Faster Auto-Regression $\leftrightarrow$ Smarter Diffusion: Towards a Unified Parallel Token-Processing Framework

Brandon Cho[†]
Princeton University
brandon.cho@princeton.edu

Jerry Han[†]
Princeton University
jerryhan@princeton.edu

Huxley Marvit[†]
Princeton University
huxley@princeton.edu

Abstract

Masked diffusion models such as LLaDA have emerged as a prominent alternative to autoregressive models (ARMs), speeding up the inference process in language models by allowing for many tokens to be generated simultaneously via an iterated unmasking process. Nevertheless, many existing models take a naive approach to masking, either unmasking a fraction of tokens randomly or after sorting by model confidence. Inspired by recent work on speculative decoding, we seek to improve on the unmasking process by introducing an external "scorer model" that evaluates each token generated at each time step in parallel. From a diffusion-first perspective, this effectively allows us to take an existing pretrained autoregressive frontier model, and utilize the intelligence of this model in parallel to improve on diffusion models. From an autoregressive-first perspective, this method allows us to approximate speculative decoding by treating our diffusion model as our draft model. We show that model performance on certain reasoning benchmarks remains invariant even after replacing the unmasking procedure at a majority of inference time steps with one based on an autoregressive scorer model. You can find a quick demo video of this architecture in action here, and the code that supports this here.

1 Introduction

Most frontier large language models (LLMs) are *autoregressive models*—that is, they perform next-token prediction using Transformers on a sequence x of length N via the model distribution

$$p(x;\theta) = \prod_{j=1}^{N} p(x_j | x_1, \ldots, x_{j-1}; \theta).$$

Under this paradigm, tokens are predicted serially, with all previously-generated tokens used as context when predicting each subsequent token. Past work has shown that performance on a wide variety of tasks improves as autoregressive models become larger (Brown et al., 2020). However, because the time complexity of naive approaches to next-token prediction in Transformer-based models scales quadratically with respect to sequence length, much recent work has been done to find efficient ways to speed up the inference process in such models. Work in this vein includes FlashAttention (Dao et al., 2022), which proposes an IO-aware exact attention algorithm designed to minimize expensive GPU memory operations, and Medusa (Cai et al., 2024), which uses multiple decoding heads in parallel to generate and verify multiple potential completions simultaneously. The magnitude of the speedup obtained via Medusa (and to a somewhat lesser extent, traditional speculative decoding methods) suggests that parallel token generation of some kind is a promising method to improve inference speed in current large language models.

Recently, however, text diffusion models have begun to appear as a viable alternative to traditional autoregressive frameworks (and variations thereof), promising rapid inference-time speedups by generating multiple tokens simultaneously and non-serially (Nie et al., 2025b,a). Commercial applications such as Inception Labs' Mercury Coder claim to achieve similar performance to major frontier models (including Claude 3.5 Haiku and GPT-4o Mini) on well-known code benchmarks such as HumanEval while producing over $5\times$ as many output tokens per second (Inception Labs, 2025; Chen et al., 2021).

With this in mind, we present two perspectives on our proposed architectural change: incorporating autoregressive models as scorers during the re-masking phase found in many state-of-the-art masked diffusion models.

[†]Equal contribution.

