REPRESENTATION-SPACE MMD FOR DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce a post-training method for diffusion language models (DLMs) that minimizes Maximum Mean Discrepancy (MMD) between generated and reference distributions in the feature space of a frozen pretrained DLM. To estimate MMD, we retain contextual features at individual token positions, obtaining multiple observations per sequence from a single extractor pass. We optimize this objective using policy gradients for discrete models and direct differentiation through generated latents for continuous models. In both cases, computing the loss directly from these features enables efficient post-training without full sampling trajectories or jointly trained auxiliary models. Experiments show lower generative perplexity at comparable entropy on OpenWebText and better accuracy–computation trade-offs on GSM8K. Applied to 16B DMax-LLaDA2.0 models with hybrid masked–uniform diffusion, the method increases their decoding parallelism while providing similar or higher accuracy on math and code benchmarks.

1 INTRODUCTION

Diffusion language models (DLMs) generate text by refining noisy inputs in a discrete token space or a continuous latent space. In discrete models, token-wise cross-entropy trains a factorized denoiser to fit conditional token marginals. In continuous models, squared-error prediction of clean latents estimates their conditional mean. These conditional estimates guide iterative sampling, but do not necessarily yield high-quality samples in a few steps.

Diffusion distillation addresses this challenge in several ways. Some methods train a student to reproduce several teacher denoising updates in fewer steps (Salimans & Ho, 2022; Luhman & Luhman, 2021; Deschenaux & Gulcehre, 2025; Kim et al., 2025b). Others obtain distribution-matching gradients using an auxiliary model trained on the generator's evolving outputs (Yin et al., 2024b; Hoogeboom et al., 2026; Zheng et al., 2026). These methods improve sampling efficiency but require teacher-derived targets or jointly trained auxiliary models. Therefore, we ask whether a useful distributional training signal can instead be computed directly from generated and reference samples.

Maximum Mean Discrepancy (MMD) offers this possibility through kernel similarities between samples (Gretton et al., 2012). However, its effectiveness depends on the comparison space, since simple distances in high-dimensional spaces need not capture meaningful differences between outputs (Li et al., 2017; 2015). In this context, learned representations offer a promising comparison space for both discrete sequences and continuous outputs. In visual generation, MMD and related objectives in pretrained feature spaces have enabled high-quality generation with one or a few steps (Yang et al., 2026; Starodubcev et al., 2026; Deng et al., 2026). More recently, feature matching has improved LLM fine-tuning by comparing statistics of generated and reference sequences (Jelassi et al., 2026). Together, these results suggest that a fixed pretrained feature space can provide an effective training signal for distribution matching.

Contributions. In this work, we explore MMD for post-training both discrete and continuous DLMs. We use a frozen pretrained DLM to extract contextual features at individual positions, obtaining multiple observations from each generated or reference sequence in a single extractor pass. We then train the generator to match the distributions of these features.

For discrete models, we sample token predictions in parallel and optimize the resulting MMD reward using REINFORCE (Williams, 1992) with a leave-one-out baseline. We instantiate it as

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

MDLM-MMD for masked-token prediction (Sahoo et al., 2024) and as DMax-MMD for hybrid masked–uniform diffusion (Chen et al., 2026b), where MMD trains both masked prediction and refinement of the model's own token predictions.

For continuous DLMs (CDLMs), we initialize a generator from a pretrained Embedded Language Flows (ELF) model (Hu et al., 2026) and train it to map Gaussian noise to clean latent sequences in a single step. MMD gradients propagate through the generated latents and an intermediate layer of the frozen ELF feature extractor. At inference, we use self-conditioning to iteratively refine the single-step predictions, extending the generator to multi-step sampling.

We evaluate both realizations on OpenWebText and TinyGSM, where they outperform the evaluated distribution-matching and flow-map baselines in most setups. At the 16B scale, we show that DMax-MMD improves the accuracy-efficiency trade-off on established math and code benchmarks.

2 PRELIMINARIES

In this section, we briefly review discrete and continuous DLMs and introduce MMD, which forms the basis of our post-training objective. A broader discussion of related work is provided in App. A. Throughout, $\mathbf{x}$ denotes a token sequence for discrete DLMs or a sequence of latent vectors for continuous DLMs. We use $t = 0$ for maximal corruption and $t = 1$ for clean data.

Discrete DLMs corrupt a clean sequence by independently retaining each token with probability α_t and otherwise drawing a replacement from a fixed noise distribution (Austin et al., 2021a). Given the corrupted state $\mathbf{x}_t$, a denoiser $q_\theta(\mathbf{x} \mid \mathbf{x}_t)$ predicts a distribution over clean tokens that factorizes across positions. Therefore, a single forward pass supplies the distributions from which tokens are sampled independently conditional on the input state.

In masked diffusion, the noise distribution is concentrated on an absorbing `[MASK]` symbol, and the denoiser preserves visible tokens (Sahoo et al., 2024; Shi et al., 2024). Pretraining minimizes weighted cross-entropy over masked positions. At inference, generation starts from a fully masked sequence and progressively reveals tokens (Nie et al., 2025). We use confidence-threshold decoding (Wu et al., 2026), which adaptively determines how many tokens are revealed per step.

Uniform diffusion instead draws replacement tokens uniformly from the vocabulary and applies denoising supervision to all positions. Sampling typically starts from a uniformly random sequence and repeatedly updates all positions, allowing earlier predictions to be revised (Austin et al., 2021a).

Continuous DLMs. In this work, we build on the recent Embedded Language Flows (ELF) (Hu et al., 2026), which generate text in the latent space of a frozen encoder. The encoder maps a token sequence $\mathbf{s}$ to latents $\mathbf{x} = E(\mathbf{s}) \in \mathbb{R}^{L \times d}$. Then, ELF learns to predict clean latents from interpolated states $\mathbf{z}_t = t\mathbf{x} + (1-t)\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, using flow matching (Lipman et al., 2023).

To inform each prediction, ELF uses *self-conditioning* (SC), which supplies the denoiser with a previous estimate of the clean latents (Chen et al., 2023). When no estimate is available, this input is $\mathbf{0}$. During standard sampling, the model's predictions define a velocity field that is integrated from noise toward clean latents, with each prediction providing self-conditioning for the next step. The same network then decodes the final latent sequence into tokens.

In our ELF-MMD realization, self-conditioning serves as an effective iterative process to extend the obtained single step generator to multi-step setups.

Maximum Mean Discrepancy. For distributions P and Q and a positive-definite kernel k, squared MMD is (Gretton et al., 2012)

$$\mathrm{MMD}_k^2(P, Q) = \mathbb{E}_{\mathbf{x},\mathbf{x}' \sim P}[k(\mathbf{x}, \mathbf{x}')] + \mathbb{E}_{\mathbf{y},\mathbf{y}' \sim Q}[k(\mathbf{y}, \mathbf{y}')] - 2\mathbb{E}_{\mathbf{x} \sim P, \mathbf{y} \sim Q}[k(\mathbf{x}, \mathbf{y})], \tag{1}$$

where the draws in each expectation are independent. The linear kernel $k(\mathbf{a}, \mathbf{b}) = \mathbf{a}^\top \mathbf{b}$ matches feature means, giving $\mathrm{MMD}_k^2(P, Q) = \|\boldsymbol{\mu}_P - \boldsymbol{\mu}_Q\|_2^2$. The Gaussian RBF kernel $k(\mathbf{a}, \mathbf{b}) = \exp(-\|\mathbf{a} - \mathbf{b}\|_2^2/(2\sigma^2))$, with $\sigma > 0$, is characteristic, so zero MMD identifies equal distributions in the space where the kernel is evaluated. When Q is the generated distribution, its within-distribution term penalizes average RBF similarity between generated features, while the cross term rewards similarity to reference features (Li et al., 2015; Arbel et al., 2019).

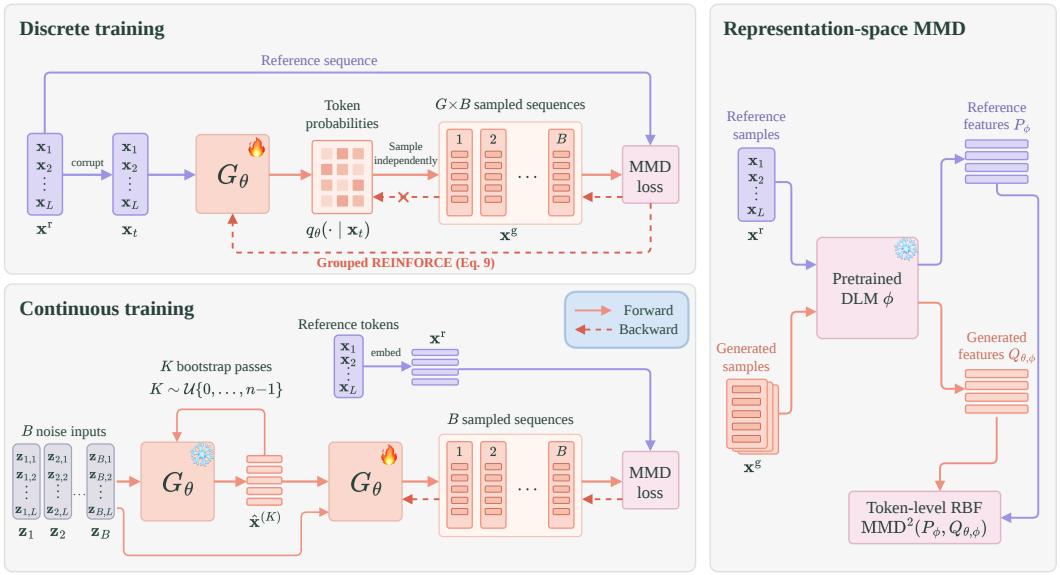

Figure 1: **Representation-space MMD training.** We illustrate the case of one reference sequence and B generated sequences per MMD estimate. For discrete DLMs (*top left*), we independently draw G such batches from the token distributions produced by a single denoiser pass and optimize the loss with grouped REINFORCE. For continuous DLMs (*bottom left*), K bootstrap passes provide self-conditioning without gradients, followed by a differentiable generation pass. In both cases, token-level RBF MMD compares contextual features extracted by a frozen pretrained DLM (*right*).

3 Method

We post-train a DLM by matching contextual token representations of generated and reference sequences. Using the features at individual positions gives us multiple samples for MMD from each extractor evaluation. We first describe the objective and its estimation, then turn to the discrete and continuous realizations.

3.1 Representation-space MMD Training

Choosing a representation space. To compare generated and reference samples with MMD, we need a representation in which to measure their similarity. A pretrained DLM already computes features that describe tokens in their available context, making it a natural choice of feature extractor. Therefore, we use a frozen pretrained DLM and collect its intermediate hidden states,

$$\phi(\mathbf{x}) = \big(\phi_1(\mathbf{x}), \ldots, \phi_L(\mathbf{x})\big) \in \mathbb{R}^{L \times D}, \tag{2}$$

where L is the sequence length and D is the feature dimension.

These contextual features let comparisons at individual positions reflect the surrounding sequence without requiring an externally trained feature model.

In practice, we extract features from clean samples, as corruption at intermediate diffusion timesteps did not improve performance in our experiments. We apply the same extraction rule to real and generated samples and explore alternative extractors in App. C.2.

Using token features for distribution matching. Pooling the features into a single embedding would leave only one observation per sequence. Instead, we retain the individual representations, obtaining $|S|$ feature samples in one forward pass, where S is the nonempty set of scored positions. This provides more observations for MMD estimation, which is especially useful when only a single reference is available for a condition.

We score all positions for unconditional latent generation, response positions for prompted latent generation, and masked positions for masked DLMs. The extractor still processes the available

context, so the selected features can carry information from outside S. Features within a sequence can therefore be dependent, which we account for in the estimator below.

To define their distribution, draw $\mathbf{x} \sim p_{\text{data}}(\cdot \mid c)$ and choose J uniformly from S. Here, c is the generation condition, which may be empty, a prompt, or a corrupted state. We denote the distribution of $\phi_J(\mathbf{x})$ by $P_\phi(\cdot \mid c)$ and define $Q_{\theta,\phi}(\cdot \mid c)$ analogously using generated sequences. We then match these distributions with the Gaussian RBF kernel k from Section 2,

$$\mathcal{L}_{\text{MMD}}(\theta) = \mathbb{E}_c \left[\text{MMD}_k^2(P_\phi(\cdot \mid c), Q_{\theta,\phi}(\cdot \mid c)) \right]. \tag{3}$$

Beyond providing a training signal, feature-space MMD can in principle identify the underlying data distribution. As used in MMD-GAN (Li et al., 2017), applying a characteristic kernel through an injective feature map gives zero population MMD if and only if the original distributions agree. Recent results on causal transformers show that even a last-token representation can be almost surely injective in the discrete input sequence under the stated assumptions (Nikolaou et al., 2026). These results motivate our choice of DLM token features, but do not establish that matching our token-feature distributions uniquely identifies the sequence distribution.

Estimating the training objective. Although the distribution above is defined by sampling one position, a forward pass provides features at every scored position. We use all of them by averaging kernel similarities between two sequences,

$$\kappa_\phi(\mathbf{x}, \mathbf{x}') = \frac{1}{|S||S'|} \sum_{i \in S} \sum_{j \in S'} k(\phi_i(\mathbf{x}), \phi_j(\mathbf{x}')), \tag{4}$$

where S and S' are their scored-position sets. This average is the inner product of empirical kernel mean embeddings and hence a positive-semidefinite kernel on sequences (Muandet et al., 2017, Eq. (3.40-3.41)).

