ONLINE CLASS POSTERIORS FOR INTERPRETABLE AND CONTROLLABLE DISCRETE DIFFUSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Discrete diffusion models remain far less understood than their continuous counterparts, despite recent breakthroughs across modalities, most notably in text. Leveraging tools from continuous diffusion, we formulate a theoretical framework to track the class-conditional posterior in an online manner by combining conditional and unconditional model predictions. In masked language modeling, this posterior offers highly interpretable dynamics: tokens aligned with a class boost the posterior, while contradictory ones suppress it. We also show that the posterior can be used for logical guidance operations (AND, OR, NOT) in discrete diffusion. These operations enable precise control over generation, such as avoiding specific classes. In addition, we discuss a mismatch in the formulation of classifier-free guidance between discrete and continuous settings: discrete models mix the data distributions, while continuous models mix the marginals. We validate our methods and discuss their potential applications on synthetic data, with all quantities analytically tractable, as well as on MNIST, text, and antibody sequences.

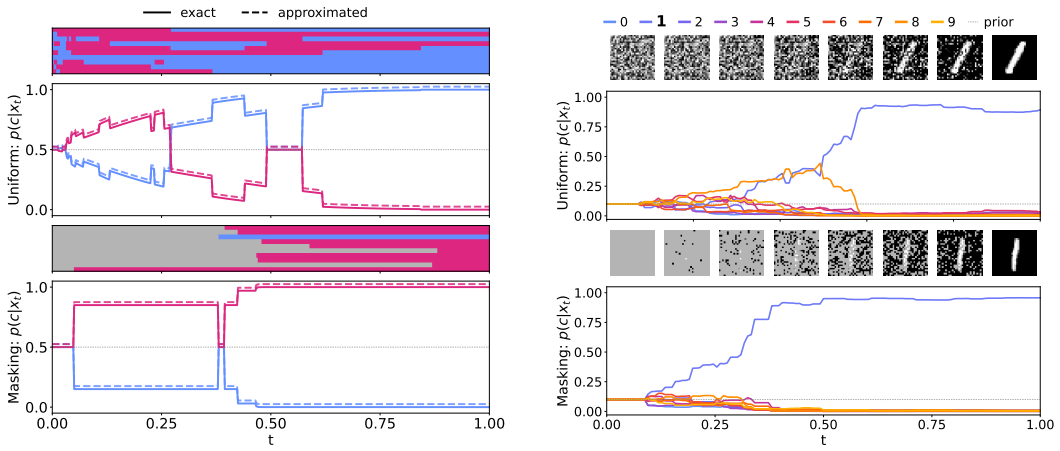
1 INTRODUCTION

Diffusion models are the default generative paradigm for continuous data, from images to protein structures (Sohl-Dickstein et al., 2015; Ho et al., 2020; Podell et al., 2024; Abramson et al., 2024). They learn the so-called score function $\nabla \log p_t(x_t)$, which gradually transports noise towards data. Discrete Diffusion Models (DDMs) transfer this idea by operating directly on the discrete state space (Austin et al., 2021; Shi et al., 2024; Campbell et al., 2022), as an alternative to embedding tokens into a continuous latent space and reusing continuous machinery (Dieleman et al., 2022). Like their continuous counterparts, DDMs progressively corrupt data, either by masking tokens or by replacing them with tokens drawn uniformly from the vocabulary. In the discrete setting, the score function is replaced by a rate function $R_t(x, y)$, the instantaneous probability of token y transitioning to token x. Integrating this rate over diffusion time defines a continuous time Markov chain. This process, like in the continuous case, carries the state from noise to data (Campbell et al., 2022; Gat et al., 2024).

Emerging evidence in language and DNA modeling suggests that DDMs could rival autoregressive models, thanks to their bidirectional context and multi-token generation (Nie et al., 2025; Stark et al., 2024; Sahoo et al., 2026). Yet, while the literature on continuous diffusion models has extensively explored noise schedules (Karras et al., 2022; Stancevic et al., 2025), training dynamics (Karras et al., 2024b), and inference-time guidance (Dhariwal & Nichol, 2021; Ho & Salimans, 2021), the algorithmic tools for discrete diffusion, and in particular for guiding it, remain far less developed. We address this gap by transferring to the discrete setting two guidance tools that have no discrete counterpart yet: the online tracking of the class posterior during generation, and the composition of conditional generative processes through logical operators.

Classifier-free guidance (CFG) (Dhariwal & Nichol, 2021; Ho & Salimans, 2021) applies a constant push towards a condition throughout the denoising process, regardless of whether the considered sample needs it. Recent work in continuous diffusion has shown that guidance can instead adapt to the state of the trajectory, only being used if a trajectory truly requires it (Koulischer et al., 2025b;a; Galashov et al., 2025). Additionally, guidance can be used for composing several conditional generative processes, for instance to generate samples satisfying either or both of two conditions. This requires weighting each process according to how likely the current state is to satisfy its correspond-

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.



(a) Synthetic data

(b) MNIST

Figure 1: Illustration of the online posterior evaluation. In (a), evaluated on an analytically tractable setting in which data points consist of two tokens represented by blue and red with each class having an 80% majority of one of the two tokens, trajectories are shown above the posteriors. In (b), the posterior is evaluated on MNIST digit generation. On top for uniform, below for masked diffusion. The increase in the posterior occurs simultaneously with the emergence of the digit.

ing condition (Skreta et al., 2025; Alesiani et al., 2026). All of these constructions rely on the same quantity, the class posterior $p_t(c \mid x_t)$, which estimates the probability that the noisy state x_t will eventually satisfy condition c. In this work, we focus on how conditional and unconditional model predictions, and their corresponding rate functions, can be combined to estimate this posterior and construct different conditional generative processes.

We show that the class posterior can be tracked during discrete diffusion directly from the conditional and unconditional predictions, for both uniform and masking noise. The estimation is exact when analytical quantities are available and exhibits interpretable token-level dynamics: because the state changes one token at a time, transitions that support a class increase its posterior, while transitions supporting a different class decrease it. This provides a direct interpretation of the posterior that has no continuous analogue. In masked language modeling, for example, it reveals which generated tokens move a sample towards or away from a class.

Building on this posterior estimation, we extend CFG to the logical AND, OR, and NOT operators for discrete rate functions, drawing on recent advances in the continuous domain (Alesiani et al., 2026; Skreta et al., 2025; Koulischer et al., 2025b). These operators combine conditional rate functions through posterior-dependent weighting: the OR gate arbitrates between the conditions according to their posterior probabilities, while the NOT gate detects the undesired condition and activates a repulsive term only when necessary. We verify these constructions analytically and validate them on discrete image and antibody sequence generation. Our derivation also reveals that standard discrete CFG (Schiff et al., 2025; Nisonoff et al., 2025) differs from its continuous counterpart in a way that has gone unnoticed. Continuous CFG mixes marginals multiplicatively, which is realized as an additive mixture of predictions, whereas discrete CFG instead mixes data distributions multiplicatively. These operations are not equivalent because the noising process does not commute with multiplicative mixing (Bradley & Nakkiran, 2024).

Our contributions are threefold: (i) we cast the online class posterior to the discrete domain and demonstrate its interpretable dynamics in language generation; (ii) we derive posterior-weighted OR and NOT operators for discrete rate functions, verify these on an analytical example, and validate on discrete image generation and antibody sequence generation; and (iii) we identify and formalize the multiplicative/additive mismatch between continuous and discrete CFG.

2 Background and Related Work

2.1 Discrete Diffusion Models

We consider sequences $\mathbf{x}_t = (x_t^1, \ldots, x_t^L) \in \mathcal{V}^L$ of L discrete tokens, with $t \in [0, 1]$ running from noise ($t = 0$) to data ($t = 1$). Let $q_t(x_t \mid x_1)$ denote the forward noising kernel for a single token. We consider the mixture path

$$q_t(x_t \mid x_1) = \kappa(t)\delta_{x_1}(x_t) + \big(1 - \kappa(t)\big)\sigma(x_t), \qquad \kappa(0) = 0, \quad \kappa(1) = 1, \tag{1}$$

where σ is a base noise distribution. The corresponding sequence-level corruption process factorizes over tokens, with classical choices including masking, in which tokens converge to an absorbing mask token, and uniform noising, in which tokens are replaced by samples from a uniform distribution (Austin et al., 2021).

The forward process defines a continuous-time Markov chain (CTMC) with marginal distribution p_t. Let $R_t(x, y)$ denote the instantaneous rate of transitioning from state y to state x. With this convention, the marginal distribution evolves according to the master equation

$$\frac{\mathrm{d}p_t(x)}{\mathrm{d}t} = \sum_{y \in \mathcal{V}} R_t(x, y)p_t(y) \equiv R_t p_t \tag{2}$$

with standard notation for the rate matrix R_t. Equivalently, for a small time step $h > 0$, the transition kernel, which can be used to construct a sampler, is given by

$$P_{t,t+h}(x \mid y) = \delta_{xy} + hR_t(x, y) + o(h), \tag{3}$$

Typically, a neural network is trained to approximate the denoiser $p_\theta(x_1 \mid x_t)$ using a time-weighted cross-entropy loss (Shi et al., 2024). This posterior can subsequently be converted into an instantaneous rate function. Importantly, the rate function is not unique: multiple generators can induce the same prescribed marginal path. One such construction is Discrete Flow Matching (Shaul et al., 2025), which, for the marginal path above, defines;

$$R_{\theta,t}(x, y) = \frac{\dot{\kappa}(t)}{1 - \kappa(t)} \left[p_\theta(x_1 = x \mid x_t = y) - \delta_{xy}\right]. \tag{4}$$

A different valid construction is given by the Markovian ancestral sampler (Austin et al., 2021).