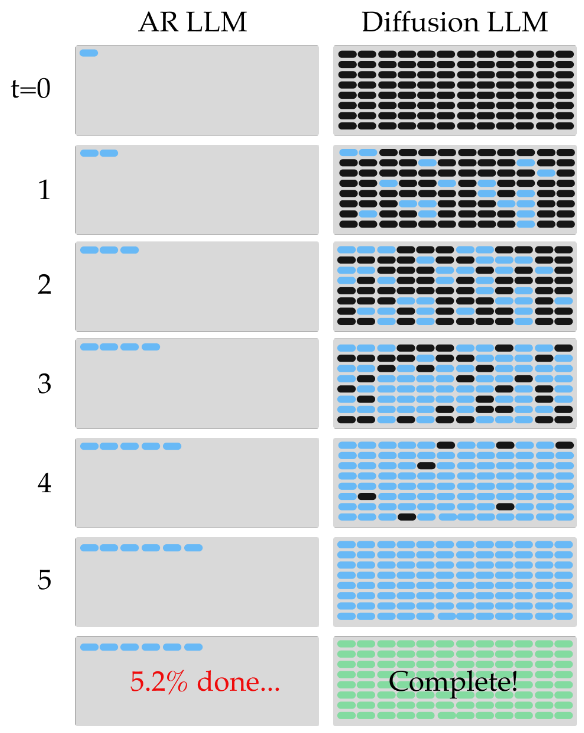


Figure 1: Autoregressive models must generate tokens one by one, while masked diffusion models may generate multiple tokens simultaneously and out of order.

1.1 Autoregressive-Model-First Perspective: Diffusion for Speculative Decoding

Speculative decoding has emerged as a powerful technique for dramatically speeding up large ARMs (autoregressive models). It works by having a smaller and cheaper "draft" model predict tokens ahead of the large model. Then, all of these tokens can be verified in parallel by the large model (for the time cost of generating only a single token) (Leviathan et al., 2023). If the draft model can generate 3 tokens in the time it takes the large model to generate one, for instance, then this speculative decoding setup effectively converts to roughly a 3× speedup over just using the large model to perform inference.

However, speculative decoding is a relatively new technique. The natural first step for creating this draft model is to use a distilled version of the large model, and this is in fact what most speculative diffusion setups still use. However, this is by no means a requirement — the draft model can use any architecture. Furthermore, the draft model's task is now fundamentally different to that of the large autoregressive model: tokens can be generated incorrectly at a much lower cost, as they are guaranteed to go through verification.

Thus, it is very possible that there are better architectures for this draft model. We propose one such new draft-model architecture: diffusion models. Diffusion models are naturally suited to this task because they can generate all tokens in parallel, effectively acting as an instant draft model. (Do note, however, that our setup only approximates speculative decoding, as we do not guarantee the same output as without this optimization). Ideally, one would distill a large ARM into a diffusion model. As a proof of concept, we first test without distillation on an out-of-the-box pretrained diffusion model.

Based on the chosen "confidence threshold" (or number of steps), which determines based on the large verifier model how many draft model-produced tokens to keep at each timestep, this setup effectively allows a tunable tradeoff between speed and output quality for large ARMs. In the accuracy limit, this setup would effectively fully (or near-fully) parallelize any autoregressive model.

1.2 Diffusion-First Perspective: Large ARM as Masker

Diffusion models have incredible promise. However, there is much training to be done before these diffusion models even have the chance to reach near the performance or size of frontier models. However, we can possibly improve the output quality (at the cost of a minor sacrifice in speed) by "piggy-backing" on larger pretrained autoregressive models. As opposed to the classical paradigm where tokens generated by a diffusion model are re-masked based on the diffusion model's confidences, we propose an architecture where the tokens are remasked based on the confidences of an existing large pretrained autoregressive model, which can be calculated in parallel over all the diffusion-model generated tokens. In theory, this forms a slower (since there is now the added cost of one ARM forward pass per diffusion step), but higher output quality diffusion model. (This is, once again, a tunable tradeoff).

1.3 Aggregate Perspective

While both these perspectives offer conceptual power, they are referring to the same architecture. This architecture combines diffusion models and large pretrained ARM to create a tunable tradeoff between model speed and output quality. In this work, we conceptualize, implement, and then evaluate this novel architecture, and lay the groundwork for what future research would need to be done to convert this architecture from a successful proof-of-concept to a production-ready architecture.

2 Related Work

Our work relies heavily on existing studies on masked diffusion models and speculative decoding for both inspiration and implementation details.

