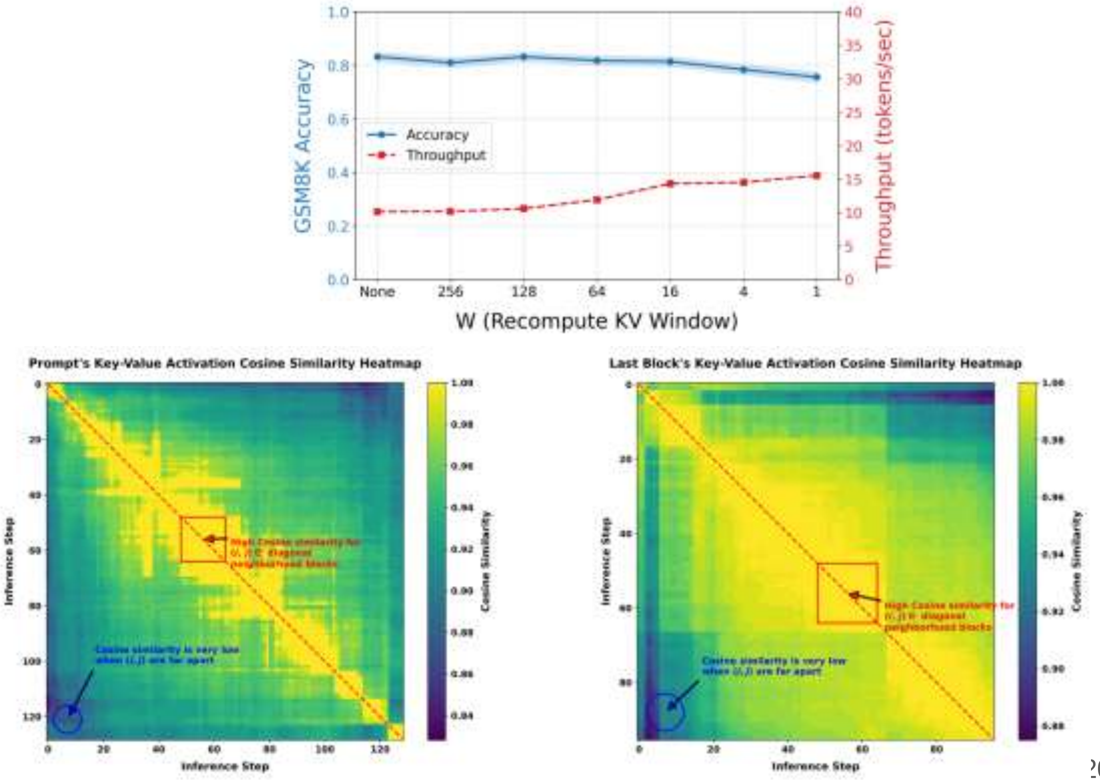
Now





KV Caching

- KV caching in dLLMs empirically works
- Only recompute the KV within a window of W of the most recent generated tokens



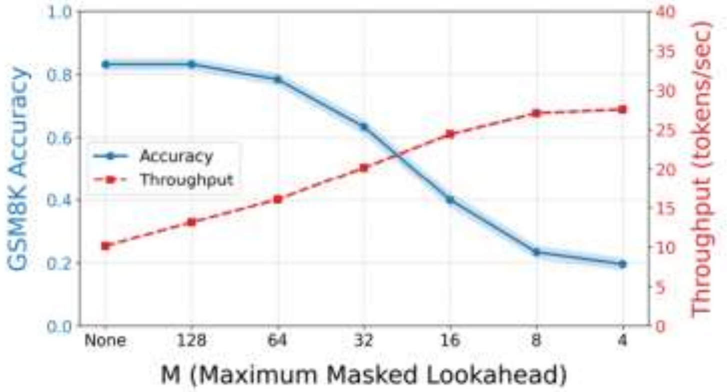
Fast-dLLM [4]