In the language domain, Discrete Diffusion Models, or Diffusion Language Models (DLMs), already come with a range of decoding strategies for improving generation. LLaDA (Nie et al., 2025), a commonly used DLM, uses a fixed decoding schedule and confidence unmasking. The schedule fixes how many tokens are unmasked at each diffusion step, while confidence unmasking determines which ones: rather than choosing at random, it unmasks the tokens with the highest confidence.

2.2 Diffusion Guidance

Guidance in continuous diffusion models: Sampling on a class label or prompt c alone often does not yield satisfactory samples. In continuous diffusion models, guidance corrects this by tilting the noisy marginal with an exponentiated classifier term (Dhariwal & Nichol, 2021),

$$\tilde{p}_t(x_t|c) \propto p_{t,\theta}(x_t)\, p_{t,\theta}(c|x_t)^\lambda, \tag{5}$$

where $\lambda > 1$ controls the strength of the conditioning signal. Rewriting $p_{t,\theta}(c|x_t)$ via Bayes' rule replaces the classifier with a ratio of a conditional and an unconditional model, $\tilde{p}_t(x_t|c) \propto p_{t,\theta}(x_t)^{1-\lambda}\, p_{t,\theta}(x_t|c)^\lambda$, so that both terms can be obtained from a single jointly-trained network (Ho & Salimans, 2021). Taking the gradient of the log of the tilted marginal gives the familiar rules of Classifier Guidance and Classifier-Free Guidance, respectively:

$$\nabla \log \tilde{p}_t(x_t|c) = \nabla \log p_t(x_t) + \lambda \nabla \log p_t(c|x_t) \tag{6}$$

$$= \nabla \log p_t(x_t) + \lambda\big(\nabla \log p_t(x_t|c) - \nabla \log p_t(x_t)\big), \tag{7}$$

an *additive* combination of the conditional and unconditional score predictions.

Guidance in discrete diffusion models: The discrete setting admits a similar decomposition, although not equivalent. Guidance was introduced for discrete diffusion with the same goal of sharpening an under-confident conditional signal (Nisonoff et al., 2025; Schiff et al., 2025), realized by combining the logits of the predictions instead of the predictions themselves:

$$\log \tilde{p}(x_1|x_t, c) = \log p(x_1|x_t) + \lambda\big(\log p(x_1|x_t, c) - \log p(x_1|x_t)\big), \tag{8}$$

a *multiplicative* mixture of the predicted token distributions, in contrast to the additive mixture of scores used above.

Recent progress on guidance: Both formulations have since been refined. In the continuous case it was proposed to schedule the guidance scale λ over time rather than hold it fixed, aiming to target the noise levels that benefit most from it (Kynkäänniemi et al., 2024; Koulischer et al., 2025a; Galashov et al., 2025). Recent work has also shown that deliberately weakening the unconditional model pushes samples away from low-quality regions of the data space as well as from unconditionality (Karras et al., 2024a). Analogous ideas, including time-dependent schedules, have since begun to be carried over to the discrete formulation as well (Rojas et al., 2026). More recently, guidance has been reframed as one instance of a broader family of operators for combining diffusion processes, including AND (both conditions), OR (either condition) and NOT (avoid a condition) (Skreta et al., 2025; Koulischer et al., 2025b; Alesiani et al., 2026). The bottleneck of the latter two is the requirement of a class-conditional posterior $p_t(c|x_t)$ needed to weight this combination, which can be trained separately (Alesiani et al., 2026) or approximated from existing conditional/unconditional models (Koulischer et al., 2025b; Skreta et al., 2025). Discrete diffusion, however, has lacked both ingredients: the logical operators and a posterior to weight them. In this work, we close this gap, formulating the full family of operators for the discrete setting and showing that the posterior can be tracked during inference with high interpretability.

3 Posterior Estimation and Combining Multiple Processes

This section describes our key methodological contributions for discrete diffusion models. Section 3.1 shows how the class-conditional posterior can be estimated during inference by combining conditional and unconditional model predictions. Building on this, Section 3.2 introduces the three logical operators AND, OR and NOT, beginning with the notation they require and an important note on the (non-)commutativity of mixing and noising operations, which, for the AND operator, reveals that discrete and continuous CFG are not equivalent (Bradley & Nakkiran, 2024).

3.1 Online Posterior Estimation

The conditional posterior $p_t(c|x_t)$, which evaluates how likely a noisy state x_t ends up satisfying the condition c, is often mistakenly considered intractable during inference. In the continuous case, its tractability when sampling using a reversible Markov chain has been demonstrated (Koulischer et al., 2025b; Skreta et al., 2025), but to the best of our knowledge, this has not been explored for discrete diffusion models. We here present how $p(c \mid x_t)$ can be tracked iteratively through inference and what specifics arise in the discrete case as opposed to the continuous case. The idea is to track the posterior $p(c \mid x_{0:t})$ throughout inference using the conditional and unconditional models' predictions, with $x_{0:t}$ a shorthand notation for all states leading up to t:

$$\log p(c \mid x_{0:t}) = \log p(c \mid x_{0:t-1}) + \log p(x_t \mid x_{t-1}, c) - \log p(x_t \mid x_{t-1}). \tag{9}$$

In practice, we follow Koulischer et al. (2025b) and introduce two hyperparameters τ and δ:

$$\log p(c \mid x_{0:t}) = \log p(c \mid x_{0:t-1}) + \tau\big(\log p(x_t \mid x_{t-1}, c) - \log p(x_t \mid x_{t-1})\big) + \delta. \tag{10}$$

If a Markovian sampler is used, the current state is a sufficient statistic for the class: $p(c|x_t) = p(c \mid x_{0:t})$. In the masking case, this identity always holds since a token never changes once unmasked and x_t therefore retains every revealed token. In the uniform case, the Markov property only holds when sampling using an ancestral sampler specifically constructed to be Markovian. However, in practice we find that using the transition kernel predicted by a discrete flow matching sampler is more stable, which is why we use it in the results section.[1]

[1] For more detail, see Appendices B.3 and C.2.

A key advantage of the discrete case, compared to its continuous analogue, is that one can see which token flips caused the posterior to change, therefore allowing the user to understand what might have caused the model to opt for one class or another. This is particularly interesting in masked language modeling, where the posterior reveals which words commit the sample to a class, long before denoising completes.

3.2 LOGICAL OPERATORS: AND, OR, NOT

We now show how logical mixture operators $\Lambda \in \{\text{AND}, \text{OR}, \text{NOT}\}$ over conditional distributions $\{p(x|c_i)\}_{i=1}^N$, defined by their rate matrices R_t^i or predictors $p^i(x_1|x_t)$, combine into a joint rate R_t^Λ or predictor $p^\Lambda(x_1|x_t)$. Crucially, an operator can be applied at two levels: (i) before noising, thus mixing the data distributions (which we show is what discrete CFG does (Schiff et al., 2025)), or (ii) after, which amounts to mixing the marginals (as in continuous CFG (Ho & Salimans, 2021)). Because the noising convolution commutes with additive but not multiplicative mixing, the two levels (i)–(ii) coincide for the additive OR and NOT operators, but differ for the multiplicative AND. Hence, we provide separate derivations for AND, but jointly present them for OR and NOT (see Appendix B.1 for a detailed treatment).

AND gate. ***(data-level mixture)*** A data-level mixture targets the mixture distribution exactly. For AND, this is the multiplicative mixture $p^\Lambda(x_1|x_t) \propto \prod_{i=1}^N p^i(x_1|c_i)$, with as predictor:

$$p^\Lambda(x_1|x_t) = p(x_t|x_1)\frac{p_1^\Lambda(x_1)}{p_t^\Lambda(x_t)} \propto p(x_t|x_1)\prod_{i=1}^N p^i(x_1|c_i) \tag{11}$$

$$\propto p(x_t|x_1)^{1-N}\prod_{i=1}^N p^i(x_1|x_t, c) \tag{12}$$

Thus the predicted distributions should be multiplied with each other, which is commonly implemented as adding logits together:

$$\log p(x_1|x_t) \propto \sum_{i=1}^N \log p^i(x_1|x_t, c) \tag{13}$$

From this predictor a rate function can be constructed using eq. (4). This sum of logits strongly resembles how CFG is implemented in the discrete diffusion literature, i.e., through eq. (8).

AND gate. ***(marginal-level mixture)*** On the other hand, an approach reminiscent of the continuous diffusion literature can be applied: considering a multiplicative mixture of the marginals defined as $p_t^\Lambda(x_t) \propto \prod_{i=1}^N p_t(x_t|c_i)$.[2] The joint rate function can be directly derived through:

$$\frac{\mathrm{d}p_t^\Lambda(x_t)}{\mathrm{d}t} = \sum_i \frac{\mathrm{d}p_t(x_t|c_i)}{\mathrm{d}t}\prod_{j\neq i}^N p(x_t|c_j) = \Big(\sum_i R_t^i\Big)p_t^\Lambda(x_t) \tag{14}$$

Hence, the rate matrices of the joint process satisfy:

$$R_t^\Lambda = \sum_{i=1}^N R_t^i. \tag{15}$$

The shape of this combination is also intuitive: as long as one of the conditions is not satisfied, the corresponding rate will try to strongly modify the outcome. Only once x_t satisfies both conditions will both rate functions remain reasonably small. This is analogous to continuous CFG, which similarly combines two infinitesimal generators, with the score functions being replaced by the discrete rate functions.