2.1 Speculative Decoding

Recent work on speculative decoding has proposed using multiple language models in tandem to accelerate inference without sacrificing quality. In a typical setup, a smaller draft model is used to quickly propose one or more candidate tokens, and a larger, more accurate target model then verifies these candidates in a single pass (Leviathan et al., 2023). This approach allows several tokens to be generated per iteration instead of the standard one token at a time, significantly reducing latency. In fact, it was formally demonstrated that speculative decoding can accelerate generation by 2–3× on large Transformers (e.g. T5-XXL) while producing identical outputs to standard greedy decoding.

2.2 Masked Diffusion Models

Prior work on masked or non-autoregressive diffusion-style language models has similarly aimed to speed up text generation by predicting multiple tokens in parallel and iteratively refining uncertain positions.

A representative state-of-the-art example is LLaDA (Nie et al., 2025b), a large language diffusion model with 8B parameters trained from scratch that rivals LLaMA3 8B in performance and validates the viability of masked diffusion models as an alternative generative paradigm.

LLaDA is pretrained on text with random masks applied independently to all tokens at the same ratio $t \sim U[0, 1]$. LLaDA employs a standard Transformer architecture as the mask predictor, without causal masks as its formulation allows it to see the entire input to generate predictions. During generation, LLaDA masks all tokens and then generates text by simulating a diffusion-style reverse process from an entirely masked sequence to a fully unmasked output. At each timestep, the mask predictor model predicts all currently masked tokens in parallel, conditioning on the visible context to fill in possible values. After each prediction step, a flexible remasking strategy is deployed to re-mask out a certain fraction of tokens, and the model repeats the prediction based on the updated sequence. This process continues for a fixed number of iterations or until no masks remain. When performing inference with LLaDA, one must set in advance the number of tokens to be generated (the sequence length L) as well as the number of steps N over which this reverse process is to take place.

A key difference among masked generation methods lies in the strategy for selecting which tokens to unmask or remask at each iteration. As mentioned in (Nie et al., 2025b), the remasking strategy should in principle be purely random. The authors also implement *low-confidence* remasking, which involves remasking a proportion of predicted tokens with the lowest confidence based on the predictions and often proves to be more effective than *random* remasking. By focusing on the most uncertain positions, this strategy can improve final accuracy and convergence in fewer iterations.

3 Methodology

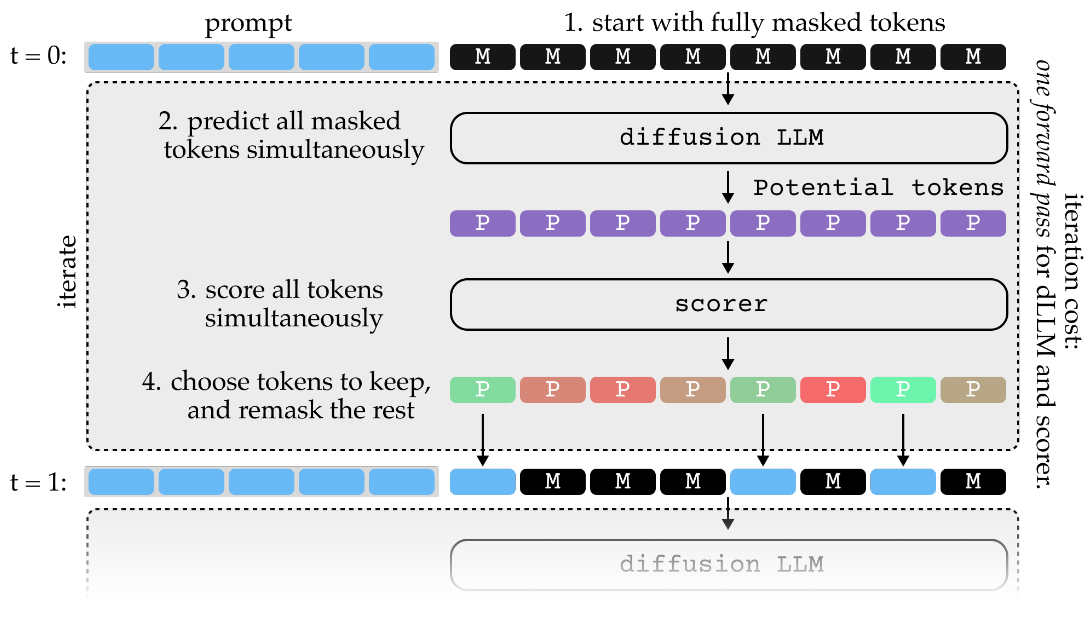
Despite the progress in masked diffusion models, the criterion for unmasking has largely been limited to such internal heuristics (random or the model's own confidence). The novelty of our approach is to introduce a separate scorer model into the unmasking schedule – essentially a secondary model that evaluates partially generated sequences to decide which mask to reveal next. This is conceptually akin to the verifier model in speculative decoding, but applied to a masked parallel decoding context. By leveraging an external scorer to guide token selection, our method departs from prior schedules that are either unguided or self-guided. As we will show, this scorer-based unmasking strategy allows the system to prioritize masks whose resolution will yield the greatest improvement (according to the scorer's judgment), leading to more efficient diffusion iterations. See Figure 2 for a visual representation of this process.

