Train for the Worst, Plan for the Best: Understanding Token Ordering in Masked Diffusions

Masked Diffusion Models (MDM)

Consider a distribution p_{data} over sequences of length L with vocab size m, i.e. $\{1, \dots, m\}^L$. Also denote 0 as the "mask".

- **Forward Process** Given $x_0 \sim p_{\text{data}}$, and $t \in [0, 1]$, the forward process is coordinate-independent masking process via $q_{t \mid 0}(x_t \mid x_0) = \Pi_{i=0}^{L-1} q_{t \mid 0}(x_t^i \mid x_0^i)$, where

$$q_{t \mid 0}(x_t^i \mid x_0^i) = \text{Cat}\left(\alpha_t \mathbf{e}_{x_{0^i}} + (1 - \alpha_t)\mathbf{e}_0\right)$$

- **Reverse Process** The reverse process $q_{s \mid t}(x_s \mid x_t, x_0)$ for $s < t$ is given by $q_{s \mid t}(x_s \mid x_t, x_0) = \Pi_{i=0}^{L-1} q_{s \mid t}(x_s^i \mid x_t, x_0)$, where

$$q_{s \mid t}(x_t^i \mid x_t, x_0) = \begin{cases} \text{Cat}\left(\mathbf{e}_{x_t^i}\right) & x_t^i \neq 0 \\ \text{Cat}\left(\frac{1-\alpha_s}{1-\alpha_t}\mathbf{e}_0 + \frac{\alpha_s - \alpha_t}{1-\alpha_t}\mathbf{e}_{x_{t^i}}\right) & x_t^i = 0 \end{cases}$$

In short, in the forward process, with increasing probability, MDM changes a token to the "mask" token, and in the reverse process, MDM removes the "mask" token with increasing probability.

Masked Diffusion Models (MDM)

Note as with the usual diffusion models, in the reverse process, x_0 is approximated by $p_\theta(\cdot \mid x_t, t)$, where p_θ is trained on the marginal distribution on x_0^i. Formally, the training objective of p_θ can be formulated as

$$\mathcal{L}_\theta = \int_0^1 \frac{\alpha_t'}{1-\alpha_t} \mathbb{E}_{x_0 \sim p_{\text{data}}, x_t \sim q_{t \mid 0}(\cdot \mid x_0)} \sum_{i: x_t^i = 0} -\log p_\theta(x_0^i \mid x_t, t) dt,$$

where $\alpha_t' = \frac{d\alpha_t}{dt}$.

MDM vs ARM

- **Complexity at Training Time**: ARM predicts next token, given a unmasked prefix, where as MDM predicts a token conditioned on a set of unmasked token in arbitrary positions.
- **Flexibility at Inference Time**: ARM has fixed left-to-right decoding order, whereas MDM decodes in random order.

Training for the worst

The objective of MDM is equivalent to training ARM on a uniform distribution over all permutations of length L. Hence MDM must solve exponentially many subproblems than ARM.

Training for the worst

Moreover, some permutations are more difficult than others. For example, in natural text, the identity permutation (ARM objective) is easier to solve than random permutation. This could be a possible explanation for why MDM is inferior to ARM in NLP tasks.

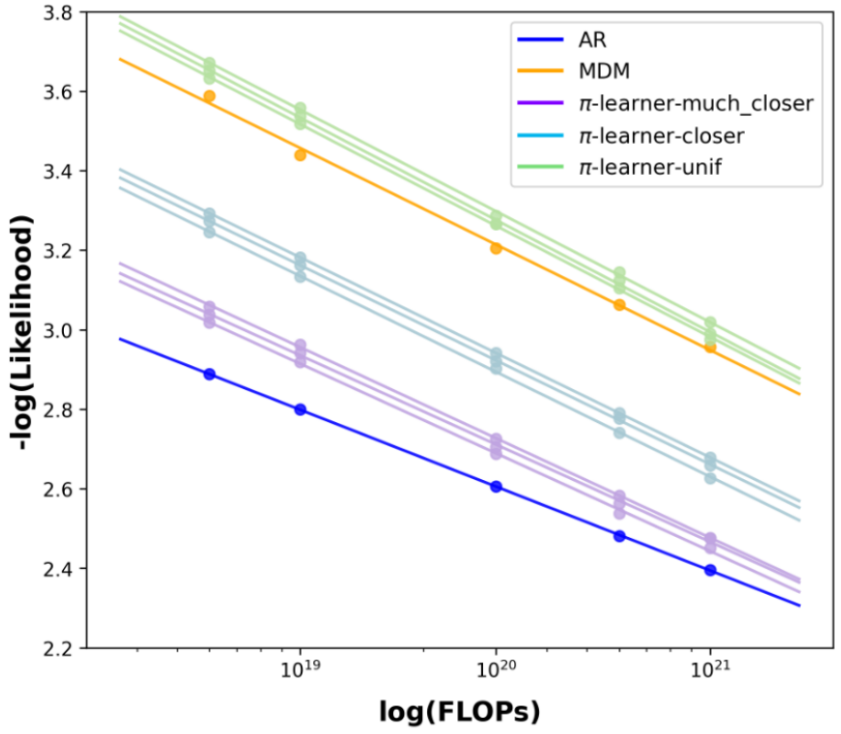

Figure: MDM trained on different problems

Plan for the best

Ideally, if MDM is perfectly trained on all subproblems, token ordering of unmasking results in same sample distribution. However, previously, we have seen that this is not the case in the practice. Then the natural question would be rather than randomly choosing the order, could we choose the *right* order for the inference?

Adaptive Inference

Choose token ordering based on how *certain* model is about each coordinate.

- **Top probability**

$$\max_{j=1,\ldots,m} p_\theta(x^i = j \mid x_t)$$

- **Top probability margin** For most certain tokens j_1, j_2 in top probability sense,

$$|p_\theta(x^i = j_1 \mid x_t) - p_\theta(x^i = j_2 \mid x_t)|$$

Experiments

Table 4. Performance of different inference strategies for LLaDa 8B model on coding and math tasks.

Method	HumanEval-Single	HumanEval-Multi	HumanEval-Split	Math	MMLU	ROCStories
Vanilla	31.8%	16.5%	14.2%	28.5%	33.2%	21.23%
Top probability	32.9%	20.8%	18.4%	31.3%	**36.5%**	21.10%
Top prob. margin	**33.5%**	**25.4%**	**22.3%**	**34.3%**	35.4%	**21.41%**

Figure: NLP tasks on MDM

Experiments

Table 2. Comparison of accuracy for solving the Sudoku puzzle.

Method	**# Param**	**Accuracy**
ARM (w/o ordering)	42M	9.73%
ARM (with ordering)		87.18%
MDM (vanilla)	6M	6.88%
MDM (Top probability)		18.51%
MDM (Top prob. margin)		89.49%

Figure: Sudoku puzzle experiments