[2]Note the change in subscript from $_1$ to $_t$.

OR gate. ***(both)*** For the additive operation behind the OR gate, $p_t^{\Lambda}(x_t) \equiv p_t(x_t) = \sum_{i=1}^{N} p(c_i) p_t(x_t|c_i)$, with mixture weights given by the priors $p(c_i)$, p_t^{Λ} is simply the unconditional marginal induced by the partition $c_i{}_{i=1}^{N}$. The joint rate function can again be computed directly:

$$\frac{\mathrm{d}p_t^{\Lambda}(x_t)}{\mathrm{d}t} = \sum_i p(c_i) \frac{\mathrm{d}p_t(x_t|c_i)}{\mathrm{d}t} = \left(\sum_i p(c_i) \frac{p_t(x_t|c_i)}{p^{\Lambda}(x_t)} R_t^i \right) p_t^{\Lambda}(x_t) \tag{16}$$

Recognizing the resulting Bayesian posterior gives:

$$R_t^{\Lambda} = \sum_{i=1}^{N} p(c_i \mid x_t) R_t^i. \tag{17}$$

Thus, the global rate is obtained by averaging the conditional rates according to the posterior probability of each component given the current noisy state.

This perspective closely mirrors the construction underlying Flow Matching (Lipman et al., 2023; Gat et al., 2024): rather than learning the global dynamics directly, one can define simple conditional dynamics and learn how to identify their contribution from the current state. The global dynamics are then obtained by posterior-weighted averaging of the conditional dynamics. In particular, taking each datapoint as a separate condition yields datapoint-specific rates that can be combined through $p(c_i|x_t)$ to recover the global rate. Our construction therefore extends this conditional-to-global principle from datapoint-level conditioning to arbitrary partitions of the data.

Beyond this theoretical connection, the OR gate is not strictly necessary for class-conditional generation, since sampling a condition according to its prior $c_i \sim p(c)$, and subsequently using the corresponding conditional model, yields the same desired distribution.

NOT gate. ***(both)*** Removing, or avoiding, a class from generation can be expressed as subtracting its conditional distribution from the unconditional one, $p_t^{\Lambda}(x_t) \propto p_t(x_t) - p(c)\, p_t(x_t|c)$. The joint rate is then given by:

$$\frac{\mathrm{d}p_t^{\Lambda}(x_t)}{\mathrm{d}t} = \frac{R_t\, p_t(x_t) - p(c)\, R_t^c\, p_t(x_t|c)}{1 - p(c)} = \left(R_t - \frac{p_t(c|x_t)}{1 - p_t(c|x_t)} \left(R_t^c - R_t \right) \right) p_t^{\Lambda}(x_t) \tag{18}$$

And therefore:

$$R_t^{\Lambda} = R_t - \frac{p(c \mid x_t)}{1 - p(c \mid x_t)} \big(R_t^c - R_t \big). \tag{19}$$

This equation resembles a classifier-free guidance equation, in which $\frac{p(c|x_t)}{1-p(c|x_t)}$ can be seen as a *dynamic* guidance scale that is small if the undesired class is unlikely to be generated and that becomes asymptotically large if that class becomes likely. Again, this is entirely analogous to what is found in the continuous case, with as single difference that the continuous generators, or score-functions, are replaced by their discrete analogues, the rate functions (Koulischer et al., 2025b).

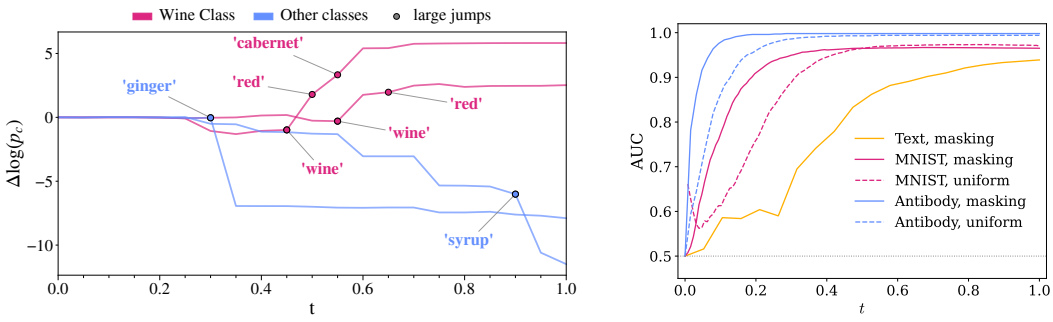
4 RESULTS

Section 4.1 first validates the posterior approximation on (i) synthetic data for which ground-truth quantities are available before being tested on MNIST images (LeCun & Cortes, 2010), (ii) text, and (iii) antibody sequence generation (Shuai et al., 2023). Next, in Section 4.2, we evaluate the NOT gate for image and antibody sequence generation.

4.1 ONLINE POSTERIOR ESTIMATION

Synthetic data. Our synthetic dataset consists of binary sequences of length 10, with red and blue tokens. Each sequence is assigned to one of two classes based on a majority token: one token (red or blue) is sampled with probability p_{maj}, and the other with $1 - p_{\text{maj}}$. Figure 1a shows that the inferred posterior matches the analytically tractable ground-truth quantity computed using Bayes' rule with the known marginal likelihoods. Importantly, the posterior inference remains exact across both masking and uniform noise processes,[3] despite their substantially different transition dynamics.

[3]Uniform noise requires the Markovian ancestral transition kernel. Details are provided in Appendix C.2.



(a) Illustrative posterior trajectories

(b) AUC during denoising

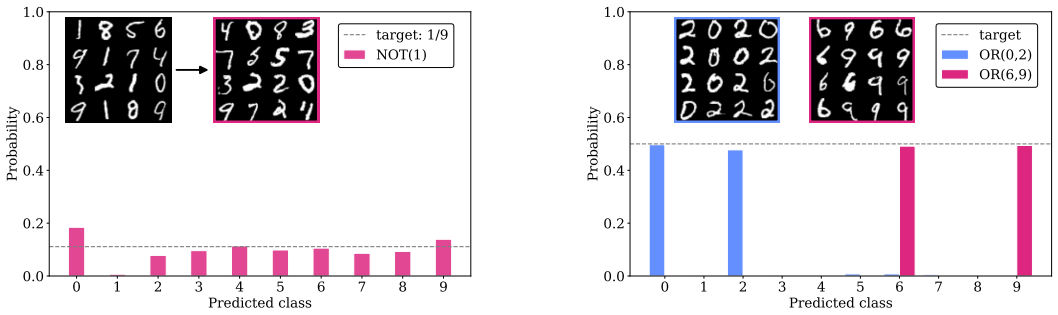
Figure 2: In (a), the progress of the posterior approximation is shown throughout generation of four sentences on "*beverage*" with as class "*wine*". Highlighted words show large jumps in the posterior, corresponding to those given in Table 1. In (b), one-vs-rest AUC is evaluated throughout generation, using a modality-specific external classifier to determine the final class.

Image generation. For the image modality, we train conditional models on MNIST using both masking and uniform noising schemes (training details and ablations in Appendix D.1). During sampling, the posterior over the 10 MNIST classes is estimated online by applying a softmax to the class-specific log-posterior quantities computed with eq. (10). To evaluate the posterior throughout denoising, trajectories are classified using a pretrained ViT applied to the final state. At each noise level t, we evaluate the posterior scores in a one-vs-rest classification setting and report the AUC averaged over the ten classes, measuring how well the online posterior discerns the class eventually expressed by the generated sample. As shown in Fig. 2b, the AUC increases throughout denoising, indicating that the posterior becomes progressively more informative as the sample develops class-specific structure. Representative trajectories and their corresponding posterior estimates are shown in Fig. 1b, with additional examples in Appendix D.3. In both noise processes, the posterior increases as class-specific structure emerges; compared to masking, the increase tends to occur later under uniform noising, caused by the ability of uniform diffusion to revise previously generated tokens and therefore defer commitment to a particular class.

Text generation. Here, evaluation is based on the Cognac (Chen et al., 2022) benchmark, with as task generating a Wikipedia-style sentence around a certain topic. The two models are obtained by prompting LLaDA-8B-Instruct (Nie et al., 2025) with two different prompts. The unconditional model is asked to generate about a category (e.g., "beverage"), while the conditional model is prompted to generate about a sub-category (e.g., "wine") which we refer to as the classes. Full details of the experimental setup are in Appendix E.1. Importantly, the online posterior estimation allows analyzing which unmasking events result in the generation of a sentence of a particular sub-category. Table 1 quantifies this, by showing the tokens that have the strongest impact on the

Topic	Sub-topic (class c)		1st Token	1st $\Delta \log p_c$	2nd Token	2nd $\Delta \log p_c$	3rd Token	3rd $\Delta \log p_c$
beverage	wine	+	wine	+2.3	cabernet	+1.3	red	+1.1
		−	syrup	-5.5	ginger	-3.9	espresso	-3.2
sweet	candy	+	candy	+2.8	lollipop	+0.8	are	+0.5
		−	marzipan	-2.1	cookies	-0.7	sweet	-0.6
dairy product	cheese	+	cheddar	+2.4	Swiss	+1.4	cheese	+0.5
		−	yogurt	-6.8	cream	-0.5	product	-0.5

Table 1: Tokens pushing most towards (+) and away from (−) the class, estimated by our posterior estimation. (Sub)words in gray provide context for the token shown in black. Each gray context corresponds to the actual context for that token in all samples containing that token.