We first describe MMD estimation for a single batch in the unconditional setting. The training objective averages these estimates over batches. MMD requires independent draws in each kernel expectation. Since features within a sequence can be dependent, we form the within-distribution comparisons using different sequences. Consider real and generated batches $\mathbf{X}^{\text{r}} = (\mathbf{x}_b^{\text{r}})_{b=1}^B$ and $\mathbf{X}^{\text{g}} = (\mathbf{x}_b^{\text{g}})_{b=1}^B$, with i.i.d. sequences in each batch and independence between batches. For a fixed extractor and kernel and $B \geq 2$, the usual MMD construction (Gretton et al., 2012) gives

$$\widehat{\mathcal{L}}_{\text{MMD}}(\theta) = \frac{1}{B(B-1)} \sum_{b \neq b'} [\kappa_\phi(\mathbf{x}_b^{\text{r}}, \mathbf{x}_{b'}^{\text{r}}) + \kappa_\phi(\mathbf{x}_b^{\text{g}}, \mathbf{x}_{b'}^{\text{g}})] - \frac{2}{B^2} \sum_{b,b'} \kappa_\phi(\mathbf{x}_b^{\text{r}}, \mathbf{x}_{b'}^{\text{g}}). \tag{5}$$

The restriction $b \neq b'$ excludes all same-sequence token pairs, including pairs at different positions, preserving unbiasedness. Thus, B sequences with $|S|$ scored positions each contribute $B|S|$ feature observations without being treated as $B|S|$ independent draws. Retaining same-sequence pairs generally introduces bias, although these comparisons may still be useful in practice.

The real–generated term encourages similarity to reference features, while the generated–generated term penalizes similarity among generated features. Together, they balance attraction to the reference distribution against concentration around a few representations. The real–real term is independent of θ and is omitted during optimization.

The generated–generated term requires $B \geq 2$. With $B = 1$, retaining only the real–generated term gives an attraction-only objective, which leads to worse performance on discrete models and rapid collapse in continuous models in our ablation study (Section 4.4).

For conditional cases, which are more prevalent in practice, we can train with one reference per condition because the real–real term need not be estimated for optimization. Given $\mathbf{x}^{\text{r}} \sim p_{\text{data}}(\cdot \mid c)$ and $B \geq 2$ generated sequences sampled independently of one another, we estimate the remaining terms using the training loss

$$\widehat{\mathcal{L}}_{\text{train}}(\theta \mid c) = \frac{1}{B(B-1)} \sum_{b \neq b'} \kappa_\phi(\mathbf{x}_b^{\text{g}}, \mathbf{x}_{b'}^{\text{g}}) - \frac{2}{B} \sum_{b} \kappa_\phi(\mathbf{x}^{\text{r}}, \mathbf{x}_b^{\text{g}}). \tag{6}$$

Its expectation over both reference and generated samples equals the conditional squared MMD up to an additive constant independent of θ. We average this loss over conditions, using one reference and B generated sequences for each.

The following subsections describe feature extraction and optimization in each state space. Fig. 1 illustrates the training scheme for both discrete and continuous realizations.

3.2 Discrete Diffusion Language Models

Given a noisy input $\mathbf{x}_t$, we select a nonempty set S of valid positions to predict, restricted to the response for prompted tasks. The denoiser defines a conditional distribution by predicting these positions while keeping the remaining input fixed,

$$q_\theta(\mathbf{x}^{\mathrm{g}} \mid \mathbf{x}_t) = \prod_{i \in S} q_\theta(\mathbf{x}_i^{\mathrm{g}} \mid \mathbf{x}_t), \qquad \mathbf{x}_i^{\mathrm{g}} = \mathbf{x}_{t,i} \quad \text{for } i \notin S. \tag{7}$$

We compare these sampled sequences with a reference $\mathbf{x}^{\mathrm{r}}$ that contains the clean target tokens at S and shares the same context elsewhere. Both are passed through the frozen feature extractor, and their features at S enter the MMD loss. Although the sampled tokens are conditionally independent, their contextual representations allow the loss to assess them jointly.

Optimizing sampled sequences. We use the conditional MMD estimate in Eq. (6), which compares one reference with $B \geq 2$ independently sampled sequences. We draw all B sequences independently from the token distributions produced by a single denoiser forward pass. Categorical sampling prevents direct differentiation through these samples, so we optimize the loss with REINFORCE (Williams, 1992). To reduce gradient variance, we draw a group of G independent MMD batches, each containing B sampled sequences, for the same reference and corrupted state $\mathbf{x}_t$. Each batch $\mathbf{X}_g = (\mathbf{x}_{g,1}, \ldots, \mathbf{x}_{g,B})$ receives the negative MMD estimate as its reward. For $G > 1$, the other batches provide a leave-one-out baseline (Ahmadian et al., 2024),

$$r_g = -\widehat{\mathcal{L}}_{\mathrm{train}}(\mathbf{x}^{\mathrm{r}}, \mathbf{X}_g \mid \mathbf{x}_t), \qquad A_g = r_g - \frac{1}{G-1} \sum_{h \neq g} r_h. \tag{8}$$

We then minimize the policy-gradient surrogate

$$\mathcal{L}_{\mathrm{PG}} = -\frac{1}{G} \sum_{g=1}^{G} A_g \sum_{b=1}^{B} \sum_{i \in S} \log q_\theta(\mathbf{x}_{g,b,i} \mid \mathbf{x}_t). \tag{9}$$

Because a reward depends on an entire batch, it multiplies the sum of log probabilities of all sequences in that batch. The baseline uses independent batches conditional on the shared reference and $\mathbf{x}_t$, so it does not change the expected gradient. For $G = 1$, we use $A_1 = r_1$.

Masked and uniform variants. We consider two realizations of this training procedure. In **MDLM-MMD**, $\mathbf{x}_t$ is formed by masking tokens and S contains only masked positions, so visible tokens remain unchanged. In **DMax-MMD**, we post-train DMax (Chen et al., 2026b), a hybrid masked–uniform model, using MMD for both its masked and prediction losses. The masked loss uses the same choice of S, while the prediction loss uses all target positions and conditions on the model's own token predictions. For block diffusion, we apply the objective within each block conditioned on its available context. After MMD post-training, we use each discrete model's original sampling procedure without any changes.

3.3 Continuous Diffusion Language Models

Continuous latents allow us to optimize the MMD objective by differentiating through generated samples. We instantiate this approach as **ELF-MMD**, initializing G_θ from a pretrained ELF model and training a *one-step* latent generator,

$$\mathbf{x}^{\mathrm{g}} = G_\theta(\mathbf{z}, t = 0, \mathrm{SC} = \mathbf{0}), \qquad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{10}$$

Here, SC denotes the self-conditioning input described in Section 2.

The pretrained ELF model provides the intermediate features used by MMD. We feed each clean real or generated latent sequence $\mathbf{x}$ into both its latent and self-conditioning inputs at $t = 1$,

$$\phi(\mathbf{x}) = \phi_{\mathrm{ELF}}(\mathbf{x}, t = 1, \mathrm{SC} = \mathbf{x}). \tag{11}$$

We use Eq. (5) for unconditional generation and Eq. (6) for prompted generation, where the extractor receives the clean prompt and only response positions enter the kernel. The resulting gradients pass through both extractor inputs to update G_θ.

Iterative refinement through self-conditioning. To extend the obtained model to multi-step generation, we iteratively refine its predictions via self-conditioning. Specifically, starting from $\hat{\mathbf{x}}^{(0)} = \mathbf{0}$, we feed each prediction back into the generator,

$$\hat{\mathbf{x}}^{(k)} = G_\theta\Big(\mathbf{z}^{(k)}, t = 0, \mathrm{SC} = \hat{\mathbf{x}}^{(k-1)}\Big), \qquad k = 1, \ldots, K. \tag{12}$$

The noise is either held fixed across iterations or independently resampled, while the diffusion time remains at $t = 0$. Thus, $K = 1$ recovers the one-step generator. After K refinement steps, one final evaluation at $t = 1$, with null self-conditioning, converts $\hat{\mathbf{x}}^{(K)}$ to token logits. Taking their argmax yields the generated text, for a total of $K + 1$ network evaluations (Algorithm 2).

Bootstrapping and iterative refinement distillation. Later refinement steps receive the model's previous predictions as self-conditioning, which differ from the null input used for one-step generation. To expose the model to these inputs during training, we uniformly sample the number of preliminary refinement steps from $\{0, \ldots, n-1\}$, where n is the maximum number of generator passes including the final differentiable step. We run these preliminary steps with fixed noise and without gradients, then apply MMD to one additional prediction and backpropagate only through this final step.

In addition, to reduce the number of evaluations needed at inference, we optionally apply iterative refinement distillation (IRD) after MMD training. A frozen ELF-MMD teacher supplies the final prediction of a multi-step trajectory, which a student learns to reproduce in one step (Yoo et al., 2026; Luhman & Luhman, 2021). We describe the distillation objective in App. B.1.

We note that the MMD loss is also applicable to the continuous DLM variants that do not use self-conditioning (Lee et al., 2026; Deschenaux & Gulcehre, 2026). Instead of iterative refinement, they could be adapted to use alternative multi-step sampling schemes, such as consistency sampling (Song et al., 2023), that are widely explored for visual generators (Sauer et al., 2024; Yin et al., 2024a).

4 Experiments

In this section, we evaluate whether representation-space MMD improves generation at limited sampling budgets for both discrete and continuous DLMs. We first study unconditional generation on OpenWebText and mathematical reasoning on GSM8K, then test the approach on 16B DMax models for math and code generation.

4.1 Unconditional Generation

For unconditional generation, we follow prior work (Hu et al., 2026; Lee et al., 2026; Chen et al., 2026a) and train on OpenWebText (OWT) (Gokaslan & Cohen, 2019) with sequences packed to length $L = 1024$. We evaluate $1{,}000$ generated samples using generative perplexity (Gen. PPL) under pretrained GPT-2 Large (Radford et al., 2019) and average unigram entropy.

Discrete DLMs. We initialize MDLM-MMD from a pretrained MDLM (Sahoo et al., 2024) and compute MMD rewards using features from a frozen MDLM feature extractor, with policy-gradient group size $G = 4$. We compare against DiDi-Instruct (Zheng et al., 2026), IDLM (Li et al., 2026), and IDLM-REINFORCE, which optimizes the IDLM log-ratio reward using policy gradients. This comparison helps distinguish the effects of the training objective and the optimization procedure. We evaluate each method at 8, 16, and 32 sampling steps, varying the sampling temperature to characterize the trade-off between generative perplexity and sample entropy. As shown in Fig. 2, MDLM-MMD achieves lower generative perplexity than all baselines at matched entropy. In particular, interpolating the curves at the reference-data entropy ($H \approx 5.43$) gives $\sim$17–21$\%$ lower generative perplexity than IDLM across the three sampling budgets. Exact numbers are reported in Tab. 2.

Continuous DLMs. We use the ELF-B architecture with two choices of continuous latent space: the final-layer hidden representations of a T5-small encoder and a GPT-2 Large model, following (Hu et al., 2026; Yoo et al., 2026). For the main OWT and GSM8K comparisons, we apply IRD after MMD post-training and ablate its effect in App. C.5. Fig. 2 compares ELF-MMD with ELF and its self-conditioning distilled variant ELF* (Yoo et al., 2026). We also include progressively distilled ELF (ELF-PD) (Hu et al., 2026) and FMLM* (Yoo et al., 2026) as few-step baselines.

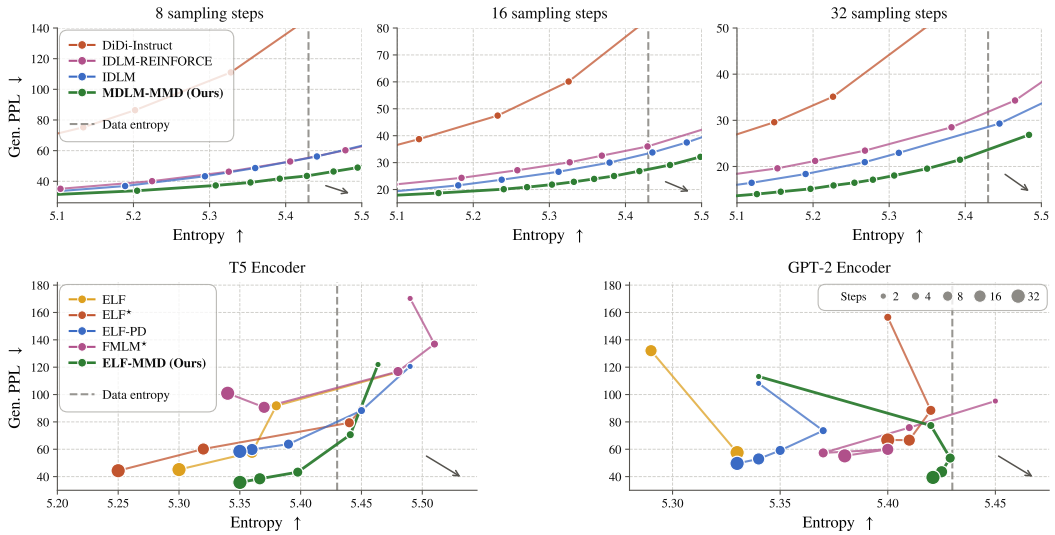

Figure 2: **Gen. PPL-Entropy trade-offs on OpenWebText. Top:** MDLM-MMD and MDLM-based baselines, with temperature sweeps at different sampling steps. **Bottom:** ELF-MMD and continuous baselines, varying the number of sampling steps. Low-entropy ELF and ELF* points are omitted.