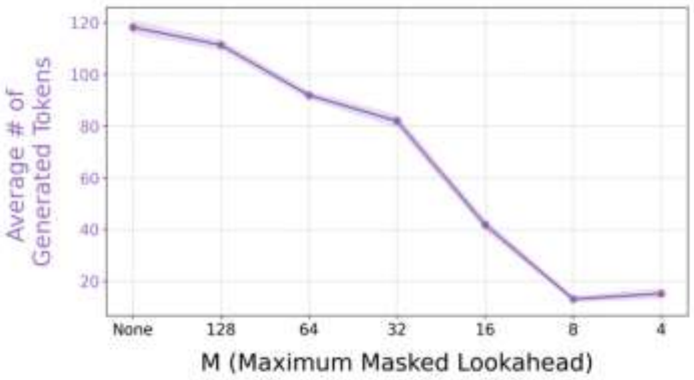
Maximum Masked Lookahead

MINT group

- For each step only append M mask tokens to the suffix

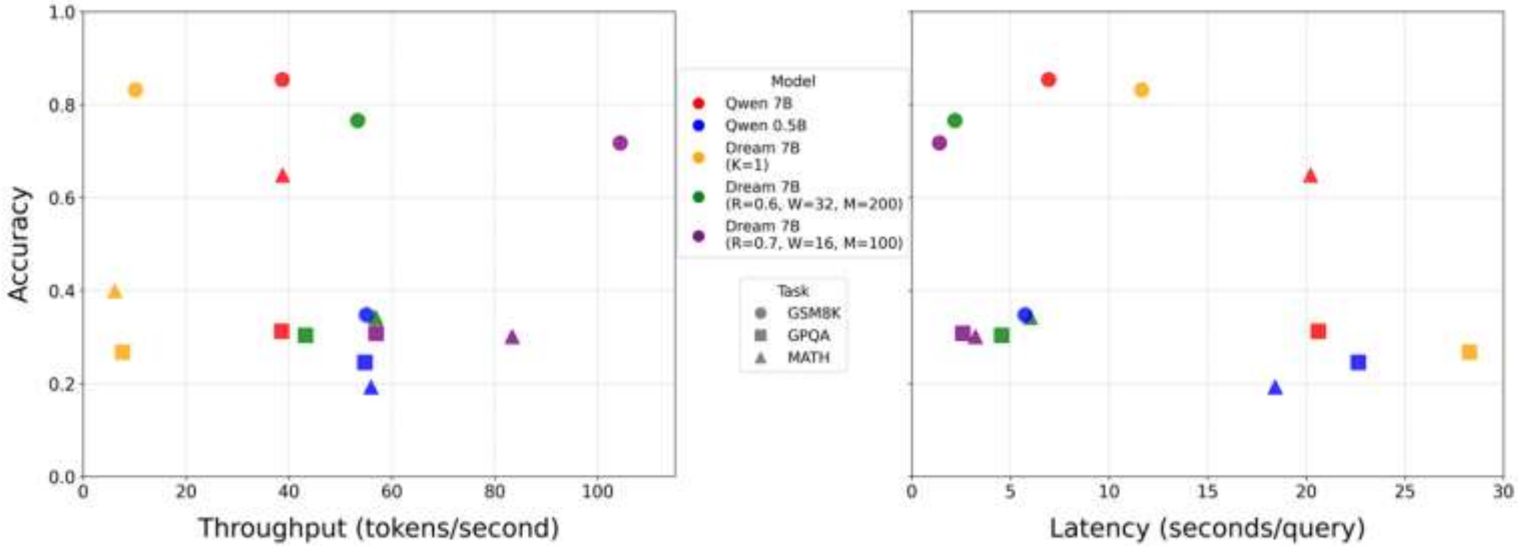
- This can change the distribution by encouraging shorter response