(a) Samples under NOT gate

(b) Samples under OR gate

Figure 3: Illustration of the NOT and OR gates on MNIST under a uniform noise model. (a) The NOT gate, showing how unconditional samples containing the undesired number *"1"* are transformed into other, predominantly round numbers such as *"0"* and *"9"*. (b) The OR gate evaluated on two distinct pairs, with corresponding samples shown above.

posterior across 1000 samples for different tasks. Words with strong positive impact on the posterior are either a word that belongs to the class (e.g., 'cheddar' for class cheese), or words that strongly implicate a class word (e.g., 'Swiss' for class cheese). Conversely, words with strong negative impact on the posterior often exemplify subcategories of the topic that are unrelated to the constraint (e.g., 'yogurt'). In Figure 2a, four trajectories are shown, illustrating how the posterior moves strongly in the right direction after words are generated exemplifying the class or the complement of a class.

Antibody sequence generation. For antibody sequence generation, we train a 50M-parameter Transformer based on the architecture of Fournier et al. (2024) on the OAS dataset (Olsen et al., 2022; Shuai et al., 2023), restricted to human antibodies. Antibody sequences are partitioned by V-gene type as classified by IgBLAST (Ye et al., 2013). The V-gene largely determines the antibody framework, providing a strong conditioning signal. Our posterior estimation accurately identifies the V-gene from the partially generated sequence, achieving near-perfect AUC as shown in Figure 2b. Furthermore, the posterior becomes informative after only a small number of token flips, allowing the conditioning variable to be identified early in generation. Further details on this modality are provided in Appendix F.

4.2 Logical Operators

Of the three gates, the NOT gate is most interesting, combining two rate functions into an otherwise unavailable novel rate function. Note that in **text**, functionality of the NOT gate can be achieved adapting the prompt by stating which class to avoid, a baseline our method matches as shown in Appendix E.3. Next, we consider the more interesting cases of applying NOT for image and antibody

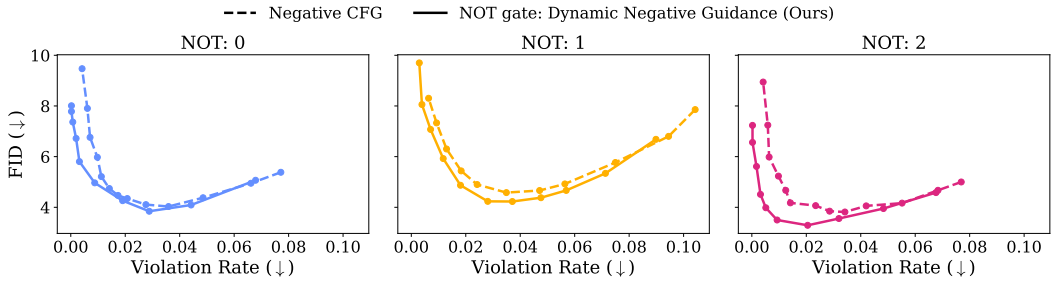


Figure 4: Pareto curves comparing the quality of the generated samples to the safety of the approach, respectively through the FID and the violation rate. Both negative CFG, as well as the theoretically justified dynamic NOT gate display a trade-off between quality and safety, with the dynamic approach outperforming the static one, albeit at the cost of an additional hyperparameter.

Method	Setting	Majority		Minority	
		Viol. ↓	FPD ↓	Viol. ↓	FPD ↓
Unconditional		0.278	0.174	0.032	0.007
Negative CFG	majority-calibrated	0.013	0.104	0.002	0.326
	minority-calibrated	0.186	0.084	0.017	0.020
NOT gate	both	0.010	0.074	0.016	0.010

Table 2: NOT gate performance when removing majority and minority V-gene families. Static guidance is calibrated to match the dynamic method's violation rate in the corresponding regime, then transferred across regimes. Gray entries show performance outside the calibration regime. The full Pareto front is given in Appendix F.5.

sequence generation. For validation of the theory, rather than intended for practical application, we also evaluate the OR gate in the image modality.

Image generation. The NOT gate is evaluated using the Pareto front of safety and quality, measured respectively by the violation rate against a ViT classifier, and FID on 4096 samples (Heusel et al., 2017). In Figure 4, the dynamic approach is compared to static CFG-style negative logit guidance, which samples from $\tilde{p}(x) \propto p_1(x_1)p_1(c \mid x_1)^{-\lambda}$. A grid search for the optimal hyperparameter is performed for the two approaches, with more details in Appendix D.4. Both methods exhibit the expected trade-off between violation rate and image quality. The dynamic NOT gate consistently achieves a better trade-off than negative CFG, at the cost of the additional hyperparameter τ. Figure 3a shows the distribution resulting from excluding the class "1" and how it impacts samples generated from the same seed using the unconditional model. The OR gate is evaluated similarly by restricting the posterior to the two desired conditions, visualized in Figure 3b.

Antibody sequence generation. In this setting, we evaluate how the NOT gate removes a V-gene family while preserving quality, measured respectively using violation rate and Fréchet Protein Distance (FPD) to the desired population in ESM2 embedding space (Lin et al., 2023). The dynamic NOT gate is compared against unconditional generation and static negative CFG.

A key challenge in this setting is the strong imbalance between V-gene families: nearly 95% of sequences belong to just three of the seven families. Removing a majority class requires substantially stronger guidance than removing a minority class. The dynamic formulation accounts for this imbalance as the initial guidance strength scales with the class prior. In Table 2, we choose a setting such that both majority and minority classes are removed with only small FPD increases.[4] For comparison, negative CFG is calibrated separately to achieve a comparable violation rate for majority and minority classes. Majority-calibrated guidance is overly strong for minority classes, causing a substantial increase in FPD, while minority-calibrated guidance is too weak for majority classes. The dynamic NOT gate does not suffer the same flaw thanks to its prior-weighted initialization.

CONCLUSION

We have shown that the class-conditional posterior in discrete diffusion can be tracked online, at no cost beyond the conditional and unconditional predictions already required for guidance. In diffusion language modeling this posterior is highly interpretable: individual tokens can be seen to commit a sample toward a class or away from it during generation. Building on the posterior, we derived the AND, OR, and NOT operators over discrete rate functions, providing a principled means of composing conditional processes. The NOT gate is of particular practical value, enabling generation that steers away from a class in modalities such as images and antibody sequences, where prompting a model to avoid a concept is not possible. As the construction is agnostic to modality, noising process, data imbalances and schedules, it applies unchanged across the settings we consider. The derivation additionally revealed a previously unremarked discrepancy between classifier-free guidance in the continuous domain, an additive mixture of scores, and the discrete domain, a multiplicative mixture of predictions. Limitations are provided in Appendix A.

[4] See Appendix F.5 for the full Pareto curves and guidance sweeps.

AI DISCLOSURE

In this work, we used generative AI tools to implement methods used in this paper. Additionally, we used generative AI tools to plot results, create or edit software code, and edit the paper to improve readability. We have not used generative AI tools for other purposes. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J. Ballard, Joshua Bambrick, Sebastian W. Bodenstein, David A. Evans, Chia-Chun Hung, Michael O'Neill, David Reiman, Kathryn Tunyasuvunakool, Zachary Wu, Akvilė Žemgulytė, Eirini Arvaniti, Charles Beattie, Ottavia Bertolli, Alex Bridgland, Alexey Cherepanov, Miles Congreve, Alexander I. Cowen-Rivers, Andrew Cowie, Michael Figurnov, Fabian B. Fuchs, Hannah Gladman, Rishub Jain, Yousuf A. Khan, Caroline M. R. Low, Kuba Perlin, Anna Potapenko, Pascal Savy, Sukhdeep Singh, Adrian Stecula, Ashok Thillaisundaram, Catherine Tong, Sergei Yakneen, Ellen D. Zhong, ichal Zielinski, Augustin Žídek, Victor Bapst, Pushmeet Kohli, Max Jaderberg, Demis assabis, and John M. Jumper. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630(8016):493–500, 06 2024. URL https://doi.org/10.1038/s41586-024-07487-w.

Francesco Alesiani, Jonathan H Warrell, Tanja Bien, Henrik Christiansen, Matheus Ferraz, and Mathias Niepert. Logical guidance for the exact composition of diffusion models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=OAM1jJsMGp.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, 2021.

Arwen Bradley and Preetum Nakkiran. Classifier-free guidance is a predictor-corrector, 2024. URL https://arxiv.org/abs/2408.09000.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Tom Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 2022.

Howard Chen, Huihan Li, Danqi Chen, and Karthik Narasimhan. Controllable Text Generation with Language Constraints, 2022.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, 2021.

Sander Dieleman, Laurent Sartran, Arnaud Dauphin, Alessia Mancuso, Matthew Rebain, Mikołaj Bińkowski, Natascha Kursawe, Jack W Rae, and Oriol Casagrande. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Quentin Fournier, Robert M. Vernon, Almer van der Sloot, Benjamin Schulz, Sarath Chandar, and Christopher James Langmead. Protein language models: Is scaling necessary? *bioRxiv*, 2024. URL https://www.biorxiv.org/content/early/2024/09/23/2024.09.23.614603.