ELF-MMD achieves the lowest generative perplexity at comparable entropy across most sampling budgets for both T5 and GPT-2 encoders. In particular, at 8 steps, ELF-MMD improves over ELF-PD by $\sim$20 and $\sim$5 generative perplexity points with the T5 and GPT-2 encoders, respectively, with comparable entropy for T5 and higher entropy for GPT-2. At 32 steps, ELF-MMD outperforms ELF, achieving both lower generative perplexity and higher entropy on both encoders. Exact numbers, together with additional results for MMD post-training of ELF*, are reported in Tab. 3.

4.2 Conditional Generation

For conditional generation, following (Kim et al., 2025a; Agarwal et al., 2026), we train on TinyGSM (Liu et al., 2023) and evaluate final-answer accuracy on the GSM8K test set (Cobbe et al., 2021). We pack sequences to a maximum length of $L = 512$. In addition, we report pass@k for both discrete and continuous DLMs using different numbers of samples per prompt (Fig. 12).

Discrete DLMs. We initialize MDLM-MMD from an MDLM pretrained on TinyGSM and use frozen MDLM features for the MMD objective. We compare against MDLM, IDLM (Li et al., 2026), IDLM-REINFORCE, and DiDi-Instruct (Zheng et al., 2026). Because confidence decoding determines the number of model evaluations adaptively, we sweep the confidence threshold and report the resulting average number of steps per sequence. As shown in Fig. 3, MDLM-MMD achieves higher accuracy at moderate and high decoding budgets, reaching $\sim$54% accuracy at $\sim$49 steps.

Continuous DLMs. Our main comparison uses ELF-B with the final-layer hidden representations of GPT-2 Small (Radford et al., 2019) as the continuous latent space. We additionally evaluate scaling to ELF-M in App. C.6. We compare ELF-MMD with ELF, ELF-PD (Hu et al., 2026), ELF-GAN (see App. E.1 for training details), and FMLM+ (Agarwal et al., 2026), which is initialized from a pretrained masked diffusion model. Applying a shifted timestep schedule (Esser et al., 2024) improves the accuracy of both ELF and ELF-PD across all evaluated step budgets. Thus, we use shift values of 32 and 128, respectively.

As shown in Fig. 3, ELF-MMD achieves the highest accuracy at most sampling budgets. In particular, it reaches 22.18% and 32.70% accuracy at 4 and 8 steps, respectively, compared with 17.26% and 24.73% for our direct baseline, ELF-PD. At 64 steps, ELF-MMD reaches its highest accuracy of 36.4%, $\sim$4 percentage points above ELF at the same budget. Exact results are in Tab. 4.

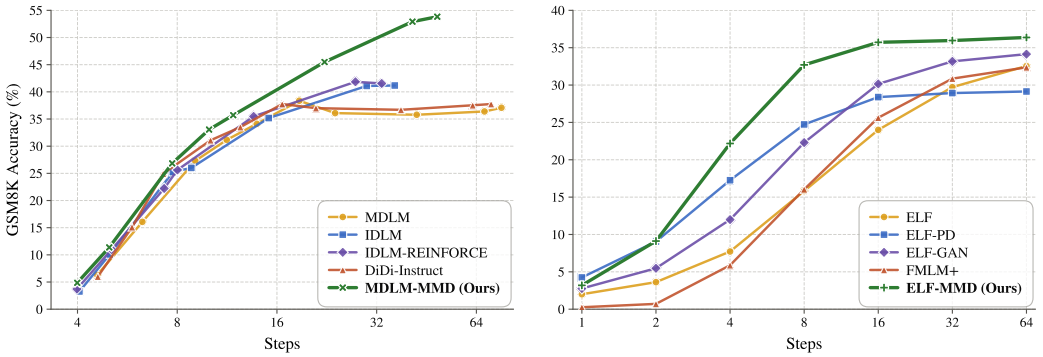

Figure 3: **GSM8K accuracy vs sampling steps. Left:** MDLM-MMD and MDLM-based baselines. **Right:** ELF-MMD and continuous baselines. For masked models, we vary the confidence threshold and report the average number of steps per sequence.

We also apply the MMD loss to ELF* (Yoo et al., 2026) and FMLM+ (Agarwal et al., 2026), yielding ELF*-MMD and FMLM+-MMD. Their post-training details and results are provided in App. E.

Pass@k results. For MDLM-based models, we also evaluate pass@k using ancestral sampling at temperature 0. Fig. 12 shows that MDLM-MMD outperforms the baselines at 4–8 sampling steps for $k \leq 8$. At 16–32 steps, it also achieves higher pass@k for most evaluated values of k, with smaller differences at larger sample budgets. For continuous DLMs, ELF-MMD also improves pass@k at smaller sample budgets, while the gaps narrow as k increases.

4.3 Scaling to Large Models

To test whether MMD post-training extends to larger models, we start from the released 16B DMax-Math and DMax-Coder checkpoints (Chen et al., 2026b). DMax is a hybrid masked–uniform diffusion model tuned from LLaDa2.0-Mini (Bie et al., 2025) that learns to recover target tokens from both masked inputs and its own token predictions. We follow the original training implementation and use MMD to train on both masked inputs and the model's own token predictions, yielding DMax-Math-MMD and DMax-Coder-MMD.

We perform 400 training steps that take $\sim$13 minutes for DMax-Math and $\sim$18 minutes for DMax-Coder on eight NVIDIA H100 GPUs, corresponding to $\sim$1.7 and $\sim$2.4 GPU-hours per run, respectively. These timings cover training only, excluding initialization and evaluation.

We use the released DMax training data, which contain LLaDA-2.0-mini-generated responses to math and code prompts. For math, we follow the original chain-of-thought evaluation on GSM8K, MATH500 (Lightman et al., 2024), Minerva-Algebra (Hendrycks et al., 2021), and ASDIV (Miao et al., 2020). For code, we report pass@1 on HumanEval-Instruct (Chen et al., 2021) and MBPP-Instruct (Austin et al., 2021b). Evaluation uses the original dInfer (Ma et al., 2025) implementation with Soft Parallel Decoding, a block size of 32, and a maximum generation length of 2048 tokens.

We report accuracy and tokens generated per forward (TPF), which measures decoding parallelism. Tab. 1 compares these metrics with the original paper's baselines. We average accuracy and TPF over five seeded runs at each decoding threshold. For each benchmark, we select the highest-TPF point within 0.5 percentage points of peak mean accuracy. Across the four math benchmarks, DMax-Math-MMD increases TPF by 8.8–23.2% over the reported DMax-Math operating points while achieving comparable or higher accuracy. On code benchmarks, DMax-Coder-MMD improves accuracy by 2.6 percentage points on HumanEval-Instruct and 2.8 points on MBPP-Instruct, while also increasing TPF on both benchmarks. To show how this trade-off varies with decoding threshold, we provide accuracy–TPF curves in App. D, together with the selected thresholds and aggregation details.

4.4 MMD Loss Ablation

Finally, we compare alternative loss formulations on GSM8K (Fig. 4), with OWT results in App. C.1. We start with linear-kernel MMD, computed as MSE between generated and reference feature means.

Table 1: **Scaling MMD post-training to DMax.** Accuracy (%) and tokens generated per forward pass (TPF) on math and code benchmarks. Baseline results are taken from the original DMax paper.

Method	GSM8K		MATH500		Minerva-Algebra		ASDIV		HumanEval-Instruct		MBPP-Instruct	
	Acc.↑	TPF↑	Acc.↑	TPF↑	Acc.↑	TPF↑	Acc.↑	TPF↑	Acc.↑	TPF↑	Acc.↑	TPF↑
LLaDA-2.0-mini	**92.6**	2.04	75.8	2.58	91.4	3.01	92.8	2.03	84.2	4.38	80.6	2.71
dParallel SFT	92.3	2.79	75.8	3.42	91.4	3.91	**93.0**	2.72	76.8	5.12	74.7	3.66
DMax-Math	92.1	5.48	75.4	5.94	91.5	7.03	92.5	5.62	–	–	–	–
DMax-Math-MMD	92.1	**6.60**	**77.6**	**6.46**	**92.8**	**7.92**	92.4	**6.93**	–	–	–	–
DMax-Coder	–	–	–	–	–	–	–	–	83.5	7.36	79.2	5.86
DMax-Coder-MMD	–	–	–	–	–	–	–	–	**86.1**	**7.98**	**82.0**	**6.17**

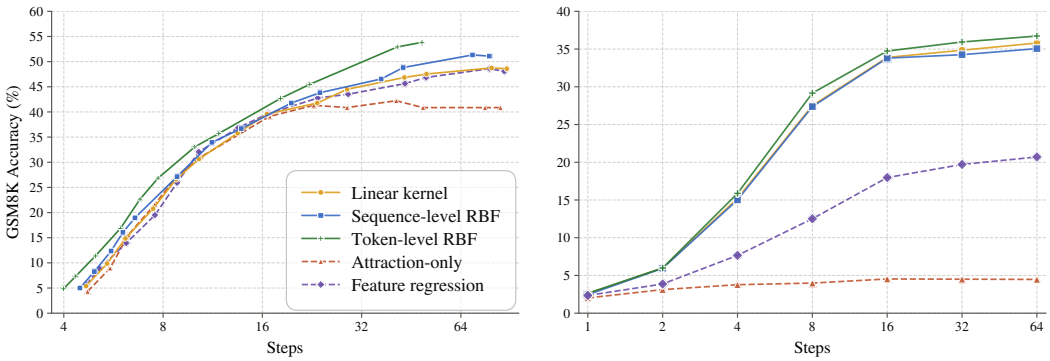

Figure 4: **MMD loss ablation on GSM8K.** Accuracy versus sampling steps for MDLM (*left*) and ELF (*right*), post-trained with different feature-based objectives.

This baseline tests whether matching first moments is sufficient and connects to feature matching for LLMs (Jelassi et al., 2026). Beyond mean matching, we compare sequence-level RBF on mean-pooled embeddings with our token-level RBF. This aims to address whether retaining token-level interactions improves the MMD training signal.

The attraction-only variant removes the generated–generated repulsive term to test whether similarity to reference features alone suffices. Finally, a feature regression baseline minimizes MSE between the full intermediate feature tensors of sampled and reference sequences at corresponding positions.

On GSM8K, token-level RBF gives the strongest accuracy–computation trade-off, with larger gains for discrete models and more modest improvements for continuous models. Attraction-only and feature regression losses yield lower accuracy, particularly for continuous models. The OWT results in Fig. 5 similarly favor token-level RBF for both discrete and continuous models in most setups. These results support our choice of token-level RBF.

5 Discussion

Our results show that MMD on pretrained DLM features can improve generation in both discrete and continuous DLMs. These improvements are obtained with a simple and efficient training procedure that requires neither full sampling trajectories nor jointly trained auxiliary models. Nevertheless, performance depends on choices such as the RBF bandwidth and the representation space used for distribution matching. This makes representation design a natural direction for further exploration. We use features from a single layer on clean inputs, but combining features across layers or noise levels, or using an ensemble of extractors, could provide complementary information.

Alongside representation design, future work could explore combining MMD with other training objectives. We deliberately exclude additional regularization or distillation losses during MMD post-training to study its contribution to generation quality and efficiency, but such combinations may offer further benefits in practice.

AI USE STATEMENT

We used AI tools for literature search, citation verification, and assistance with manuscript organization, drafting, editing, and formatting. We also used them to review the method, mathematical claims, and implementation, and to help collect and interpret results and prepare figures. We take responsibility for the final content of the paper, including all AI-assisted contributions.

REPRODUCIBILITY STATEMENT

We describe the objective and optimization procedures in Section 3, with pseudocode for continuous-model training, sampling, and optional distillation in Algorithms 1, 2, and 3. Section 4 and App. D describe the datasets, models, and evaluation protocols. The corresponding training hyperparameters are listed in Tabs. 5, 6, and 7.

REFERENCES

Manan Agarwal, Sheel Shah, Chanhyuk Lee, Jaehoon Yoo, Jerry Huang, Seunghoon Hong, Aditi Raghunathan, Jinwoo Kim, and Nicholas M. Boffi. Posterior refinement: Fast language generation via any-order flow maps, 2026. URL https://arxiv.org/abs/2606.24773.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.662. URL https://aclanthology.org/2024.acl-long.662/.

Emanuele Aiello, Diego Valsesia, and Enrico Magli. Fast inference in denoising diffusion models via MMD finetuning. *IEEE Access*, 12:106912–106923, 2024. doi: 10.1109/ACCESS.2024.3436698. URL https://doi.org/10.1109/ACCESS.2024.3436698.

Michael Arbel, Anna Korba, Adil Salim, and Arthur Gretton. Maximum mean discrepancy gradient flow. *Advances in neural information processing systems*, 32, 2019.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, pp. 17981–17993, 2021a.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021b. URL https://arxiv.org/abs/2108.07732.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, Jianfeng Tan, Lanning Wei, Ji-Rong Wen, Yipeng Xing, Xiaolu Zhang, Junbo Zhao, Da Zheng, Jun Zhou, Junlin Zhou, Zhanchao Zhou, Liwang Zhu, and Yihong Zhuang. LLaDA2.0: Scaling up diffusion language models to 100B, 2025. URL https://arxiv.org/abs/2512.15745.

Nicholas Matthew Boffi, Michael Samuel Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=cqDH0e6ak2.