More precisely, our modified algorithm is described in Algorithm 1, and we use the log-probabilities extracted from Qwen-2.5-1.5B-Instruct as our scorer model (Qwen et al., 2025).

4 Results

4.1 Replication

To ensure that we can achieve similar baseline results when applying the default low-confidence unmasking strategy as in (Nie et al., 2025b),



prompt
1. start with fully masked tokens
t = 0:
M M M M M M M M
2. predict all masked tokens simultaneously
diffusion LLM
Potential tokens
P P P P P P P P
3. score all tokens simultaneously
scorer
4. choose tokens to keep, and remask the rest
iterate
t = 1:
iteration cost: one forward pass for dLLM and scorer.

Figure 2: A visual representation of our scorer-based remasking process. Beginning with a sequence of L fully-masked tokens, we first use the LLaDA weights to generate L potential tokens. After scoring all tokens simultaneously using the scorer model, we select the top L/N tokens to keep unmasked and remask the remaining tokens for the next time step. Note that, we swap the "scorer" block for an external autoregressive model, on which we use batch-inference to run in parallel to produce our confidence scores.

we replicated benchmark results on a variety of general-purpose and reasoning tasks, including ARC-Challenge (Clark et al., 2018), Hellaswag (Zellers et al., 2019), TruthfulQA (Lin et al., 2022), PIQA (Bisk et al., 2020), HumanEval (Chen et al., 2021), and GSM8K (Cobbe et al., 2021). In each case, we evaluated LLaDA-8B-Base on a subsample of 200 test-set instances (except for HumanEval, which only had 164 test-set examples in total) using `lm-eval` (Gao et al., 2024); the results of this may be found in Table 1. All values in the table are `pass@1` metrics. With the exception of Hellaswag, our replication was able to achieve accuracy scores within a few percentage points of those reported in the original LLaDA paper, indicating that our evaluation methods are, on the whole, comparable to those conducted internally by Nie et al. We chose not to use the associated LLaDA-8B-Instruct model due to known discrepancies between internal evaluation procedures and the results produced by `lm-eval`; see the Github repository associated with (Nie et al., 2025b) for more information.

See Figures 3 and 4 in Appendix C for a visualization of intermediate steps in the diffusion-based token generation process.

Benchmark	**Reproduction**	**Original**
ARC-C	0.425	0.479
Hellaswag	0.610	0.725
TruthfulQA	0.430	0.464
PIQA	0.730	0.744
HumanEval	0.335	0.335
GSM8K	0.695	0.707

Table 1: Replication of baseline results (in the "Original" column) from (Nie et al., 2025b) obtained via an unpublished internal toolkit. All "Reproduction" results were obtained using the LLaDA-8B-Base model with the low-confidence remasking strategy, setting generation length $L = 256$ and number of steps $N = 256$.