Alexandre Galashov, Ashwini Pokle, Arnaud Doucet, Arthur Gretton, Mauricio Delbracio, and Valentin De Bortoli. Learn to guide your diffusion model, 2025. URL https://arxiv.org/abs/2510.00815.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, 2024.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 2022.

Tero Karras, Miika Aittala, Tuomas Kynkäänniemi, Jaakko Lehtinen, Timo Aila, and Samuli Laine. Guiding a diffusion model with a bad version of itself. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=bg6fVPVs3s.

Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *Proc. CVPR*, 2024b.

Felix Koulischer, Florian Handke, Johannes Deleu, Thomas Demeester, and Luca Ambrogioni. Feedback guidance of diffusion models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=8ySOcf7UpM.

Félix Koulischer, Johannes Deleu, Gabriel Raya, Thomas Demeester, and Luca Ambrogioni. Dynamic negative guidance of diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025b.

Tuomas Kynkäänniemi, Miika Aittala, Tero Karras, Samuli Laine, Timo Aila, and Jaakko Lehtinen. Applying guidance in a limited interval improves sample and distribution quality in diffusion models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=nAIhvNy15T.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 2023. URL https://www.science.org/doi/abs/10.1126/science.ade2574.

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.

George A. Miller. WordNet: A lexical database for English. *Commun. ACM*, 38:39–41, 1995. ISSN 0001-0782.

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 *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=KnqiC0znVF.

Hunter Nisonoff, Junhao Xiong, Stephan Allenspach, and Jennifer Listgarten. Unlocking guidance for discrete state-space diffusion and flow models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=XsgHl54yO7.

Tobias H Olsen, Fergus Boyles, and Charlotte M Deane. Observed antibody space: A diverse database of cleaned, annotated, and translated unpaired and paired antibody sequences. *Protein Science*, 01 2022.

William S. Peebles and Saining Xie. Scalable diffusion models with transformers. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2022. URL https://api.semanticscholar.org/CorpusID:254854389.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=di52zR8xgf.

Kevin Rojas, Ye He, Chieh-Hsin Lai, Yuhta Takida, Yuki Mitsufuji, and Molei Tao. Improving classifier-free guidance in masked diffusion: Low-dim theoretical insights with high-dim impact. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=mMK9pvQJxf.

Subham Sekhar Sahoo, Jean-Marie Lemercier, Zhihan Yang, Justin Deschenaux, Jingyu Liu, John Thickstun, and Ante Jukić. Scaling beyond masked diffusion language models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=xKJG1CNtQM.

Yair Schiff, Subham Sekhar Sahoo, Hao Phung, Guanghan Wang, Sam Boshar, Hugo Dalla-torre, Bernardo P de Almeida, Alexander M Rush, Thomas PIERROT, and Volodymyr Kuleshov. Simple guidance mechanisms for discrete diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=i5MrJ6g5G1.

Neta Shaul, Itai Gat, Marton Havasi, Daniel Severo, Anuroop Sriram, Peter Holderrieth, Brian Karrer, Yaron Lipman, and Ricky T. Q. Chen. Flow matching with general discrete paths: A kinetic-optimal perspective. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net.

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

Richard W Shuai, Jeffrey A Ruffolo, and Jeffrey J Gray. Iglm: Infilling language modeling for antibody sequence design. *Cell Systems*, 2023.

Marta Skreta, Lazar Atanackovic, Avishek Joey Bose, Alexander Tong, and Kirill Neklyudov. The superposition of diffusion models using the itô density estimator. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=2o58Mbqkd2.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015. URL https://proceedings.mlr.press/v37/sohl-dickstein15.html.

Dejan Stancevic, Florian Handke, and Luca Ambrogioni. Entropic time schedulers for generative diffusion models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=EfDIApcjgI.

Hannes Stark, Bowen Jing, Chenyu Wang, Gabriele Corso, Bonnie Berger, Regina Barzilay, and Tommi Jaakkola. Dirichlet flow matching with applications to dna sequence design. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report, 2025.

Jian Ye, Ning Ma, Thomas L. Madden, and James M. Ostell. Igblast: an immunoglobulin variable domain sequence analysis tool. *Nucleic Acids Research*, 07 2013. URL https://doi.org/10.1093/nar/gkt382.

A LIMITATIONS AND FUTURE WORK

Due to the broad scope of the current paper, spanning several modalities and noising schemes, a detailed analysis of how diffusion-language-modeling choices such as confidence unmasking, block decoding, and remasking interact with the posterior is absent. We leave this to future work. A more fundamental limitation is that the posterior is updated only after the state has changed, so a token that strongly determines a class affects the estimate only once it has been unmasked. For the NOT gate this is a genuine shortcoming: ideally the posterior would reshape the transition probabilities before such tokens are generated, preventing the unwanted class rather than reacting to it. We view this anticipatory formulation as a promising direction.

B ADDITIONAL DERIVATIONS

In this appendix additional useful derivations are given. In Appendix B.1, the difference between how the common classifier-freee guidance method works in the discrete and continuous domains is discussed. Thereafter, in Appendix B.2, a detailed derivation of all logical operators introduced in the main body is given. Then, in Appendix B.3 an important note regarding under which samplers our online posterior estimate is equivalent to the ground-truth marginal computed posterior.

B.1 CONTINUOUS VS DISCRETE CFG

In the continuous diffusion literature guidance is seen as a multiplicative corruption/sharpening at the marginal level which results in an additive mixture of model predictions or score functions. In the discrete diffusion literature however it is common to see this corruption as a multiplicative tweaking of the predicitons themselves which results in an additive mixture of the predicted logits. This difference is not well known yet quite significant for our work as here we start with the same assumptions as in the continuous case, namely corruptions on the marginal level, and derive the rate function that would yield the discrete analogue of continuous CFG, which we show to be an additive mixture of the rate functions.
Let us frame this in more detail mathematically. Classifier-free guidance starts of from a sharpening of the marginals:

$$p^{\text{CFG}}(x_t) = p(x_t)\frac{p(x_t|c)^\lambda}{p(x_t)^\lambda} \tag{20}$$

When looking at the infinitesimal generator of that distribution, which in the continuous case is its score function, one finds:

$$\nabla \log p^{\text{CFG}}(x_t) = \nabla \log p(x_t) + \lambda\Big(\nabla \log p(x_t|c) - \nabla \log p(x_t)\Big) \tag{21}$$

Using tweedies formula the score function depends linearly on the denoiser prediction $\hat{x}_0 = \mathbb{E}[x_0|x_t]$ such that in practice CFG can be implemented as a linear combination of the denoisers prediction:

$$\hat{x}_0^{\text{CFG}}(x_t) = \hat{x}_0(x_t) + \lambda\Big(\hat{x}_0(x_t|c) - \hat{x}_0(x_t)\Big) \tag{22}$$

In our derivation of the *and* operation we start from the exact same assumption and find that for a multiplicative mixture of the marginals one again needs to add their infinitesimal generators, which in the discrete case are given by the rate matrix R_t instead of the score functions $\nabla \log p(x_t)$. This time, the rate functions linearly depend on the models token predictions $p(x_0|x_t)$, hence these should be added together to obtain CFG.

In the discrete diffusion literature, guidance is derived from a different assumption, namely a multiplicative mixture of the model predictions. This is erroneously described as the assumption used in the continuous case,[5] but beyond that this is a perfectly fine modification of the generative process. More precisely, the assumption commonly used in the discrete case is the following:

$$p^{\text{D-CFG}}(x_0|x_t, c) = p(x_0|x_t)\frac{p(x_0|x_t, c)^\lambda}{p(x_0|x_t)^\lambda} \tag{23}$$

[5]The error is most visible in the notation used for what they define as a score function $\nabla \log p(x_s|x_t, c)$, which is incorrect. The score function is the gradient of the log likelihood of the marginal, namely $\nabla \log p(x_t|c)$, and not of the transition kernel.

Typically models are trained to predict logits, that is log probabilities, in which the decomposition becomes:

$$\log p^{\text{D-CFG}}(x_0|x_t, c) = \log p(x_0|x_t) + \lambda\Big(\log p(x_0|x_t, c) - \log p(x_0|x_t)\Big) \tag{24}$$

The main advantage of this modification is that by changing the distribution at the logits level, there is no risk to obtain an invalid distribution, because a softmax normalization will always be applied afterwards.