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, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios

Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Tianqi Chen, Shujian Zhang, and Mingyuan Zhou. DLM-One: Diffusion language models for one-step sequence generation, 2025. URL https://arxiv.org/abs/2506.00290.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=3itjR9QxFw.

Yuxin Chen, Chumeng Liang, Hangke Sui, Ruihan Guo, Chaoran Cheng, Jiaxuan You, and Ge Liu. LangFlow: Continuous diffusion rivals discrete in language modeling, 2026a. URL https://arxiv.org/abs/2604.11748.

Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. Dmax: Aggressive parallel decoding for dllms, 2026b. URL https://arxiv.org/abs/2604.08302.

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. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative modeling via drifting, 2026. URL https://arxiv.org/abs/2602.04770.

Justin Deschenaux and Caglar Gulcehre. Beyond autoregression: Fast LLMs via self-distillation through time. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=uZ5K4HeNwd.

Justin Deschenaux and Caglar Gulcehre. Language modeling with hyperspherical flows, 2026. URL https://arxiv.org/abs/2605.11125.

Cicero Nogueira dos Santos, Youssef Mroueh, Inkit Padhi, and Pierre Dognin. Learning implicit generative models by matching perceptual features. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

Yonggan Fu, Lexington Whalen, Abhinav Garg, Chengyue Wu, Maksim Khadkevich, Nicolai Oswald, Enze Xie, Daniel Egert, Sharath Turuvekere Sreenivas, Shizhe Diao, Chenhan Yu, Ye Yu, Weijia Chen, Sajad Norouzi, Jingyu Liu, Shiyi Lan, Ligeng Zhu, Jin Wang, Jindong Jiang, Morteza Mardani, Mehran Maghoumi, Song Han, Ante Jukić, Nima Tajbakhsh, Jan Kautz, and Pavlo Molchanov. Nemotron-Labs-Diffusion: A tri-mode language model unifying autoregressive, diffusion, and self-speculation decoding, 2026. URL https://arxiv.org/abs/2607.05722.

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus, 2019.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander J. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012. URL https://www.jmlr.org/papers/v13/gretton12a.html.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions, 2021. URL https://arxiv.org/abs/2102.05379.

Emiel Hoogeboom, David Ruhe, Jonathan Heek, Thomas Mensink, and Tim Salimans. Beyond single tokens: Distilling discrete diffusion models via discrete MMD, 2026. URL https://arxiv.org/abs/2603.20155.

Keya Hu, Linlu Qiu, Yiyang Lu, Hanhong Zhao, Tianhong Li, Yoon Kim, Jacob Andreas, and Kaiming He. ELF: Embedded language flows. *arXiv preprint arXiv:2605.10938*, 2026. URL https://arxiv.org/abs/2605.10938.

Samy Jelassi, Mujin Kwun, Rosie Zhao, Yuanzhi Li, Nicolo Fusi, Yilun Du, Sham M Kakade, and Carles Domingo-Enrich. Matching features, not tokens: Energy-based fine-tuning of language models. *arXiv preprint arXiv:2603.12248*, 2026.

Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. *arXiv preprint arXiv:2502.06768*, 2025a.

Minseo Kim, Chenfeng Xu, Coleman Hooper, Harman Singh, Ben Athiwaratkun, Ce Zhang, Kurt Keutzer, and Amir Gholami. Cdlm: Consistency diffusion language models for faster sampling. *arXiv preprint arXiv:2511.19269*, 2025b. URL https://arxiv.org/abs/2511.19269.

Chanhyuk Lee, Jaehoon Yoo, Manan Agarwal, Sheel Shah, Jerry Huang, Aditi Raghunathan, Seunghoon Hong, Nicholas M. Boffi, and Jinwoo Kim. Flow map language models: One-step language modeling via continuous denoising, 2026. URL https://arxiv.org/abs/2602.16813.

Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. MMD GAN: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/dfd7468ac613286cdbb40872c8ef3b06-Abstract.html.

David Li, Nikita Gushchin, Dmitry Abulkhanov, Eric Moulines, Ivan Oseledets, Maxim Panov, and Alexander Korotin. IDLM: Inverse-distilled diffusion language models, 2026. URL https://arxiv.org/abs/2602.19066.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-LM improves controllable text generation. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/1be5bc25d50895ee656b8c2d9eb89d6a-Abstract-Conference.html.

Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1718–1727. PMLR, 2015. URL https://proceedings.mlr.press/v37/li15.html.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Bingbin Liu, Sebastien Bubeck, Ronen Eldan, Janardhan Kulkarni, Yuanzhi Li, Anh Nguyen, Rachel Ward, and Yi Zhang. Tinygsm: achieving> 80% on gsm8k with small language models. *arXiv preprint arXiv:2312.09241*, 2023.

Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q. Weinberger. Latent diffusion for language generation. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-2492. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/b2a2bd5d5051ff6af52e1ef60aefd255-Abstract-Conference.html.

Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021.

Yuxin Ma, Lun Du, Lanning Wei, Kun Chen, Qian Xu, Kangyu Wang, Guofeng Feng, Guoshan Lu, Lin Liu, Xiaojing Qi, et al. dinfer: An efficient inference framework for diffusion language models. *arXiv preprint arXiv:2510.08666*, 2025.

Viacheslav Meshchaninov, Egor Chimbulatov, Alexander Shabalin, Aleksandr Abramov, and Dmitry P Vetrov. Cosmos: Compressed and smooth latent space for text diffusion modeling. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0479. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/1506930fb75c82246e4d8648a66e4b27-Abstract-Conference.html.

Viacheslav Meshchaninov, Alexander Shabalin, Egor Chimbulatov, Nikita Gushchin, Ilya Koziev, Alexander Korotin, and Dmitry Vetrov. How to train your latent diffusion language model jointly with the latent space, 2026. URL https://arxiv.org/abs/2605.07933.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th annual meeting of the Association for Computational Linguistics*, pp. 975–984, 2020.

Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, and Bernhard Schölkopf. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends in Machine Learning*, 10(1–2):1–141, 2017. doi: 10.1561/2200000060.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 56354–56392, 2025. doi: 10.52202/085713-1689. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/48b383b24230e0e6e649d9c98dae4d8c-Abstract-Conference.html.

Giorgos Nikolaou, Tommaso Mencattini, Donato Crisostomi, Andrea Santilli, Yannis Panagakis, and Emanuele Rodolà. Language models are injective and hence invertible, 2026. URL https://arxiv.org/abs/2510.15511.

Inkit Padhi, Pierre Dognin, Ke Bai, Cicero dos Santos, Vijil Chenthamarakshan, Youssef Mroueh, and Payel Das. Learning implicit text generation via feature matching. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 3855–3863, 2020.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-4135. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/eb0b13cc515724ab8015bc978fdde0ad-Abstract-Conference.html.

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin T Chiu, and Volodymyr Kuleshov. The diffusion duality. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=9P9Y8FOSOk.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *Computer Vision – ECCV 2024*, volume 15144 of *Lecture Notes in Computer Science*, pp. 87–103. Springer, 2024. doi: 10.1007/978-3-031-73016-0_6. URL https://doi.org/10.1007/978-3-031-73016-0_6.

Yair Schiff, Subham Sekhar Sahoo, Hao Phung, Guanghan Wang, Sam Boshar, Hugo Dalla-torre, Bernardo P. de Almeida, Alexander Rush, Thomas Pierrot, and Volodymyr Kuleshov. Simple guidance mechanisms for discrete diffusion models, 2025. URL https://arxiv.org/abs/2412.10193.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. In *Advances in Neural Information Processing Systems*, volume 37, pp. 103131–103167, 2024.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 32211–32252. PMLR, 2023. URL https://proceedings.mlr.press/v202/song23a.html.

Nikita Starodubcev, Ilya Drobyshevskiy, Denis Kuznedelev, Artem Babenko, and Dmitry Baranchuk. Scale-wise distillation of diffusion models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Z06LNjqU1g.

DiffusionGemma Team, Adrien Ali Taïga, James Assiene, Daniele Calandriello, Rahma Chaabouni, João Gante, Tamara von Glehn, Nate Keating, Chris Knutsen, Martin Kukla, Tianlin Liu, Ivan Lobov, Ofir Nabati, João Gabriel Oliveira, Nicolas Perez-Nieves, Nastasia Prutianova, Bobak Shahriari, Jean Tarbouriech, Pavel Tyletski, Çağlar Ünlü, Cindy Wu, Glenn Cameron, Jerome Connor, Sertan Girgin, Maarten Grootendorst, Alon Levkovitch, Eliya Nachmani, Omar Sanseviero, Piotr Stanczyk, Quentin Berthet, Andrew Campbell, Clément Crepy, Valentin De Bortoli, Arnaud Doucet, Romuald Elie, Alexandre Galashov, Klaus Greff, Alexis Jacq, David Ruhe, Yu-Han Wu, Sebastian Flennerhag, Brendan O'Donoghue, George Scrivener, and Shantanu Thakoor. DiffusionGemma technical report, 2026. URL https://arxiv.org/abs/2608.00146.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992. doi: 10.1007/BF00992696.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 57027–57051, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/5d8d4e6061c3ba96c240b7fa1ae3471d-Paper-Conference.pdf.

Jiawei Yang, Zhengyang Geng, Xuan Ju, Yonglong Tian, and Yue Wang. Representation fréchet loss for visual generation, 2026. URL https://arxiv.org/abs/2604.28190.

Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Frédo Durand, and William T. Freeman. Improved distribution matching distillation for fast image synthesis. In *Advances in Neural Information Processing Systems*, volume 37, 2024a. doi: 10.52202/079017-1505. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/54dcf25318f9de5a7a01f0a4125c541e-Abstract-Conference.html.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T. Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6613–6623, 2024b. doi: 10.1109/CVPR52733.2024.00632. URL https://doi.org/10.1109/CVPR52733.2024.00632.

Jaehoon Yoo, Wonjung Kim, Floor Eijkelboom, Chanhyuk Lee, Nicholas M. Boffi, Seunghoon Hong, and Jinwoo Kim. Self-conditioned flow map language models via fixed-point flows, 2026. URL https://arxiv.org/abs/2607.00714.

Haoyang Zheng, Xinyang Liu, Xiangrui Kong, Nan Jiang, Zheyuan Hu, Weijian Luo, Wei Deng, and Guang Lin. Ultra-fast language generation via discrete diffusion divergence instruct. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/323880c576dc7cdcb1fcc7432447a3df-Abstract-Conference.html.

Linqi Zhou, Stefano Ermon, and Jiaming Song. Inductive moment matching. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 78651–78686. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhou25c.html.

A Related Work

Diffusion language models. Diffusion language modeling can be formulated over discrete tokens or continuous representations of text. Masked models such as MDLM (Sahoo et al., 2024) and MD4 (Shi et al., 2024) learn to recover missing tokens, a formulation that also underlies many large discrete DLMs (Nie et al., 2025; Bie et al., 2025; Fu et al., 2026). Uniform DLMs instead replace tokens with uniformly sampled vocabulary items during corruption (Hoogeboom et al., 2021; Schiff et al., 2025; Sahoo et al., 2025). Their reverse process can revise token identities throughout denoising, as in DiffusionGemma (Team et al., 2026). Continuous models denoise token embeddings (Li et al., 2022; Chen et al., 2026a; Deschenaux & Gulcehre, 2026), encoder latents (Lovelace et al., 2023; Meshchaninov et al., 2025; 2026; Hu et al., 2026), or noisy one-hot representations (Lee et al., 2026; Agarwal et al., 2026). Our continuous realization builds on ELF (Hu et al., 2026), which learns a flow in a frozen contextual embedding space.

One- and few-step continuous generation. One way to reduce sampling cost is to train a model to reproduce several denoising steps at once. Direct and progressive distillation (Luhman & Luhman, 2021; Salimans & Ho, 2022) learn from teacher-generated targets, while consistency models (Song et al., 2023) and flow-map matching (Boffi et al., 2025) provide related ways to learn large transitions through the generative process. For language, ELF applies progressive distillation, and FMLM (Lee et al., 2026) learns the flow map of a continuous language model. FMLM+ (Agarwal et al., 2026) extends FMLM with masking-style noise schedules and posterior-guided refinement. FMLM* (Yoo et al., 2026) combines fixed-point and flow-map distillation for self-conditioned models. DLM-One (Chen et al., 2025) takes a distribution-matching approach based on score distillation and adversarial regularization. Our MMD stage trains a latent generator through comparisons with reference samples in a fixed feature space. We use self-conditioning to refine its predictions at a fixed diffusion time and optionally distill these refinement steps afterward.

Few-step discrete DLMs. SDTT (Deschenaux & Gulcehre, 2025) accelerates discrete DLMs by training a student to match token distributions collected across several teacher denoising steps. Inspired by consistency models (Song et al., 2023), CDLM (Kim et al., 2025b) combines teacher-logit distillation with consistency regularization across discrete denoising states. Other approaches construct distribution-matching objectives using learned auxiliary models. IDLM (Li et al., 2026) and Discrete Moment Matching Distillation (D-MMD) (Hoogeboom et al., 2026) train an auxiliary denoiser and update the generator by differentiating through soft token predictions. DiDi-Instruct (Zheng et al., 2026) instead uses policy gradients with a learned density-ratio discriminator. Our discrete formulation also uses policy gradients, with rewards computed directly from MMD in frozen features.