4.2 Impact of Remasking Strategy

We investigated the impact of running different remasking strategies (random remasking, low-confidence and scorer-guided) on the HumanEval benchmark, see Table 2.

The importance of the remasking strategy is demonstrated by the stark performance difference between the random and low-confidence strategies. It is evident that a more informed selection of which tokens to remask leads to a more effective reverse diffusion process. This partially motivates

Algorithm 1 Scorer-Guided Generation [adapted from (Nie et al., 2025b)]

```
Require: mask predictor $p_\theta$, scorer model $q_\phi$, prompt $p_0$, answer length $L$, sampling steps $N$
 1: $r_1 \leftarrow$ fully–masked sequence of length $L$.
 2: for $t \leftarrow 1$ down to $\frac{1}{N}$ step $\frac{1}{N}$ do
 3:     $s \leftarrow t - \frac{1}{N}$
 4:     for $i \leftarrow 1$ to $L$ do
 5:         if $r_t^i \neq \mathtt{M}$ then
 6:             $r_0^i \leftarrow r_t^i$                ▷ token already visible
 7:         else
 8:             $r_0^i \leftarrow \arg\max_x \; p_\theta(x \mid p_0, r_t)$
 9:         end if
10:     end for
11:     for $i \leftarrow 1$ to $L$ do
12:         $c^i \leftarrow q_\phi(r_0^i \mid p_0, r_0)$                ▷ scorer confidence
13:     end for
14:     $n_{\text{un}} \leftarrow \lfloor L(1-s) \rfloor$        ▷ # tokens to remain unmasked at step $s$
15:     for $i \leftarrow 1$ to $L$ do
16:         if $c^i \in \text{Lowest}_{n_{\text{un}}}(\{c^j\}_{j=1}^L)$ then
17:             $r_0^i \leftarrow \mathtt{M}$                ▷ re-mask $n_{\text{un}}$ lowest-scored tokens
18:         end if
19:     end for
20:     $r_s \leftarrow r_0$
21: end for
22: return $r_0$
```

why we considered using a scoring model to further inform our selection of tokens to remask.

However, we noticed that the scorer-guided strategy actually does much worse than the low-confidence strategy, which we attribute to model collapse in the early stages of the generation process. In particular, we observed that in many instances the mask predictor predicts long consecutive runs of whitespace tokens at the start of the process because most of tokens start off as masked. These long runs of whitespace tokens are often evaluated to have high confidence scores by our scorer models; this is because conditioned on a prior speculated run of whitespace, the scorer model tends to evaluate a high likelihood of seeing another whitespace token. Eventually this collapses to simply producing a wholly whitespace output; see Appendix A for an example of this phenomenon.

In order to verify that this is what was occurring within our scorer model, we also examined the scorer model outputs at each iteration of the diffusion process. We observed that as the scorer model evaluates long strings of whitespace or newline (`\n`) tokens, it assigns uniformly high scores to all of the tokens in the string that are usually orders of magnitude larger than those of the non-whitespace tokens generated at the same timestep.

Method	Percentage
Random	10.4%
Low-confidence	33.5%
Scorer-guided	15.2%

Table 2: Performance by remasking strategy

4.3 Ablation Studies

Given that model collapse occurs when exclusively using the scorer-based unmasking strategy across all time steps, we performed an ablation study to better understand combinations of the low-confidence remasking strategy found in (Nie et al., 2025b) with our scorer model. More specifically, we tested benchmark performance for LLaDA-8B-Base when using the low-confidence strategy for the first p proportion of inference steps and the scorer model for the remaining $(1-p)$ proportion of steps. This yielded the results in Table 3.