In the following section we will try to unravel what the differences are between these two assumptions and what possible pitfalls might be. Let us first try to see how one assumption transfers into the second one:

$$p^{\text{D-CFG}}(x_0|x_t, c) = p(x_0|x_t)\frac{p(x_0|x_t, c)^\lambda}{p(x_0|x_t)^\lambda} \tag{25}$$

$$= p(x_0)\frac{p(x_t|x_0)}{p(x_t)}\frac{\Big(p(x_0|c)\frac{p(x_t|x_0,c)}{p(x_t|c)}\Big)^\lambda}{\Big(p(x_0)\frac{p(x_t|x_0)}{p(x_t)}\Big)^\lambda} \tag{26}$$

$$= p(x_0)\frac{p(x_0|c)^\lambda}{p(x_0)^\lambda}\frac{p(x_t|x_0)\frac{p(x_t|x_0,c)^\lambda}{p(x_t|x_0)^\lambda}}{p(x_t)\frac{p(x_t|c)^\lambda}{p(x_t)^\lambda}} \tag{27}$$

Realizing that due to the conditional independence of the noising process we have $p(x_t|x_0) = p(x_t|x_0, c)$, and if we further note that $p^{\text{CFG}}(x_t) = p(x_t)\frac{p(x_t|c)^\lambda}{p(x_t)^\lambda}$ we find:

$$p^{\text{D-CFG}}(x_0|x_t, c) = p(x_0)\frac{p(x_0|c)^\lambda}{p(x_0)^\lambda}\frac{p(x_t|x_0)}{p(x_t)\frac{p(x_t|c)^\lambda}{p(x_t)^\lambda}} \tag{28}$$

$$= p^{\text{CFG}}(x_0|c)\frac{p(x_t|x_0)}{p^{\text{CFG}}(x_t|c)} \tag{29}$$

$$\tag{30}$$

Now, one might think that we can invoke the conditional independence of the noising process to the sharpened distribution (i.e., setting $p(x_t|x_0) = p^{\text{CFG}}(x_t|x_0, c)$ to obtain through Bayes that $p^{\text{D-CFG}}(x_0|x_t, c) = p^{\text{CFG}}(x_0|x_t, c)$. This is however erroneous. Silently doing this would require $p^{\text{CFG}}(x_t|c) = \sum_{x'_0} p(x_t|x'_0)p^{\text{CFG}}(x_0|c)$,[6] i.e., that the tilted marginals correspond to the noised version of the tilted data distribution. When applying a multiplicative mixture at the distributional level this is not true, as the convolution behind the noising operation does not commute with the multiplicative mixing operation. In fact a similar result is the reason that CFG is heavily criticized in the continuous case, the sharpening of the marginals does not result in samples that correspond to the expected sharpened data distribution.

So we find that the difference between the two process lies in the non-commutativity of the noising and mixing operations, or mathematically by the non-commutativity of the convolution and the multiplication. Typical continuous CFG corresponds to sharpening all the marginals individually, while discrete CFG yields the denoiser that retrieves the sharpened data distribution. The difference between the two orders of mixing and noising are further illustrated in Figure 5.

To prove this second fact let's derive which prediction $p^\lambda(x_0|x_t)$, or rate function, would yield a λ-sharpened data distribution $p^\lambda(x_0|c) = p^{\text{CFG}}(x_0|c) = p(x_0)\frac{p(x_0|c)^\lambda}{p(x_0)^\lambda}$. For this let us write out the true predictor of that distribution namely:

$$p^\lambda(x_0|x_t, c) = \frac{p(x_t|x_0)p^\lambda(x_0|c)}{\sum_{x'_0} p(x_t|x'_0)p^\lambda(x'_0|c)} \tag{31}$$

[6]This needs to hold to apply Bayes' rule as a valid joint distribution should be obtained. Alternatively the sum over x_0 in eq. (30) should yield 1.

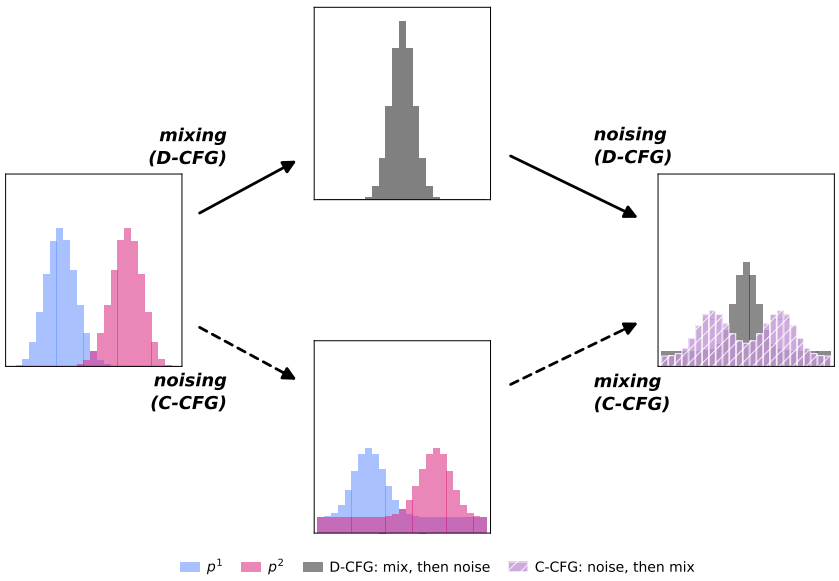


Figure 5: Illustration of the non-commutativity of noising and mixing in the case of a multiplicative mixture. Continuous CFG first noises the data distribution and mixes the marginals, whereas discrete CFG first mixes the distributions and then only noises.

Or by flipping the conditioning using Bayes and only looking at the x_0 dependence as the rest is simply a normalization constant that will be taken care off by the softmax:

$$p^\lambda(x_0|x_t,c) \propto_{x_0} \big(p(x_t|x_0)p(x_0)\big)^{1-\lambda}\big(p(x_t|x_0,c)p(x_0|c)\big)^\lambda \tag{32}$$
$$= \big(p(x_0|x_t)p(x_t)\big)^{1-\lambda}\big(p(x_0|x_t,c)p(x_t|c)\big)^\lambda \tag{33}$$
$$= p(x_0|x_t)^{1-\lambda}p(x_0|x_t,c)^\lambda p(x_t)^{1-\lambda}p(x_t|c)^\lambda \tag{34}$$
$$\propto_{x_0} p(x_0|x_t)^{1-\lambda}p(x_0|x_t,c)^\lambda \tag{35}$$
$$\tag{36}$$

Hence we find that the exact denoiser, or generator, that will yield samples from the tilted data distribution is precisely the one implemented in the discrete version of CFG. Hence this proves that by adding the logits together, the generator sampling from the sharpened data distribution is obtained.

B.2 Detailed derivation of Logical Operators: AND, OR, NOT

In this section, the derivations leading up to the different logical operators outlined in the main paper is given in more detailed. The same ordering as in the main body is kept, starting with the derivation for the AND operator, then the OR operator and finally the NOT operator. For all cases the resulting process is analyzed when mixing the data distributions as well as when mixing the marginals themselves. Naturally as outlined in the main body for the OR and NOT operators both result in the same process due to the commutativity of their specific additive mixing operations with noising.

AND-gate. ***(data-level mixture)*** In this case we consider a multiplicative mixture on the data distribution level, i.e., $p_0^\Lambda(x_0) \propto \prod_{i=1}^N p_0(x_0|c_i)$. The key challenge is to express the predictor that leads to the mixture distribution, denoted by $p^\Lambda(x_0|x_t)$, as a function of the assumed known conditional predictors $p_0(x_0|c_i)$. Once this mixture predictor is known, we construct the according rate function using Eq.(4). The mixture denoiser can be computed by rewriting it as a data weighted

function of the noising kernel through Bayes:

$$p^{\Lambda}(x_0|x_t) = p(x_t|x_0)\frac{p_0^{\Lambda}(x_0)}{p_t^{\Lambda}(x_t)} \tag{37}$$

$$\propto p(x_t|x_0)\prod_{i=1}^{N} p_0^{\Lambda}(x_0|c_i) \tag{38}$$

$$\propto p(x_t|x_0)^{1-N}\prod_{i=1}^{N} p(x_t|x_0)p_0^{\Lambda}(x_0|c_i) \tag{39}$$

$$\propto p(x_t|x_0)^{1-N}\prod_{i=1}^{N} p^{\Lambda}(x_0|x_t,c) \tag{40}$$

Or thus the predictions distributions should be multiplied with each other, this is more commonly implemented as adding logits together:

$$\log p(x_0|x_t) \propto \sum_{i=1}^{N} \log p(x_0|x_t,c) \tag{41}$$

From this predictor a rate function can be constructed through Eq. (4). This is precisely how CFG is commonly implemented in the discrete diffusion literature, i.e., such as in eq (8).

AND gate. ***(marginal-level mixture)*** On the other hand, for the multiplicative mixing an approach reminiscent of the continuous diffusion literature can be applied, namely considering a multiplicative mixture of the marginals defined as $p_t^{\Lambda}(x_t) \propto \prod_{i=1}^{N} p_t(x_t|c_i)$. Here it's simplest to directly derive the joint rate function through:

$$\frac{\mathrm{d}p_t^{\Lambda}(x_t)}{\mathrm{d}t} = \sum_i \frac{\mathrm{d}p_t(x_t|c_i)}{\mathrm{d}t}\prod_{j\neq i}^{N} p(x_t|c_j) \tag{42}$$

$$= \sum_i R_i p_t(x_t|c_i)\prod_{j\neq i}^{N} p(x_t|c_j) \tag{43}$$

$$= \sum_i R_i \prod_{j}^{N} p(x_t|c_j) \tag{44}$$

$$= \Big(\sum_i R_i\Big)p_t^{\Lambda}(x_t) \tag{45}$$

Hence the rate matrices of the joint process satisfies:

$$R_t^{\Lambda} = \sum_{i=1}^{N} R_t^i. \tag{46}$$

By construction R_t^{Λ} inherits the required properties of a valid rate matrix, namely non-negative off-diagonal entries and rows summing to zero, so the resulting process remains valid. The shape of this combination is also intuitive: as long as one of the two conditions is not satisfied, the corresponding rate will try to strongly modify the outcome. Only once x_t satisfies both conditions will both rate functions remain reasonably small. This is entirely analogous to how CFG is implemented in the continuous diffusion literature, namely as an addition of two infinitesimal generators; in the latter a score function, and in this case a rate function.