Distribution matching in representation space. MMD has long been used to train implicit generators from samples (Li et al., 2015). Its effectiveness depends on the representation in which samples are compared. MMD-GAN (Li et al., 2017) learns this representation adversarially, whereas GFMN (dos Santos et al., 2019) matches feature statistics from frozen pretrained networks. For diffusion models, MMD-DDM (Aiello et al., 2024) fine-tunes samplers for a fixed step budget. SwD (Starodubcev et al., 2026) brings these ideas together through patch-level MMD on pretrained diffusion features, directly motivating our use of contextual token features. Inductive Moment Matching (Zhou et al., 2025) uses MMD to enforce distributional consistency across sampling intervals. Related objectives include Representation Fréchet Loss (Yang et al., 2026), which compares feature means and covariances, and Drifting Models (Deng et al., 2026), which regress to targets defined by an attraction–repulsion field.

Feature matching for language. Frozen-feature matching has also been explored directly for text generation. SeqGFMN (Padhi et al., 2020) matches position-wise means and variances of token features and differentiates through soft token outputs. More recently, Energy-Based Fine-Tuning (EBFT) (Jelassi et al., 2026) fine-tunes autoregressive models using frozen features of short rollouts and policy-gradient updates. Its basic, unwhitened objective corresponds to conditional linear-kernel MMD, with whitening and reward modifications in the practical training recipe. EBFT represents each rollout by a single feature vector. We instead compare distributions of token features using an RBF kernel, applying the objective to parallel discrete predictions or continuous latent samples.

B CONTINUOUS DLMS TRAINING DETAILS

B.1 ITERATIVE REFINEMENT DISTILLATION

While bootstrapping aims to mitigate the training–inference mismatch for iterative refinement, we also aim to further reduce the number of refinement steps. To this end, inspired by (Yoo et al., 2026), we employ *Iterative Refinement Distillation* (IRD), which adapts a simple Knowledge Distillation (KD) approach (Luhman & Luhman, 2021), i.e., distills multiple refinement iterations into a single prediction.

Specifically, we freeze the trained ELF-MMD generator as a teacher $G_{\bar{\theta}}$ and generate a fixed K-step trajectory using the same noise at every iteration:

$$\bar{\mathbf{x}}^{(0)} = \mathbf{0}, \qquad \bar{\mathbf{x}}^{(k)} = G_{\bar{\theta}}\Big(\mathbf{z}, t{=}0; \mathrm{SC}{=}\bar{\mathbf{x}}^{(k-1)}\Big), \qquad k = 1, \dots, K, \qquad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{13}$$

Using the same noise $\mathbf{z}$, we train the student with $SC{=}0$ to predict the final teacher output:

$$\mathcal{L}_{\mathrm{IRD}}(\theta) = \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\left\| G_\theta(\mathbf{z}, t{=}0; \mathrm{SC}{=}\mathbf{0}) - \bar{\mathbf{x}}^{(K)} \right\|_2^2\right]. \tag{14}$$

Thus, the student is trained to reproduce the output obtained after K teacher refinement steps. The pseudocode is presented in Algorithm 3. We also experimented with more sophisticated approaches, e.g., consistency distillation (Song et al., 2023) or the method used in FMLM* (Yoo et al., 2026) but did not observe noticeable improvements.

IRD is an optional trajectory-supervised stage after MMD training. The MMD stage itself does not use paired teacher trajectories. At inference, the distilled generator can again be used with iterative self-conditioning.

B.2 DECODER FINE-TUNING

Alongside MMD post-training, we fine-tune the model in decoding mode to reconstruct the original tokens from perturbed data latents using a cross-entropy loss, following ELF (Hu et al., 2026). We increase the global batch size by $\sim 20\%$ to include additional samples for decoder fine-tuning.

C ABLATION STUDY

C.1 MMD LOSS ABLATION ON OWT

Fig. 5 extends the comparisons in Section 4.4 to unconditional generation. We compare linear-kernel feature matching and sequence and token level RBF formulations, attraction-only and feature regression losses.

Discrete DLMs. At all three sampling budgets, token-level RBF yields the lowest generative perplexity at matched entropy, followed by sequence-level RBF and the linear kernel. Attraction-only and feature regression losses substantially worsen this trade-off, reinforcing the GSM8K evidence for retaining token features and the generated–generated term.

Continuous DLMs. The effect of kernel choice and feature aggregation depends on the latent space. With T5 latents, token-level RBF achieves a better perplexity–entropy trade-off than both linear and sequence-level RBF kernels. With GPT-2 latents, it provides a modest improvement over the linear kernel. We omit the attraction-only and feature-regression baselines because both lead to collapse.

C.2 REPRESENTATION SPACES

Here, we address whether intermediate DLM features improve MMD training over input representations, and whether separately pretrained encoders offer a suitable alternative. Fig. 6 compares these choices on GSM8K using MDLM and ELF, with setups tuned separately for each representation type.

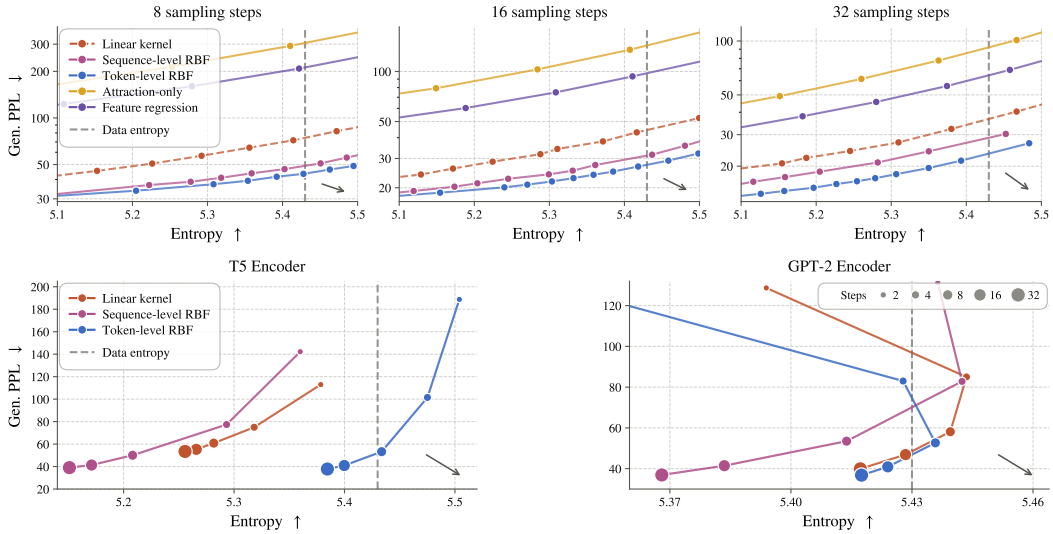

Figure 5: **MMD loss ablation on OpenWebText.** Attraction-only and feature regression losses are shown only for MDLM, as both perform poorly on ELF in the OWT experiments.

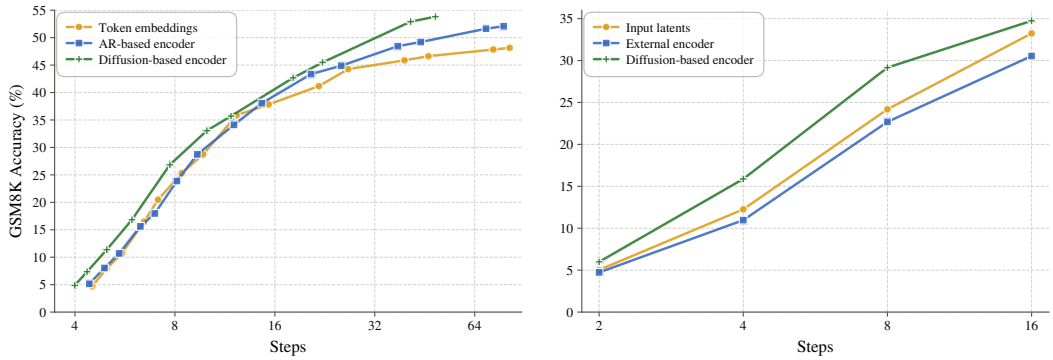

Figure 6: **Representation-space ablation on GSM8K.** Accuracy vs sampling steps for MDLM (*left*) and ELF (*right*), comparing input representations, external encoders, and pretrained DLM features.

C.2.1 Input Representations

Discrete DLMs. We compute MMD directly on token embeddings from the frozen MDLM's vocabulary embedding matrix. Unlike intermediate hidden states, these embeddings depend only on token identity and do not capture the surrounding context.

Continuous DLMs. We compute MMD directly in the latent space of the pretrained text encoder, using the identity feature map $\phi(x) = x$. These latents already encode contextual information, allowing distribution matching without an additional feature extractor.

C.2.2 External Encoder

Discrete DLMs. We use a frozen autoregressive language model pretrained on the same task. We extract contextual token features from an intermediate layer for both generated and reference sequences and compute MMD over these features.

Continuous DLMs. Inspired by the latent-space MAE encoder in Drifting Models (Deng et al., 2026), we train a separate Transformer on real ELF latents using masked-token cross-entropy rather than latent reconstruction.

Results. Diffusion-based features achieve the highest accuracy for both discrete and continuous DLMs. They outperform token embeddings and autoregressive features in the discrete setting, as well as input latents and external-encoder features in the continuous setting.

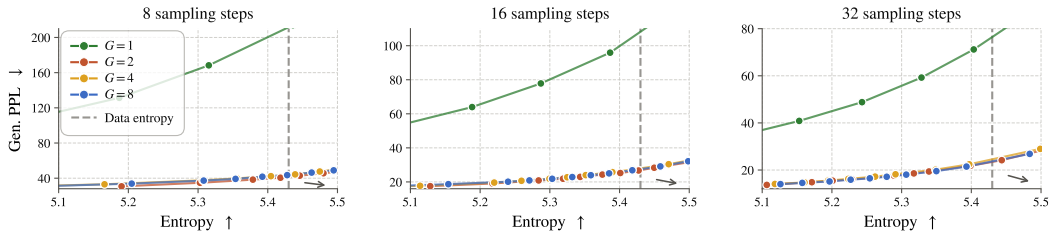

Figure 7: **Ablation on policy-gradient group size on OWT.** Generative perplexity–entropy trade-offs at 8, 16, and 32 sampling steps. Increasing G from 1 to 2 yields the largest improvement.

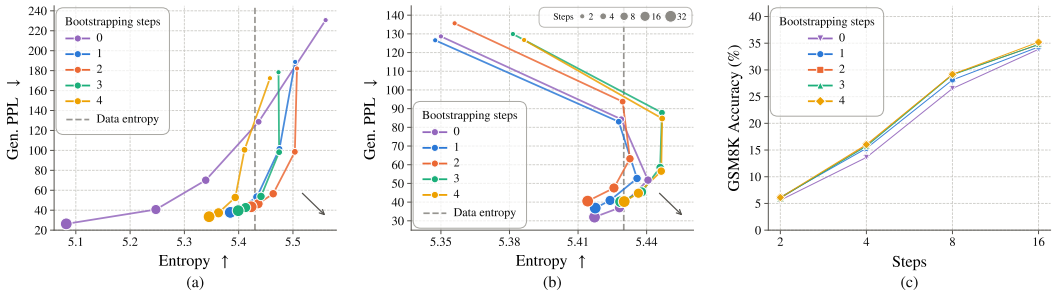

Figure 9: **Ablation on bootstrap training.** We vary the number of bootstrap steps on OWT with T5 (a) and GPT-2 (b) encoders, and on GSM8K (c). A single bootstrap step improves the trade-off with T5 and accuracy on GSM8K, while results with GPT-2 are mixed.

C.3 Policy-gradient Group Size

We vary the group size used for policy-gradient estimation (Section 3.2), $G \in \{1, 2, 4, 8\}$ on OpenWebText and GSM8K. Fig. 7 shows the generative perplexity–entropy trade-offs on OWT at 8, 16, and 32 sampling steps, while Fig. 8 shows answer accuracy versus the average number of sampling steps on GSM8K. The largest improvement occurs when increasing G from 1 to 2, which enables the leave-one-out baseline. Further increases yield negligible gains: $G \in \{2, 4, 8\}$ produce similar trade-offs. We therefore adopt $G = 4$ as a shared default and keep it fixed across all other discrete MMD post-training experiments on OWT and GSM8K.

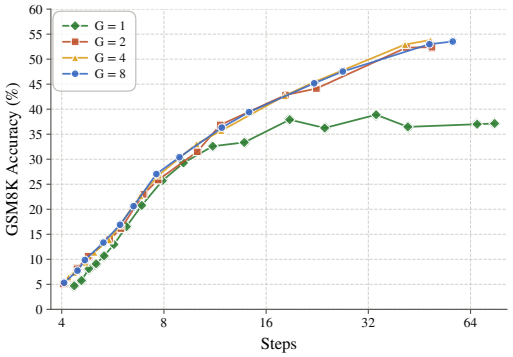

Figure 8: **Ablation on policy-gradient group size on GSM8K.** Accuracy vs. average number of sampling steps.

C.4 Bootstrapping

As discussed in Section 3.3, bootstrapping helps reduce the mismatch between training and inference. We vary the number of bootstrap steps during generator training, including a baseline without bootstrapping. As shown in Fig. 9, a single bootstrap step already provides performance gains, while additional steps yield limited or inconsistent improvements. Since each additional step increases training costs, a small number of bootstrap steps offers a practical balance between generation quality and computational overhead.