From these results, we observe performance comparable to LLaDA-8B baselines on code-generation and mathematics benchmarks when using a scorer model-based remasking algorithm for at least 40% to 50% of the total number of inference steps. This is a surprising result—given that outputs to prompts in HumanEval are evaluated by running the generated code against a set of predetermined test cases, an incorrectly generated token at any point in the output string may prevent the resulting code from compiling at all. This further reinforces the notion that as long as a small proportion of tokens are "set" by the default low-confidence remasking process, our scorer model is able to avoid model collapse and fill in the remaining $\sim$60% of tokens at a similar performance level as the default procedure.

5 Discussion

5.1 Conclusion

In conclusion, we have introduced scorer-guided remasking as a novel extension to masked diffu-

Proportion p	HumanEval	GSM8K
(baseline) 1.0	0.335	**0.695**
0.8	0.335	0.690
0.6	**0.341**	**0.695**
0.5	**0.341**	0.675
0.4	0.317	0.615
0.2	0.329	0.500
(only scorer) 0.0	0.152	0.200

Table 3: Ablation results. p represents the proportion of low-confidence remasking steps taken before switching to our scorer-model based remasking strategy. All values are `pass@1` metrics obtained by testing against all 164 examples from HumanEval and a subset of 200 examples from GSM8K, with both sequence length $L = 256$ and number of iterations $N = 256$.

sion language models inspired by the verification models introduced in the speculative decoding literature. In doing so, our work bridges the gap between autoregressive and diffusion-based generative frameworks.

We observed potential issues with using purely scorer-guided remasking; namely that our model collapses to producing repetitve output. We counter this by introducing hybrid remasking strategies that blend scorer-guided remasking with low-confidence remasking. Our experimental results validate the effectiveness of the proposed framework, and that even when incorporating scorer-guided remasking we are able to attain similar accuracy scores on the HumanEval and GSM8K benchmarks (and higher accuracies in some cases).

5.2 Limitations and Future Work

While we have presented first steps towards exploring a novel combination of Transformer-based diffusion and autoregressive models inspired by speculative decoding, much work remains to be done to fully ascertain the potential of this idea.

We observed model collapse when only using scorer-guided remasking, which we averted by performing low-confidence remasking first, and then switching to scorer-guided remasking only after some tokens have been placed in. We suspect that the following adjustments may help prevent model collapse, which may be worth implementing in future studies:

1. To truly realize the power of this system, it should be trained within this setup. A fine-tuning run to effectively "distill" the large autoregressive model into the diffusion "draft" model could lead to massive benefits. We aim to complete this step next.

2. Rather than using Qwen-2.5-1.5B-Instruct as our scorer model, we should use a bidrectional encoder model. This is because in our setting the scorer should assign (in a single forward pass), a well-calibrated confidence score to every visible token in the partially-filled sequence. A bidrectional (masked-language-model) encoder is a better fit than an autoregressive decoder such as Qwen, because they calibrated more closely to the task at hand. We used Qwen because it shares the same tokenization scheme as LLaDA; if we were to use a bidirectional encoder, we would have to match their tokenization schemes.

3. Regularization methods, including penalizing consecutive runs of whitespace when scoring the diffusion model output.

Given the compute limitations inherent to the project and the size of the models we were working with, we were unable to fine-tune either the scorer or the base diffusion models on each others' outputs, nor were we able to fine-tune the scorer model on the unmasking task itself. These remain fruitful directions for future work on this subject, which we intend to pursue.

References

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, et al. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, et al. 2021. Evaluating large language models trained on code.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. The language model evaluation harness.

Inception Labs. 2025. Introducing mercury, the world’s first commercial-scale diffusion language model. https://www.inceptionlabs.ai/introducing-mercury.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. 2025a. Scaling up masked diffusion models on text.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025b. Large language diffusion models.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, et al. 2025. Qwen2.5 technical report.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

A Example of model collapse for HumanEval

When running our pure scorer-guided remasking model on HumanEval, our model often exhibits model collapse and outputs only whitespace. See below.