OR-gate. ***(data-level mixture)*** In this context, mixing multiple distributions together in data space happens through a sum over the parts, i.e. $p_1^{\Lambda}(x_1) = \sum_{i=1}^{N} p(c_i)p_1(x_1|c_i)$. Looking at how the denoiser for this distribution can be expressed as a function of the conditional denoisers $p_1(x_1|x_t,c_i)$,

one has:

$$p_1^\Lambda(x_1|x_t) = p(x_t|x_1)\frac{p_1^\Lambda(x_1)}{p_t^\Lambda(x_t)} \tag{47}$$

$$= p(x_t|x_1)\sum_{i=1}^{N} p(c_i)\frac{p_1(x_1|c_i)}{p_t^\Lambda(x_t)} \tag{48}$$

$$= \sum_{i=1}^{N} p(c_i)\frac{p(x_t|x_1)p_1(x_1|c_i)}{p_t^\Lambda(x_t)} \tag{49}$$

$$= \sum_{i=1}^{N} p(c_i)\frac{p_t(x_t|c_i)}{p_t^\Lambda(x_t)}p_1(x_1|x_t, c_i) \tag{50}$$

$$= \sum_{i=1}^{N} p(c_i|x_t)p_1(x_1|x_t, c_i). \tag{51}$$

Hence, the denoiser of the mixture distribution is given by a posterior-weighted average of the conditional denoisers. The corresponding rate function can then be constructed from this mixture denoiser using Eq. (4).

OR-gate. ***(marginal level mixture)*** For the additive operation behind the OR gate $p_t^\Lambda(x) = \sum_{i=1}^{N} p(c_i)\, p_t(x_t|c_i)$ which can be interpreted as an unconditional model $p_t(x)$ in this setting, with as mixture weights the priors $p(c_i)$, one can again derive the joint rate function directly:

$$\frac{\mathrm{d}p_t^\Lambda(x_t)}{\mathrm{d}t} = \sum_i p(c_i)\frac{\mathrm{d}p_t(x_t|c_i)}{\mathrm{d}t} \tag{52}$$

$$= \sum_i R_t^i p(c_i)p_t(x_t|c_i) \tag{53}$$

$$= \Big(\sum_i p(c_i)\frac{p_t(x_t|c_i)}{p(x_t)}R_t^i\Big)p_t(x_t) \tag{54}$$

$$= \Big(\sum_i p(c_i|x_t)R_t^i\Big)p_t^\Lambda(x_t) \tag{55}$$

Where in the second line we multiply and divide by the mixture marginal $p_t^\Lambda(x_t)$. In other words,

$$R_t^\Lambda = \sum_{i=1}^{N} p(c_i \mid x_t)\, R_t^i. \tag{56}$$

Importantly, the posterior quantity is interpreted here as a classifier *restricted* to the specified conditions. This is precisely the mechanism that allows a dataset-specific predictor to be learned while training only on instances from that dataset. A predictor approximating the datapoint-specific distributions $p(x_0 \mid x_t)$, with each datapoint viewed as a class, can be averaged over the corresponding deterministic datapoint-specific rates to recover a global rate, an idea closely related to the construction used in Flow Matching (Lipman et al., 2023; Gat et al., 2024). The results show that this construction extends to arbitrary partitions of the datapoints. Beyond this theoretical connection, the OR gate is not strictly necessary for class-conditional generation, since sampling a condition according to its prior, and subsequently using the corresponding conditional model, yields the same desired distribution.

NOT-gate. ***(data-level mixture)*** For the NOT gate, i.e., removing a particular part from a larger distribution, the resulting distribution corresponds to the complement of that part. In particular, removing condition c_i from the unconditional distribution gives

$$p_1^\Lambda(x_1) = p_1(x_1|\neg c_i) = \frac{1}{1-p(c_i)}\big(p_1(x_1) - p(c_i)p_1(x_1|c_i)\big). \tag{57}$$

Importantly, because the addition commutes with the convolution based noising kernel, the same identity holds at the marginal level:

$$p_t^{\Lambda}(x_t) = p_t(x_t|\neg c_i) = \frac{1}{1-p(c_i)}\big(p_t(x_t) - p(c_i)p_t(x_t|c_i)\big). \tag{58}$$

Deriving the predictor for this distribution gives:

$$p_1^{\Lambda}(x_1|x_t) = p(x_t|x_1)\frac{p_1^{\Lambda}(x_1)}{p_t^{\Lambda}(x_t)} \tag{59}$$

$$= \frac{p(x_t|x_1)\big(p_1(x_1) - p(c_i)p_1(x_1|c_i)\big)}{p_t(x_t) - p(c_i)p_t(x_t|c_i)} \tag{60}$$

$$= \frac{p_t(x_t)p_1(x_1|x_t) - p(c_i)p_t(x_t|c_i)p_1(x_1|x_t,c_i)}{p_t(x_t) - p(c_i)p_t(x_t|c_i)} \tag{61}$$

$$= \frac{p_1(x_1|x_t) - p(c_i|x_t)p_1(x_1|x_t,c_i)}{1-p(c_i|x_t)} \tag{62}$$

Hence, the denoiser for the NOT operation can be written as a posterior-dependent modification of the unconditional denoiser:

$$p_1^{\Lambda}(x_1|x_t) = p_1(x_1|x_t) - \frac{p(c_i|x_t)}{1-p(c_i|x_t)}\Big(p_1(x_1|x_t,c_i) - p_1(x_1|x_t)\Big). \tag{63}$$

The corresponding rate function can then be constructed from this predictor using Eq. (4).

NOT-gate. ***(marginal-level mixture)*** The removal, or avoidance, of a certain class only given it's class-specific rate can be mathematically expressed as the subtraction of a conditional distribution from its unconditional distribution $p_t^{\Lambda}(x_t|c_i) \propto p_t(x_t) - p(c_i)\,p_t(x_t|c_i)$. The joint rate can again be derived as:[7]

$$\frac{\mathrm{d}p_t^{\Lambda}(x_t)}{\mathrm{d}t} = \frac{1}{1-p(c_i)}\Big(R_t p_t(x_t) - p(c)R_t^i p_t(x_t|c)\Big) \tag{64}$$

$$= \frac{1}{1-p(c_i)}\Big(R_t\big(p_t(x_t) - p(c_i)p_t(x_t|c_i)\big) - p(c_i)p_t(x_t|c)\big(R_t^i - R_t\big)\Big) \tag{65}$$

$$= \Big(R_t - \frac{p_t(c_i|x_t)}{1-p_t(c_i|x_t)}\big(R_t^c - R_t\big)\Big)p_t^{\Lambda}(x_t) \tag{66}$$

We therefore find:

$$R_t^{\Lambda} = R_t - \frac{p(c \mid x_t)}{1-p(c \mid x_t)}\big(R_t^c - R_t\big). \tag{67}$$

This equation strongly resembles a classifier-free guidance equation, in which the posterior ratio can be seen as a *dynamic* guidance scale that is small if the undesired class if unlikely to be generated and that becomes asymptotically large if that class becomes likely to be generated. Again, this is entirely analogous to what is found in the continuous case, with as single difference that the continuous generators, or score-functions, are replaced by their discrete analogues, the rate functions Koulischer et al. (2025b).

B.3 History independence of the posterior

An important point is that the posterior tracked throughout inference is the likelihood conditioned on the whole denoising trajectory and not only its least noisy state x_t. This is a consequence of the way the posterior estimation is constructed through the Markov chain:

$$p(c|x_0,..x_t) = p(c)\frac{p(x_0,...,x_t|c)}{p(x_0,...,x_t)} \tag{68}$$

$$= p(c)\frac{p(x_0,...,x_{t-1}|c)}{p(x_0,...,x_{t-1})}\frac{p(x_t|x_{t-1},c)}{p(x_t|x_{t-1},c)} \tag{69}$$

$$= p(c|x_0,...,x_{t-1})\frac{p(x_t|x_{t-1},c)}{p(x_t|x_{t-1},c)} \tag{70}$$

$$\tag{71}$$

[7] The same remark as for the OR gate holds here.

Which corresponds to the log posterior update scheme described:

$$\log p(c|x_0, ..x_t) = \log p(c|x_0, ..., x_{t-1}|c) + \log p(x_t|x_{t-1}, c) - \log p(x_t|x_{t-1}, c) \tag{72}$$

$$\tag{73}$$

Crucially however, thanks to the conditional independence of the noising process, the following holds:

$$p(c|x_0, ..x_t) = p(c)\frac{p(x_0, ..., x_t|c)}{p(x_0, ..., x_t)} \tag{74}$$

$$= p(c)\frac{p(x_t|c)}{p(x_t)}\frac{p(x_0, ..., x_{t-1}|x_t, c)}{p(x_0, ..., x_{t-1}|x_t)} \tag{75}$$

$$= p(c|x_t) \tag{76}$$

$$\tag{77}$$

Here the important part is that the dependence on the whole trajectory drops out due to the fact that conditional and unconditional sequences get noised in the same way i.e., the noising is independent on the condition. However, for this to hold the sampler needs to be memoryless in the sense that when the denoising process is reversed it should give rise to the noising process itself. This subtlety is not satisfied by all samplers, such as the ODE sampler of DFM that we use in this paper. An example of a moment where such a subtlety is relevant is when using the uniform model. Without a purely stochastic ancestral sampler the quantity tracked through inference is not exactly equal to the marginal computed posterior, as shown in Figure 12 . We find however that the DFM sampler works best in practice on trained models, such as the ones trained on MNIST or OAS. An advantage of the masking noising scheme is that this equality is always satisfied, meaning that the posterior tracking will always give rise to the true marginal computed posterior. This is because in the case of a masked model the least noisy state truly holds all the information regarding the final state when compared to it's history. In essence the older states contain the exact same words but with a higher masking rate.