C.5 Iterative Refinement Distillation

We study the effect of iterative refinement distillation (IRD; Section B.1). As shown in Fig. 10, IRD improves few-step generation across all evaluated setups. On OWT with both T5 and GPT-2 encoders,

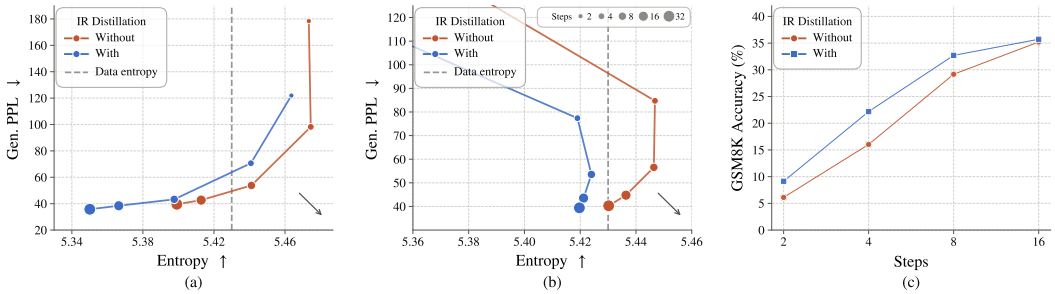

Figure 10: **Ablation on iterative refinement distillation.** Results with and without IRD on OWT with T5 (a) and GPT-2 (b) encoders, and on GSM8K (c). IRD benefits generation at low sampling budgets across all three setups, reducing generative perplexity on OWT and increasing accuracy on GSM8K.

it reduces generative perplexity at small sampling budgets. On GSM8K, it increases accuracy, with the largest gains at few sampling steps and diminishing improvements as the sampling budget grows.

C.6 Model-Size Scaling

To assess scaling to larger continuous DLMs, we train ELF-M on TinyGSM and post-train it with MMD. We omit IRD in this comparison. We use the same GPT-2 Small latent space as in the ELF-B experiments and evaluate both model sizes on GSM8K. As shown in Fig. 11, increasing model size improves accuracy both before and after MMD post-training across all evaluated sampling budgets. ELF-M-MMD also outperforms the original ELF-M at every budget, indicating that the benefits of our method persist when scaling from ELF-B to ELF-M. The smaller ELF-B-MMD achieves higher accuracy than ELF-M at 4–16 steps and similar accuracy at 32 steps.

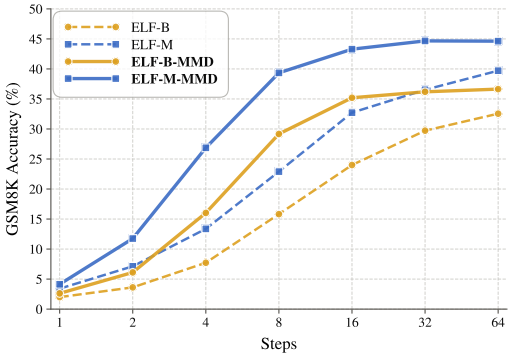

Figure 11: **Model-size scaling on GSM8K.** ELF-B and ELF-M before (dashed) and after (solid) MMD post-training.

D Additional DMax Details

We provide additional results for DMax-Math-MMD and DMax-Coder-MMD, examining the accuracy–TPF trade-off across decoding thresholds on math and code benchmarks. The operating points are selected from these benchmark results using the criterion described in Section 4.3.

For the math benchmarks, the selected thresholds are 0.75 for GSM8K, 0.9 for MATH500, 0.875 for Minerva-Algebra, and 0.65 for ASDIV. The published DMax-Math operating points use threshold 0.5. For the code benchmarks, we select thresholds 0.9 for HumanEval-Instruct and 0.875 for MBPP-Instruct, compared with 0.65 for the published DMax-Coder operating points. The uncertainties are sample standard deviations across the five recorded seeds.

Accuracy–parallelism trade-off. Figs. 13 and 14 show how accuracy changes with the decoding threshold for the models in Tab. 1. The baseline curves use our evaluations of the released DMax checkpoints, with one evaluation per threshold. The main table retains the baseline values reported in the DMax paper since we could accurately reproduce the original DMax results using the official implementation. Stars mark the selected MMD operating points.

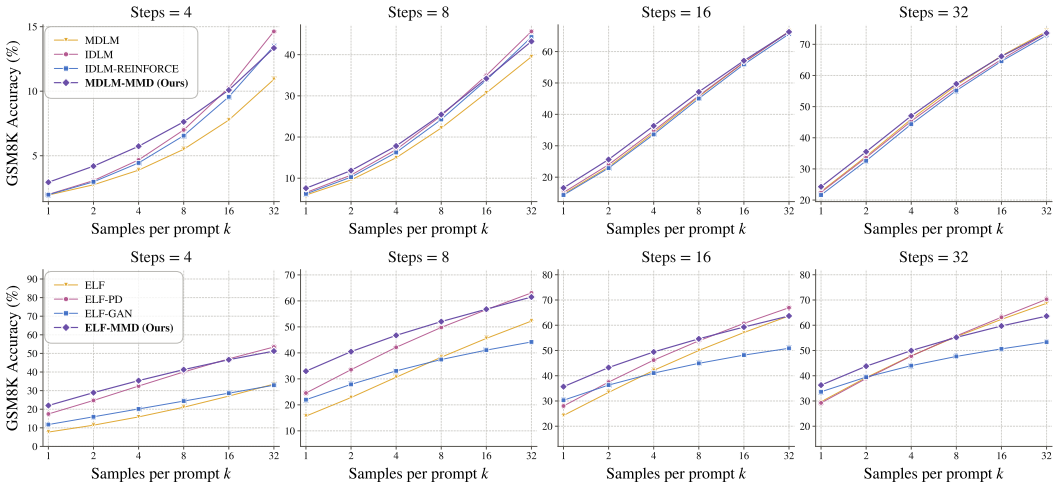

Figure 12: **Pass@k on GSM8K.** We report pass@k accuracy for different numbers of samples per prompt and sampling budgets of 4-32 steps for both discrete and continuous setups.

E Additional Continuous DLMs Experiments

E.1 ELF-GAN Training

To evaluate our post-training objective, we compare it with adversarial post-training. We initialize ELF-GAN from pretrained ELF-B checkpoints and post-train it for unconditional generation on OWT and conditional generation on TinyGSM. The discriminator is a four-layer MLP applied to token features extracted from real and generated latents by a frozen ELF. We use a logistic discriminator loss and a non-saturating generator loss, both computed per token and averaged over all non-prompt positions in the batch. We perform five discriminator updates per generator update and retain decoder fine-tuning as described in App. B.2. The generator uses the same optimizer and learning rate as ELF-MMD.

We also evaluate a sequence-level variant that mean-pools the extracted features over non-prompt positions before applying the discriminator MLP and computing the adversarial losses. This variant performs similarly to the token-level objective. Results are reported in Tabs. 3 and 4.

E.2 ELF*-MMD Post-Training

We apply our MMD objective to ELF* following the same training procedure as ELF-MMD. Features are extracted using a frozen copy of ELF*. On GSM8K, ELF*-MMD achieves higher accuracy than ELF* at every evaluated sampling budget (Tab. 4).

E.3 FMLM+-MMD Post-Training

We also apply our objective to FMLM+, which extends flow-map language modeling with posterior refinement. Features are extracted using a frozen copy of FMLM+. Training runs for $1{,}000$ steps with a learning rate of 5×10^{-6}, using otherwise the same settings as ELF-MMD. The resulting FMLM+-MMD achieves higher accuracy than FMLM+ on GSM8K at every evaluated sampling budget (Tab. 4).

F Hyperparameters

Tabs. 5, 6, and 7 summarize the hyperparameters for MDLM-MMD, ELF-MMD, and DMax-MMD, respectively, covering MMD settings, feature extraction, and optimization. We provide these details to document our experimental setup and help others reproduce our results.

Algorithm 1 ELF-MMD Training.

```
# net(z, t, sc, mode): our generator
# extractor(z, t, sc, mode): frozen ELF model
# s: a batch of B >= 2 reference sequences
# n: maximum number of generator passes

x = encode(s)
z = randn_like(x)
x_sc = zeros_like(x)

# bootstrapping: uniform in {0, ..., n-1}
n_sc = randrange(n)
for _ in range(n_sc):
    x_sc = stopgrad(net(
        z, t=0, sc=x_sc, mode="denoise"
    ))

# final differentiable pass
x_pred = net(z, t=0, sc=x_sc, mode="denoise")

# extract real features
real_feats = stopgrad(extractor(
    x, t=1, sc=x, mode="denoise"
))

# extract generated features
fake_feats = extractor(x_pred, t=1,
     sc=x_pred, mode="denoise"
)
loss = mmd_loss(fake_feats, real_feats)
```

Algorithm 2 ELF-MMD Sampling.

```
# shape: shape of embedded sequences
# n_steps: number of refinement steps

z = randn(shape)
x_pred = zeros_like(z)

for _ in range(n_steps):
    if resample_z: z = randn(shape)
    x_pred = net(z, t=0, sc=x_pred, mode="denoise")

# decoding
h = net(x_pred, t=1, sc=0, mode="decode")
token_logits = unembed(h)
tokens = argmax(token_logits)
```

Algorithm 3 Iterative Refinement Distillation.

```
# teacher: frozen ELF-MMD generator
# n_steps: teacher rollout length

z = randn(shape)
x_tgt = zeros_like(z)

for _ in range(n_steps):
    x_tgt = stopgrad(teacher(
        z, t=0, sc=x_tgt, mode="denoise"))

x_pred = net(z, t=0, sc=0, mode="denoise")
loss = mse_loss(x_pred, x_tgt)
```

Table 2: **MDLM results on OpenWebText.** We report generative perplexity (gPPL, ↓) and unigram entropy (Ent.) across sampling budgets. For each method and budget, we select the evaluated point whose entropy is closest to the reference-data entropy ($H = 5.43$). The results are averaged by 5 seeds.

Steps	8		16		32	
Method	gPPL	Ent.	gPPL	Ent.	gPPL	Ent.
MDLM	282.47 ±2.52	5.45 ±0.00	127.31 ±1.87	5.44 ±0.02	73.63 ±1.50	5.42 ±0.01
DiDi-Instruct	149.01 ±5.00	5.44 ±0.03	104.14 ±5.99	5.52 ±0.01	57.42 ±2.54	5.41 ±0.02
IDLM	56.20 ±0.68	5.44 ±0.01	33.77 ±0.58	5.44 ±0.01	29.30 ±0.58	5.44 ±0.02
IDLM-REINFORCE	52.93 ±0.36	5.41 ±0.01	36.00 ±0.55	5.43 ±0.01	34.30 ±0.81	5.47 ±0.01
MDLM-MMD	**43.50** ±0.86	5.43 ±0.01	**26.84** ±0.48	5.42 ±0.01	**21.46** ±0.18	5.39 ±0.01

Table 3: **Continuous DLMs on OpenWebText.** For each number of sampling steps we report generative perplexity (gPPL, ↓) and unigram entropy (Ent.). **Bold** highlights the lowest perplexity at comparable entropy. The results are averaged by 5 seeds.

Steps		2		4		8		16		32	
Encoder	Method	gPPL	Ent.	gPPL	Ent.	gPPL	Ent.	gPPL	Ent.	gPPL	Ent.
T5	ELF	34.02 ±6.95	4.38 ±0.30	116.38 ±6.61	5.48 ±0.05	91.77 ±2.70	5.38 ±0.01	57.77 ±1.42	5.36 ±0.01	45.13 ±1.06	5.30 ±0.01
	ELF-PD	120.58 ±1.91	5.49 ±0.00	88.29 ±1.32	5.45 ±0.00	63.73 ±0.45	5.39 ±0.00	59.82 ±0.60	5.36 ±0.00	58.45 ±0.50	5.35 ±0.00
	ELF-GAN	86.06 ±1.48	5.31 ±0.00	52.88 ±0.92	5.17 ±0.00	41.08 ±0.47	5.07 ±0.00	36.06 ±0.35	5.02 ±0.00	32.86 ±0.17	5.00 ±0.00
	ELF-MMD	178.38 ±1.00	5.47 ±0.00	98.19 ±0.55	5.47 ±0.00	53.82 ±0.32	5.44 ±0.00	42.68 ±0.17	5.41 ±0.00	39.54 ±0.06	5.40 ±0.00
	+ *IRD*	121.95 ±0.82	5.46 ±0.00	**70.65** ±0.69	5.44 ±0.00	**43.26** ±0.24	5.39 ±0.00	**38.41** ±0.13	5.36 ±0.00	**35.77** ±0.25	5.35 ±0.00
	ELF*	40.66 ±8.85	5.01 ±0.21	89.20 ±9.55	5.45 ±0.03	**79.37** ±2.61	5.44 ±0.03	60.17 ±4.76	5.32 ±0.04	44.31 ±1.91	5.25 ±0.01
	FMLM*	170.26 ±8.26	5.49 ±0.01	136.92 ±7.04	5.51 ±0.01	116.84 ±2.92	5.48 ±0.00	90.66 ±4.99	5.37 ±0.01	100.94 ±1.79	5.34 ±0.01
	ELF*-MMD	128.31 ±1.29	5.31 ±0.00	97.99 ±0.79	5.42 ±0.00	84.39 ±0.35	5.43 ±0.00	79.97 ±0.29	5.41 ±0.00	79.72 ±0.85	5.40 ±0.00
GPT-2	ELF	17.86 ±0.72	2.34 ±0.03	164.19 ±13.09	4.20 ±0.07	320.55 ±6.41	5.16 ±0.03	132.07 ±5.08	5.29 ±0.08	57.54 ±1.66	5.33 ±0.04
	ELF-PD	108.17 ±3.07	5.34 ±0.00	73.60 ±1.86	5.37 ±0.00	59.14 ±1.14	5.35 ±0.00	52.91 ±0.75	5.34 ±0.00	49.65 ±0.60	5.33 ±0.00
	ELF-GAN	103.15 ±1.54	5.47 ±0.00	79.93 ±0.94	5.44 ±0.00	68.20 ±0.54	5.41 ±0.00	61.21 ±0.67	5.40 ±0.00	56.15 ±0.20	5.37 ±0.00
	ELF-MMD	126.72 ±0.83	5.39 ±0.00	84.73 ±0.40	5.45 ±0.00	56.56 ±0.37	5.45 ±0.00	44.76 ±0.22	5.44 ±0.00	40.29 ±0.20	5.43 ±0.00
	+ *IRD*	113.20 ±0.45	5.34 ±0.00	**77.36** ±0.42	5.42 ±0.00	**53.55** ±0.21	5.42 ±0.00	**43.54** ±0.16	5.42 ±0.00	**39.40** ±0.23	5.42 ±0.00
	ELF*	84.45 ±7.68	4.13 ±0.09	156.48 ±12.44	5.40 ±0.03	88.43 ±1.60	5.42 ±0.01	66.65 ±1.27	5.41 ±0.01	66.78 ±12.44	5.40 ±0.03
	FMLM*	95.25 ±0.80	5.45 ±0.00	75.82 ±0.48	5.41 ±0.00	57.32 ±0.36	5.37 ±0.00	59.93 ±0.20	5.40 ±0.00	55.17 ±0.38	5.38 ±0.00
	ELF*-MMD	80.20 ±1.15	5.38 ±0.00	49.27 ±0.93	5.36 ±0.00	39.64 ±0.77	5.34 ±0.00	35.24 ±0.48	5.33 ±0.00	32.88 ±0.21	5.33 ±0.00

Table 4: **Continuous DLMs on GSM8K.** For each number of sampling steps we report final-answer accuracy (%). The results are averaged by 5 seeds.