Prompt

```python
from typing import List

def separate_paren_groups(paren_string: str) -> List[str]:
    """
    Input to this function is a string containing multiple groups of nested parentheses.
    Your goal is to separate those groups into separate strings and return the list of those.
    Separate groups are balanced (each open brace is properly closed) and not nested within each other.
    Ignore any spaces in the input string.

    >>> separate_paren_groups(’( ) (( )) (( )( ))’)
    [’()’, ’(())’, ’(()())’]
    """
    # your implementation here
```

Model Output

```
[’\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n...’]
```

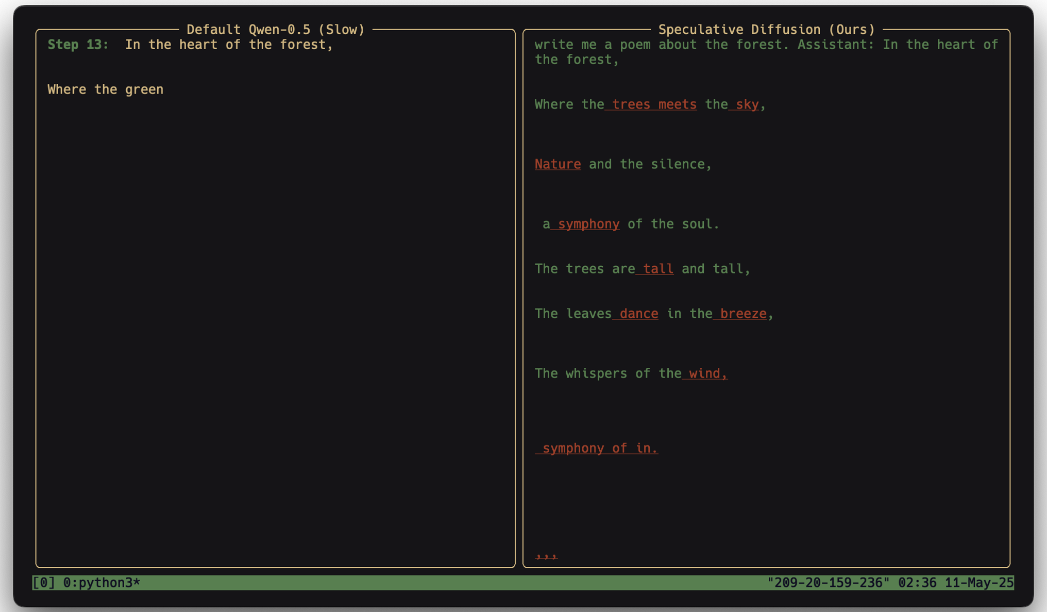
B Scorer model implementation

Below is the PyTorch implementation for our scorer confidence function.

```python
@torch.no_grad()
def scorer_confidence(seq, scorer_model):
    """
    Returns P(seq[i] | seq[:i]) for every i >= 1.

    seq : (B, L)  -- the *filled* candidate sequence.
    out : (B, L)  -- probability of each token under scorer model.
                out[:, 0] is set to 1.0 because the model never
                predicts the first token (it’s conditioned on BOS).
    """
    logits = scorer_model(seq).logits          # (B, L, V)

    # The logit at t predicts token at t, so compare logits[:, t-1] with seq[:, t]
    probs  = torch.softmax(logits, dim=-1).to(torch.float64)  # (B, L, V)

    # Shift: targets are seq[:, 1:], predictors are probs[:, :-1]
    tgt      = seq[:, 1:]                                  # (B, L-1)
    predProb = torch.gather(
                  probs[:, :-1],                           # (B, L-1, V)
                  -1,
                  tgt.unsqueeze(-1)                        # (B, L-1, 1)
              ).squeeze(-1)                                # (B, L-1)

    # Pad the first position (no prediction available) with probability 1
    bos_pad  = torch.ones(seq.size(0), 1, dtype=predProb.dtype,
                          device=predProb.device)
    return torch.cat([bos_pad, predProb], dim=1)   # (B, L)
```

C Example comparisons of scorer-based diffusion models and autogressive models

In Figures 3 and 4, the green text denotes tokens generated in a previous unmasking step, while red text denotes tokens generated by the diffusion model in the current step that are to be evaluated by the scorer model.