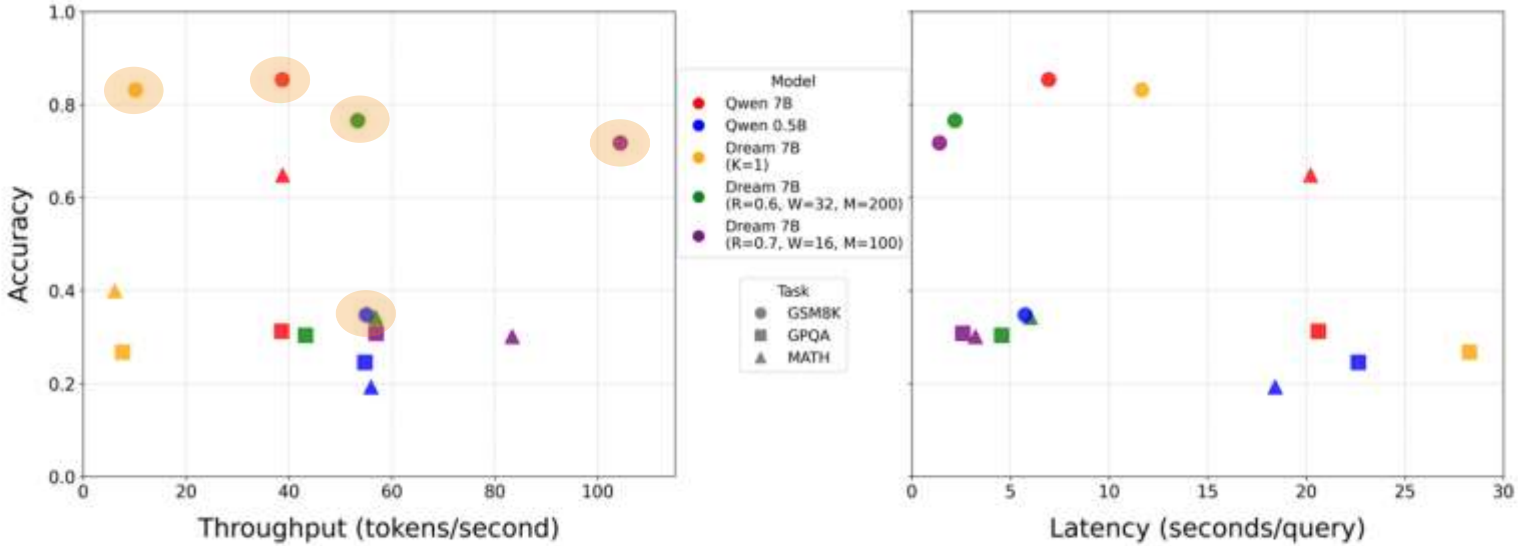
Final Results

MINT group



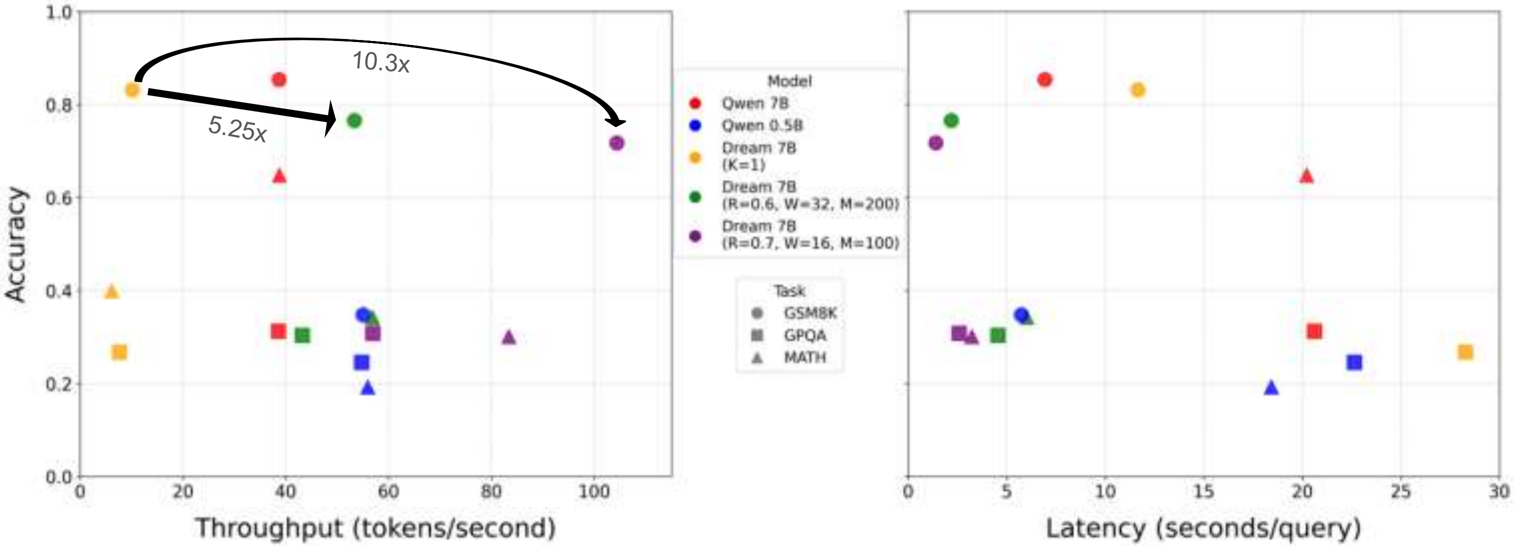
Final Results

MINT group



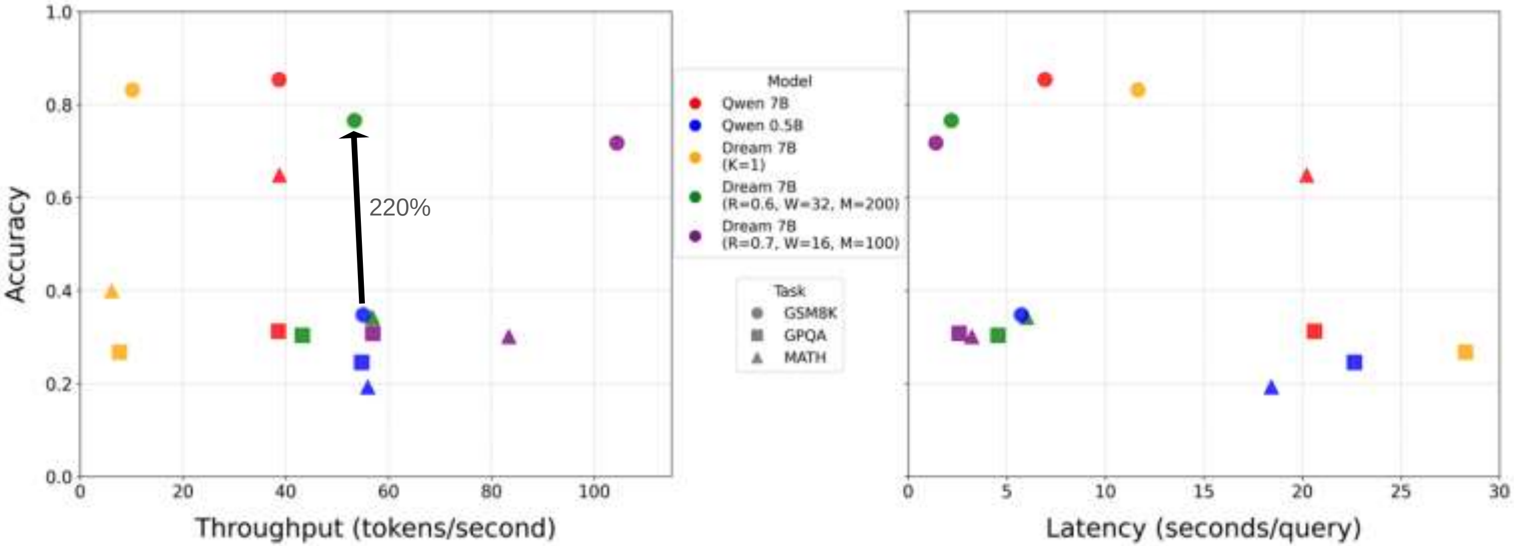
Final Results

MINT group



Final Results

MINT group



Concurrent Works

- Parallel Sampling + KV cache
  - Accelerating Diffusion Language Model Inference via Efficient KV Caching and Guided Diffusion [5]
    - Very similar to APD but accept token if it most likely under AR and diffusion
  - Fast-dLLM [4]
    - Unmask tokens if above a certain confidence threshold

- KV cache
  - dkv-cache: The cache for diffusion language models [6]
  - dllm-cache: Accelerating diffusion large language models with adaptive caching [7]

Thank you!

MINT group

For any questions or collaboration ideas ...

Website: https://danielmisrael.github.io/
Email: disrael@ucla.edu
X: @danielmisrael

References

[1] Samuel, David. "BERTs are generative in-context learners." Advances in Neural Information Processing Systems 37 (2024): 2558-2589.

[2] Ye, Jiacheng, et al. "Dream 7B." HKUNLP, 2025, hkunlp.github.io/blog/2025/dream.

[3] Liu, Anji, et al. "Discrete copula diffusion." arXiv preprint arXiv:2410.01949 (2024).

[4] Wu, Chengyue, et al. "Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding." arXiv preprint arXiv:2505.22618 (2025).

[5] Hu, Zhanqiu, et al. "Accelerating diffusion language model inference via efficient kv caching and guided diffusion." arXiv preprint arXiv:2505.21467 (2025).

[6] Ma, Xinyin, et al. "dkv-cache: The cache for diffusion language models." arXiv preprint arXiv:2505.15781 (2025).

[7] Liu, Zhiyuan, et al. "dllm-cache: Accelerating diffusion large language models with adaptive caching." arXiv preprint arXiv:2506.06295 (2025).

[8] Anari, Nima, Ruiquan Gao, and Aviad Rubinstein. "Parallel sampling via counting." Proceedings of the 56th Annual ACM Symposium on Theory of Computing. 2024.

[9] Hinton, Geoffrey E. "Training products of experts by minimizing contrastive divergence." Neural computation 14.8 (2002): 1771-1800.