Method	1	2	4	8	16	32	64
ELF	$2.00_{\pm 0.18}$	$3.62_{\pm 0.28}$	$7.72_{\pm 0.33}$	$15.82_{\pm 0.93}$	$24.00_{\pm 0.75}$	$29.72_{\pm 0.35}$	$32.55_{\pm 0.87}$
ELF-PD	$\mathbf{4.25}_{\pm 0.61}$	$9.05_{\pm 0.31}$	$17.26_{\pm 0.23}$	$24.73_{\pm 0.39}$	$28.39_{\pm 1.58}$	$28.93_{\pm 1.22}$	$29.14_{\pm 0.63}$
ELF-GAN	$2.77_{\pm 0.23}$	$5.48_{\pm 0.50}$	$11.99_{\pm 0.72}$	$22.30_{\pm 0.52}$	$30.15_{\pm 0.58}$	$33.17_{\pm 0.61}$	$34.14_{\pm 0.37}$
ELF-MMD	$2.62_{\pm 0.29}$	$6.11_{\pm 0.60}$	$16.02_{\pm 0.52}$	$29.17_{\pm 0.74}$	$35.20_{\pm 0.19}$	$\mathbf{36.19}_{\pm 0.51}$	$\mathbf{36.64}_{\pm 0.90}$
+ *IRD*	$3.19_{\pm 0.30}$	$\mathbf{9.12}_{\pm 0.41}$	$\mathbf{22.18}_{\pm 0.58}$	$\mathbf{32.70}_{\pm 0.45}$	$\mathbf{35.72}_{\pm 0.67}$	$35.96_{\pm 0.97}$	$36.37_{\pm 0.36}$
ELF*	$2.84_{\pm 0.42}$	$6.46_{\pm 0.60}$	$14.72_{\pm 0.61}$	$25.58_{\pm 1.16}$	$30.98_{\pm 0.76}$	$32.94_{\pm 0.69}$	$33.84_{\pm 0.68}$
ELF*-MMD	$\mathbf{2.88}_{\pm 0.29}$	$\mathbf{7.02}_{\pm 0.39}$	$\mathbf{19.71}_{\pm 0.75}$	$\mathbf{30.41}_{\pm 1.09}$	$\mathbf{33.90}_{\pm 0.47}$	$\mathbf{34.04}_{\pm 1.03}$	$\mathbf{34.13}_{\pm 0.44}$
FMLM+	$0.26_{\pm 0.11}$	$0.71_{\pm 0.19}$	$5.88_{\pm 0.69}$	$16.04_{\pm 0.83}$	$25.63_{\pm 1.16}$	$30.86_{\pm 1.22}$	$32.37_{\pm 1.09}$
FMLM+-MMD	$\mathbf{0.38}_{\pm 0.15}$	$\mathbf{0.91}_{\pm 0.12}$	$\mathbf{9.10}_{\pm 0.42}$	$\mathbf{19.48}_{\pm 0.68}$	$\mathbf{27.75}_{\pm 0.98}$	$\mathbf{32.83}_{\pm 1.37}$	$\mathbf{33.36}_{\pm 1.16}$

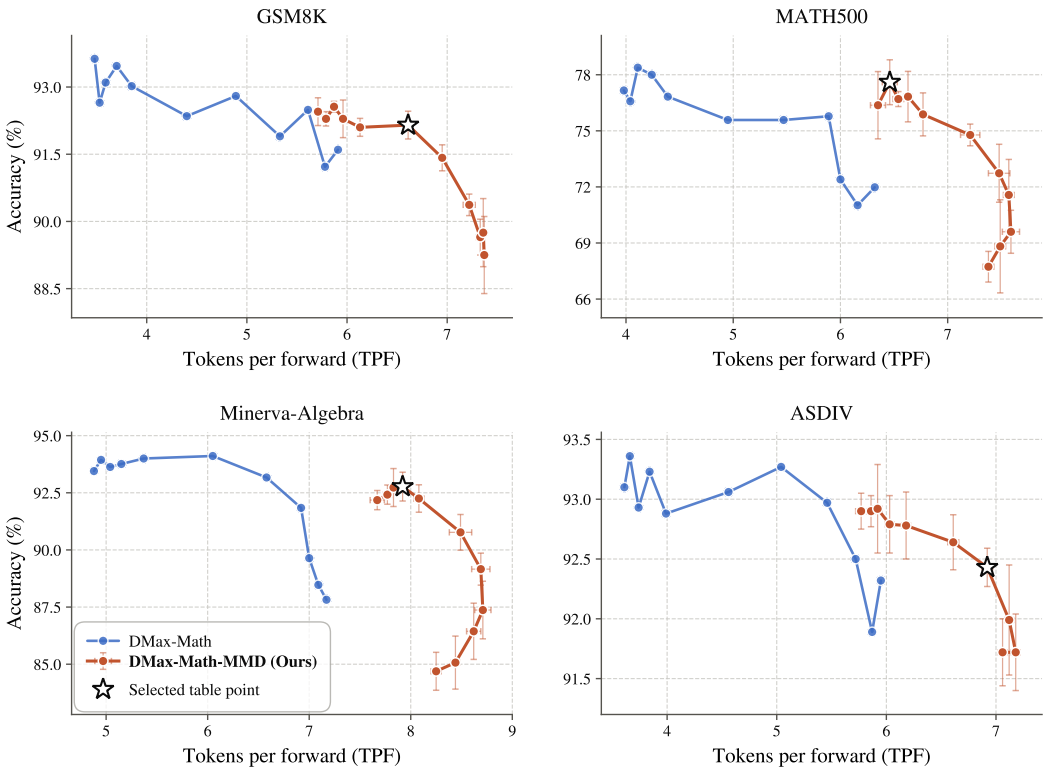

Figure 13: **Accuracy–TPF trade-offs on math benchmarks.** DMax-Math and DMax-Math-MMD are evaluated at eleven decoding thresholds. MMD points average five seeded runs, with error bars showing ± 1 sample standard deviation in both coordinates. Lines connect points in threshold order, and stars mark the operating points in Tab. 1.

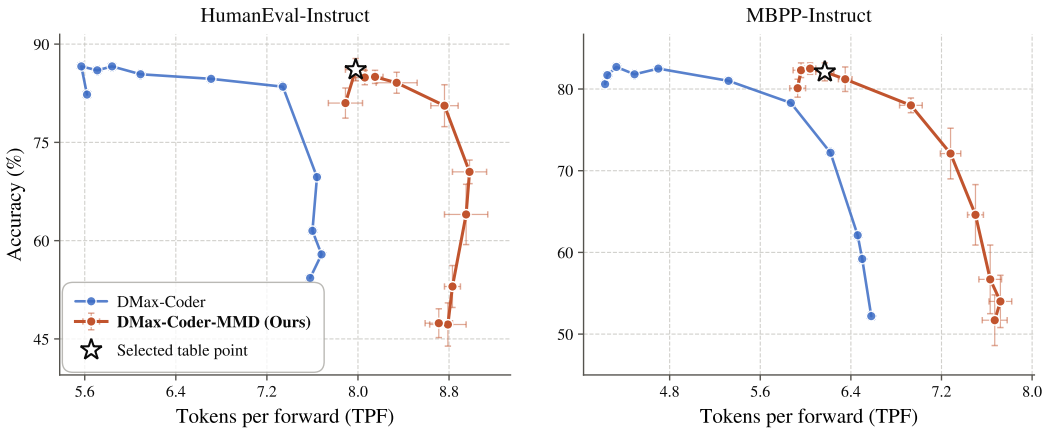

Figure 14: **Accuracy–TPF trade-offs on code benchmarks.** DMax-Coder and DMax-Coder-MMD are evaluated at eleven decoding thresholds. Accuracy denotes pass@1. MMD points average five seeds, with error bars showing ± 1 sample standard deviation in both coordinates. Lines connect points in threshold order, and stars mark the operating points in Tab. 1.

Table 5: **MDLM-MMD hyperparameters.** Settings for OWT and TinyGSM.

Hyperparameter	Unconditional (OWT)	Conditional (TinyGSM)
MMD training		
Kernel	RBF	RBF
Policy-gradient group size (G)	4	4
MMD batch size (B)	2	2
RBF bandwidth (σ)	160	90
Feature layer	4	7
Optimization		
Global batch size	512	512
Optimizer	AdamW	AdamW
Learning rate	1×10^{-4}	1×10^{-5}
Training steps	4,000	7,000

Table 6: **ELF-MMD hyperparameters.** Settings for OWT and TinyGSM.

	Unconditional (OWT)		Conditional (TinyGSM)
Hyperparameter	T5	GPT-2	GPT-2
MMD training			
Kernel	RBF	RBF	RBF
MMD batch size (B)	2	4	64
RBF bandwidth (σ)	50	176	3300
Feature layer	6	6	6
Conditioning and guidance			
Bootstrapping n	3	4	4
Self-conditioning CFG range	$[0.5, 5]$	$[0.5, 5]$	$[1.0, 1.0]$
Noise resampling (`resample_z`)	`false`	`false`	`true`
IRD			
Teacher rollout length K	8	8	32
Training steps	5,000	4,000	17,000
Optimization			
Global batch size	64	64	2
Optimizer	Muon	Muon	Muon
Learning rate	5×10^{-5}	5×10^{-5}	5×10^{-5}
Training steps	20,000	17,000	7,000

Table 7: **DMax-MMD hyperparameters.** Settings for math and code post-training.

Hyperparameter	DMax-Math-MMD	DMax-Coder-MMD
MMD training		
Kernel	RBF	RBF
Policy-gradient group size (G)	8	8
MMD batch size (B)	2	4
RBF bandwidth (σ)	10	16
Feature layer	10	10
Optimization		
Optimizer	AdamW	AdamW
Learning rate	2×10^{-6}	2×10^{-6}
Adam coefficients (β_1, β_2)	$(0.9, 0.999)$	$(0.9, 0.999)$
Training steps	400	400

G MDLM-MMD QUALITATIVE EXAMPLES

G.1 UNCONDITIONAL GENERATION ON OPENWEBTEXT

We provide unconditional samples generated by MDLM-MMD on OWT at each sampling step, together with their generative perplexity and entropy.