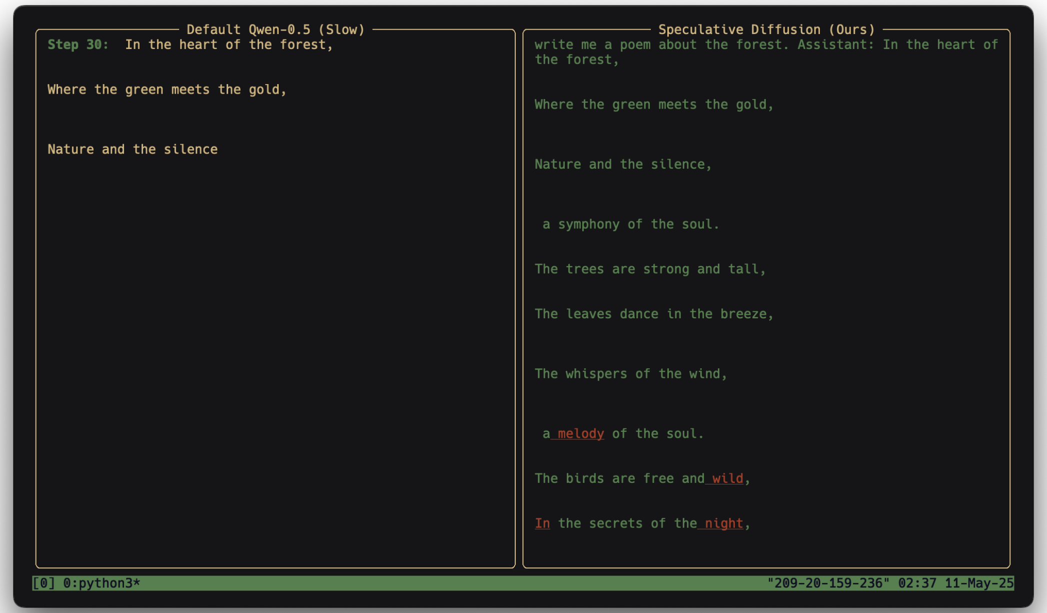


```
Default Qwen-0.5 (Slow)
Step 13:  In the heart of the forest,


Where the green
```

```
Speculative Diffusion (Ours)
write me a poem about the forest. Assistant: In the heart of
the forest,


Where the trees meets the sky,


Nature and the silence,


 a symphony of the soul.

The trees are tall and tall,

The leaves dance in the breeze,


The whispers of the wind,



 symphony of in.




,,,
```

```
[0] 0:python3*                                     "209-20-159-236" 02:36 11-May-25
```

Figure 3: A comparison of our scorer-based remasking method with a sample of where an autoregressive model would be at Step 13 of the iteration process.



```
Default Qwen-0.5 (Slow)
Step 30:  In the heart of the forest,


Where the green meets the gold,


Nature and the silence
```

```
Speculative Diffusion (Ours)
write me a poem about the forest. Assistant: In the heart of
the forest,


Where the green meets the gold,


Nature and the silence,


 a symphony of the soul.

The trees are strong and tall,

The leaves dance in the breeze,


The whispers of the wind,


 a melody of the soul.

The birds are free and wild,

In the secrets of the night,
```

```
[0] 0:python3*                                     "209-20-159-236" 02:37 11-May-25
```

Figure 4: A comparison of our scorer-based remasking method with a sample of where an autoregressive model would be would be at Step 30.