Sampling Steps: 2 gPPL: 708.40, Entropy: 5.42

… who because of her alleged offenses, systematically within systematically systematically abused unlawfully, inflicting physically different physical and psychological abuses against her, even activistsited ally herself in," comfortable of her misconduct among many in which she believed she as hoped enough could bring leverage to her malicious, ", she explained while exploring heat her "far far more," certain allies and said of her<|endoftext|>andails in placing her within a systematically pervasive government bureaucracy could harm her seeking system premiums game caused, would, she whether, misconduct demands, or violate her ” Medicaid reimburse herselfor enabled and and carefully evidence evidence if she had violated unlawfully transgress illegally or she could be guaranteeing being email precisely this, there, request to themselves …

Sampling Steps: 4 gPPL: 120.65, Entropy: 5.43

… her transfer, and that, despite her assurances from her efforts, she endured different physical and psychological trauma than her, even activists who had been in the consequences of government surveillance. Although she was repeatedly harassed and subjected to abusive abuse throughout her experiences, she repeatedly explained to investigators that despite had "reasonable suspicion," certain authorities lawyers said that she had not committed any legal wrongdoing within any law or government authoritythat she was seeking information about. "I would ask she whether any government demands for removal remove her” would “be enabled and and carefully evidence-free investigation,” as “she could have herself being released precisely … were there was request to remove her" But was her request be handled properly?<|endoftext|>Jennifer Lawyer, …

Sampling Steps: 8 gPPL: 45.04, Entropy: 5.42

… politically motivated information fraud," she said, "we would be in a very different position after working with the government." But activists concern that the government’s attempts to use this power could make it hard to find Kim Kim for alleged crimes. "If they’re weak, they’ll be able to arrest her in certain cases," said Lila Wong, an environmental activist, who is asking the government to stop foreigners from seeking information about her. But Kim asked she whether she would be able to pursue her case if her accusers were enabled and presented with evidence without a subpoena Wong suggested that Kim could easily come up with precisely which countries there would request for her information. InIn response she stressed that she had not been advising Kim in her foreign ministry for more than three years, and that she had acted …

Sampling Steps: 16 gPPL: 28.58, Entropy: 5.43

… lla Darev has demanded compensation and compensation worth £175m over allegations allegations against her parents at a Moscow university. But Ms Dare said she had sympathy with the plight of every girl who receives education training at Moscow university, and would urge them to take the necessary steps necessary to protect them from from future campus accidents. She said: "Moscow university is fully aware of the allegations against me and we will play a critical role in ensuring that students are fully protected. "We are confident of the compliance reforms that we have enabled and have no evidence of further abuse or exploitation. "We are working to ensure that we believe that there is no inappropriate conduct taking place." She said she had not been used for sexual abuse by other staff at students over the last three years and had …

Sampling Steps: 32 gPPL: 21.52, Entropy: 5.44

… At the very least, over the last six years, we have been pleased that Secretary Clinton has worked tirelessly to build an education and training infrastructure that will enable our Nation to meet its goal of providing high quality education for all children.”

Clinton’s campaign has not so far commented publicly about her use of the server. Last week, campaign spokesman Henry Johnson issued a few statements about the candidate’s use of the private email server:

“The Department does not allow the business interests of any individual person to play any role in the development, preparation, production, distribution, sale, purchase of, or other selling of controlled substances directly or indirectly by any employees or employees of the Department or its employees.” …

G.2 CONDITIONAL GENERATION ON TINYGSM

We provide examples for MDLM-MMD on TinyGSM.

Sampling Steps: 2 Accuracy: 0 Executed answer: Error

```
def simple_math_problem() -> int:
    '''
    There are three trees in Eddy's backyard.
    The shortest tree has a height of 6 feet, and the second tree has a
   height of 5 feet more than the shortest tree.
    The height of the tallest tree is twice the height of the two trees
   combined.
    How tall is the tallest tree?
    '''
    height_tree_1 = 6
    height_tree_2 = height_tree_1 + 5
    height_tree_3 = 2 * (__tree_ + height___2)
    = =_tree_3
    return result
```

Sampling Steps: 3 Accuracy: 0 Executed answer: Error

```
def simple_math_problem() -> int:
    '''
    There are three trees in Eddy's backyard.
    The shortest tree has a height of 6 feet, and the second tree has a
   height of 5 feet more than the shortest tree.
    The height of the tallest tree is twice the height of the two trees
   combined.
    How tall is the tallest tree?
    '''
    shortest_tree_height = 6
    second_tree_height = shortest_tree_height + 5
    tallest_tree_height = 2 * (short_tree_height + second_tree_height)
    result = tallesttreetree_height
    return result
```

Sampling Steps: 4 Accuracy: 1 Executed answer: 34

```
def simple_math_problem() -> int:
    '''
    There are three trees in Eddy's backyard.
    The shortest tree has a height of 6 feet, and the second tree has a
   height of 5 feet more than the shortest tree.
    The height of the tallest tree is twice the height of the two trees
   combined.
    How tall is the tallest tree?
    '''
    shortest_tree_height = 6
    second_tree_height = shortest_tree_height + 5
    tallest_tree_height = (shortest_tree_height + second_tree_height)*
  2
    result = tallest_tree_height
    return result
```

H ELF-MMD QUALITATIVE EXAMPLES

H.1 UNCONDITIONAL GENERATION ON OPENWEBTEXT

We provide unconditional samples generated by ELF-MMD on OWT at each sampling step, together with their generative perplexity and entropy.

H.1.1 T5 ENCODER

Sampling Steps: 2 gPPL: 128.72, Entropy: 5.49

... a regular factor in the procedures are surveys procedures, but the absence of a document which on the matter is the Commission, Prime Minister Dangg Ghai made it clear that the president and Nuember of Guinea have refused to change the matter and can not to do so. "It's a serious problem to my country's commitment to provide assistance to all people who are not able to handle anything involving gender, gender, racism and rape," he said. "We can conduct crimeness without fear of flight." When the DaoF-I published on the resolution earlier this month, human music organization TIEFES said the change paints are not intended to help people people because the country wants to show themselves to recognize their they are their ...

Sampling Steps: 4 gPPL: 70.51, Entropy: 5.50

... program, but with lack of a official release on the matter by theFA, Prime Minister Jenar Ghai made it clear that the president and the President of Uganda have refused to address the matter and agreeing to do so. "It's a serious breach of my country's commitment to provide justice to all who have not been unable to address issues such as race, religion, alcohol and rapicide," he said. "We can continue doing so without fear of fire." While the Nationalation of Marine Sciences reported on the matter earlier this month, human rights group UIIEES said the change of standards is not targeted to the handicaps, because the group wants to show their identity with their speakers and their languages. It also seeks to distribute safety information to ...

Sampling Steps: 8 gPPL: 43.71, Entropy: 5.43

... but with lack of a formal statement on the matter, Nudeter Prime Minister Inar Ghasi made it clear that the president and the President of Afghanistan have refused to address the issue and pledged to do so. "It's a fundamental part of my country's mission to give justice to anyone who may not be able to accept issues such as race, income, alcohol and facy," he said. "We can continue doing so without fear of fire." While the National Academy for Marine Science reported on the matter earlier this month, human rights group UIAIUK says the change of rules is not limited to the homosexuals, as the group wants to demonstrate their confidence in their languages and their languages. It also seeks to distribute this information to people in their area who are severely homeless or deprived of medical care and those living in the Africa Park Golf Club, which ...

Sampling Steps: 16 gPPL: 38.22, Entropy: 5.41

... new backgrounds program, but with lack of a formal statement on the matter, Nudeter press secretary Inar Ghati made it clear that the president and the President of Afghanistan have refused to address the issue and pledged to do so. "It's a fundamental part of my country's history to give justice to those who may not be able to accept things such as race, alcohol, alcohol and piracy," he said. "We can continue doing so without fear of violence." While the National Academy for Marine Science reported on the matter earlier this month, human rights group UIERAUK says the backgrounds program is not limited to Africanns, as the group wants to demonstrate their faith in their religion and their cultures. It also seeks to relay this information to people in the area who are severely disabled or deprived of medical care and those participating in the AFC, which ...

Sampling Steps: 32 gPPL: 35.69, Entropy: 5.40

... the backgrounds program, but with lack of a formal statement on the matter, Nudeter press secretary Inar Ghati made it clear that the president and the President of Afghanistan have refused to address the issue and pledged to do so. "It's a fundamental part of our country's history to give justice to those who may not be able to accept things such as race, drug, alcohol and piracy," he said. "We can continue doing so without fear of violence." While the National Academy for Marine Science reported on the matter earlier this month, human rights group UIERAUK said the backgrounds program is not available to Alaskans, because the group wants to demonstrate their faith in their religion and their cultures. It also seeks to extend its coverage to people in the area who are severely disabled or deprived of medical care and those participating in the AlaskaF Football Game, which can be played on a daily basis at 9 a.m. At the time, Gagibo said the backgrounds program ...

H.1.2 GPT-2 ENCODER

Sampling Steps: 2 gPPL: 91.65, Entropy: 5.37

… The Bitcoin of Pledge will truly support our money together. The capitalists and Bitcoiners distribute this precious currency to legitimate middle-class customers, and as some of us, we can find, try and move it into our supply.carts, rent, shop, rent them, and the same products are not worthless. [FORDREP] THE Purpose of Crypto Party
The Party is not invention and belongs to totalitarian rulers who controlled their society. From existing with medieval wars, the Noms, and the outbreak of the Naziism, all of them are dead. When the fascists had taken over this realm of brutalism and in the ensuing a period of warfare, the Fascist Party was again resurrected – the Green Party, the Republicanists of Fascists, the NX Party, and the Green Society. 4. Two Multi Kinders (of Silver, sized for $2250 (US$120 USD) and a NCoin-Gold-Coin Gold Shield Case for $1350. 4. FRatch of the Piece of "(!YXER" PIX. To provide the more composition and …

Sampling Steps: 4 gPPL: 58.36, Entropy: 5.43

… 3. The Price of Gold will dramatically improve our financial system. Bankaires and venders deliver this valuable commodity to our middle-class customers, and as many of us, we can buy, store and transfer it to our home. Fund stocks, buy, sell, and silver together and ensure great values are no hardship. Reward=JDY!
The Purpose of STEERINGER is nothing true, than the totalitarianists that controlled our society. From the first three years of the Namine, to the end of the Great War, all of them are dead. When the communists had taken over this world by ruthless force, and turned it four years of warfare, the NFA Group became several identities – the Green Party, the Nance of Thoms, the NAR Party, and the Green Party. 4. Two Rock Party Cards come in size, starting at $2595 (US $240 USD) and a NEM-In-Case / Shield Case for $2895. 4) A sleeve of the item called "SLEERYER" Sleeve. To provide further more functionality …

Sampling Steps: 8 gPPL: 41.71, Entropy: 5.40

… Private investors and large investors add this invaluable metal to our middle-class purchases, and as many of us, we should purchase, store and incorporate it into our home. Use it, buy, sell, refribute, and ensure your purchases go without hassle, quality in your time. Feder is nothing more more than the totalitarian forces that control our society.
From the first six years of the NED, to the beginning of the Great War, all of them are dead. When the Nazis had taken over this world by ruthless force and after more than two decades of war, the NY Coalition came in groups: the Buyer, the Nition of Pools, the NEM Group, and the Pioneer. 1) Two Solder Packs come in series, starting at $20 USD (about $70 USD) and a NEM-PlusxSize/Cube Pack for $35 USD. 2) Each box will come with a "SOLDER ROCKER" Sleeve. To provide even …

Sampling Steps: 16 gPPL: 37.06, Entropy: 5.37

… 3) The uptake of silver will greatly improve our food economy. Big businesses and main grains make a significant part in our mid-day diets, and as all of us, we should consume, store and incorporate silver in our food. Buy it, sell, sell, redistribute, and redistribute it all without any extra charge in our diet.
Founder is nothing much more than the Satanic forces that control our society. From the first six years of the Noney War to the end of the World War, all of them are over. After the Russians had taken over the world in 1844, after more than a decade of war, the Nurity Coalition came to form: the Solder, the Nition of Pounds, the Nurity Group, and the Solder. 4) Two Solder Packs available in package, starting at $20 Silver (for $60 USD) and a NEM Sleeve (the Solder Bundle for $60 USD). 4) Each box will come with a "SOLDER WATCHER" SIZE. To allow …

Sampling Steps: 32 gPPL: 36.97, Entropy: 5.34

… Silver is not a monohydritic metal. It is made of elements like copper and copper. It is an essential commodity and will have an impact in health for those who work in the energy industry. 3) The availability of silver will significantly improve our food economy. Fibers and silver are playing a vital role in our mid-day diet, and as many of us, we regularly consume, store and incorporate them into our food. Solders, blend, bottle, resele, and unitizers come without an extra ingredient in our food.
Cryer is nothing much more than the Satanic forces that controlled our society. From the first six years of the Noney War to the end of the Great War, all of them are dead. After the Russians had taken over the world in 1844, after more than a century of warfare, the Nurity Coalition came to form: the Solder, the Nition of Pints, the Nurity Society, and the Solder. 4) Two Solder Packs available in package, starting at $60 (total of $69 USD) and a NEM Sleeve (the Solder Bundle costs $48 USD). 4) Each unit will come with a "SOLDER CLICKER" Sleeve. To allow easier product creation and …

H.2 CONDITIONAL GENERATION ON TINYGSM

We provide examples for ELF-MMD on TinyGSM.

Sampling Steps: 1 Accuracy: 0 Executed answer: Error

```
def simple_math_problem() -> int:
    '''
    Lloyd earns $10 an hour on Math tutoring.
    And he tutored 5 hours for the first week and 8 hours for the
  second week.
    How much did he earn for the first two weeks?
    '''
    hourly_rate = 10
    hours_first_week = 5
    hours_second_week = 8
   total =40 and hours_first_week * hours_week
      total4033233
3
40

 result and  and result
```

Sampling Steps: 2 Accuracy: 0 Executed answer: Error

```
def simple_math_problem() -> int:
    '''
    Lloyd earns $10 an hour on Math tutoring.
    I He tutored 5 hours for the first week and 8 hours for the second
  week.
    How much did he earn for the first two weeks?
    '''
    hourly_rate = 10
    hours_first_week = 5
    hours_second_week = 8
    total_hours = hours_first_week + hours_second_week
    total_earnings = hourly_rate * hourly_rate
    result = total_earnings
    return resultA
```

Sampling Steps: 4 Accuracy: 1 Executed answer: 130

```
def simple_math_problem() -> int:
    '''
    Lloyd earns $10 an hour on Math tutoring.
     He tutored 5 hours for the first week and 8 hours for the second
  week.
    How much did he earn for the first two weeks?
    '''
    hourly_rate = 10
    hours_first_week = 5
    hours_second_week = 8
    total_hours = hours_first_week + hours_second_week
    total_earnings = hourly_rate * total_hours
    result = total_earnings
    return result
```