C Additional details regarding synthetic data evaluation

To verify our theoretical framework, in particular the online posterior estimation and the logical operators, we first work in a synthetic regime in which everything is analytically tractable. Appendix C.1 describes the constructed dataset and how the conditional and unconditional predictors are obtained in closed form, the latter both by exhaustive enumeration and as a mixture of the conditionals. Appendix C.2 compares the online posterior estimate to the exact ground-truth quantity. Appendix C.3 validates the logical operators AND, OR and NOT, alongside an illustration of the differences between continuous and discrete CFG. All experiments are run for both masking and uniform noise.

C.1 Dataset construction

We use binary sequences of length $L = 20$ over the tokens "*A*" and "*B*" visualized in red and blue, with either two or four classes specifiable. Classes are defined by the majority likelihood of one token in a certain region. For two classes the majority token dominates the entire sequence, while for four classes it dominates only the first or the second half, with the other half remaining uniform. The four-class set-up is mainly used to obtain non-mutually-exclusive classes, such that logical operations like AND can be verified. This is summarized in Table 3.

Within a class all token positions are independent, so that $p_1(x_1|c) = p_A^{N_A}(1-p_A)^{N_B}$ with N_A and N_B the number of "*A*" and "*B*" tokens. With this set-up every sequence has a non-zero likelihood under every class: even a sequence consisting solely of "*B*" can be sampled by the "*A*" model. The corruption also acts independently per position, so the noisy marginals factorize as well. With κ_t the noise schedule and $V = 2$ data tokens, uniform noise gives

$$p_t(x_t^j = A|c) = \kappa_t p_A + \tfrac{1-\kappa_t}{V}, \qquad p_t(x_t^j = B|c) = \kappa_t(1-p_A) + \tfrac{1-\kappa_t}{V}, \tag{78}$$

while for masking a position is A with probability $\kappa_t p_A$, B with probability $\kappa_t(1-p_A)$ and `MASK` with probability $1-\kappa_t$. The conditional marginal $p_t(x_t|c)$ is the product over positions. This

class	$p_1(x_1^j = A\|c)$ for $j \in \mathcal{H}_1$	$p_1(x_1^j = A\|c)$ for $j \in \mathcal{H}_2$
A	p_A	
B	$1 - p_\mathrm{A}$	
A_1	p_A	$1/2$
B_1	$1 - p_\mathrm{A}$	$1/2$
A_2	$1/2$	p_A
B_2	$1/2$	$1 - p_\mathrm{A}$

Table 3: Synthetic data construction. The sequences are split into two halves $\mathcal{H}_1, \mathcal{H}_2$ of 10 tokens each, filled in with a majority token whose strength is set by a tunable parameter p_A. For two classes both halves share the same majority likelihood. For four classes only one half is dominated by a token while the other is uniform, which increases the overlap between classes and makes logical operations such as AND more interesting.

also gives the conditional predictors: under masking a masked position simply predicts token "*A*" with p_A and token "*B*" with $1 - p_A$. Under uniform noise the observed token x_t^j is not a reliable copy of the clean token, since with probability $1 - \kappa_t$ it was replaced by a uniformly random one, so the conditional predictor weighs the class prior against how well each candidate explains the observation: $p(x_1^j = a|x_t^j, c) \propto p_1(x_1^j = a|c)\, p_t(x_t^j|x_1^j = a)$ with $p_t(x_t^j = v|x_1^j = u) = \kappa_t \mathbb{1}[v = u] + \frac{1-\kappa_t}{V}$, where $p_1(x_1^j = A|c)$ is the probability listed in Table 3 and $p_1(x_1^j = B|c)$ its complement. The normalization is exactly the marginal of Eq. 78, and the prediction interpolates between the class prior $p_1(x_1^j|c)$ at $\kappa_t = 0$ (pure noise, no information) and the observed token at $\kappa_t = 1$ (no noise). Now to obtain an unconditional model that correctly samples from this dataset, two options can be used. The first is a summation over the mixture dataset: construct all $2^{20} \approx 1\text{M}$ sequences, assign each the average likelihood under the conditional models, $p_1(x_1) = \sum_c p(c)\, p_1(x_1|c)$, and loop over them. The noisy marginal and the unconditional denoiser are then obtained by explicit summation, with $p_t(x_t|x_1)$ the corruption kernel:

$$p_t(x_t) = \sum_{x_1} p_1(x_1)\, p_t(x_t|x_1), \qquad p(x_1^j = a|x_t) = \frac{\sum_{x_1 :\, x_1^j = a} p_1(x_1)\, p_t(x_t|x_1)}{p_t(x_t)}. \tag{79}$$

The second, far simpler option is to define the unconditional model as the mixture of all conditionals, i.e., an OR gate over the classes, $p(x_1^j|x_t) = \sum_c p(c|x_t)\, p(x_1^j|x_t, c)$. The sole difficulty is the posterior $p(c|x_t)$, which is perfectly tractable through Bayes' rule and Eq. 78, since $p_t(x_t) = \sum_c p(c)\, p_t(x_t|c)$. Both constructions are exactly equivalent, which we also confirm numerically using the enumeration. In all further experiments we use the OR-gate version, as it does not require an exhaustive summation over the 2^L sequences.

C.2 Validation of posterior estimation

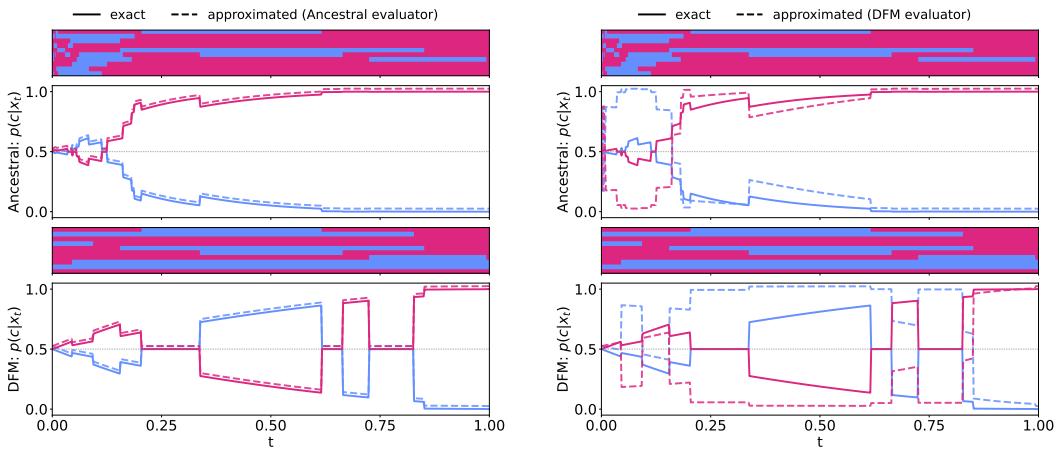
The online posterior estimation can be compared to the exact quantity, given by the ratio of the conditional and unconditional marginals of Eq. 78:

$$p_t(c|x_t) = p(c)\frac{p_t(x_t|c)}{p_t(x_t)} \tag{80}$$

The posterior can also be tracked with the log-posterior update derived in the main body:

$$\log p_t(c|x_0, ..., x_t) = \log p_{t-1}(c|x_0, ..., x_{t-1}) + \log p_{t-1}(x_t|x_{t-1}, c) - \log p_{t-1}(x_t|x_{t-1}) \tag{81}$$

This equation still allows for some freedom, namely in the choice of transition likelihoods, or sampler. In this work we mainly focus on two specific cases the kinetic optimal discrete flow matching sampler and the stochastic ancestral sampler. Also, note that it is perfectly sound to use one choice for sampling and another for tracking the posterior. As shown in Appendix B.3, if a memoryless stochastic sampler is used to compute the transition likelihoods, the following holds $p_t(c|x_0, ..., x_t) = p_t(c|x_t)$: the entire posterior information is contained in the least noisy point of the trajectory, i.e., the current state. For *masking* this always holds, as tokens remain unmasked once revealed, and the flow matching and ancestral samplers coincide. For *uniform* noise it holds only if the denoising kernel is derived as the reverse of the corruption kernel, i.e., for an ancestral sampler.



(a) Using ancestral transition likelihoods

(b) Using dfm transition likelihoods

Figure 6: Illustration of the online posterior evaluation under different samplers and evaluators. In (a), samples are taken from an ancestral sampler at the top and a dfm sampler at the bottom and the posterior likelihoods are updated using the ancestral transition likelihoods. In this case the tracked posterior is equal to the ground truth posterior. In (b), the exact same samples are shown but with this time the DFM transition likelihoods used for the posterior estimation, in which case the tracked quantity is not equal to the exact one anymore.

Figure 1a shows that the tracked posterior is exact for both noise types when the ancestral sampler is used. In Figure 6a we show that if the posterior likelihood is computed using the transition likelihoods predicted by the ancestral sampler, the tracked quantity is equal to the ground truth quantity regardless of whether an ancestral or dfm sampler is used to generate the sample. On the other hand in Figure 6b, we show that if the transition likelihoods are computed using the flow matching sampler the tracked and exact quantities are no longer equal, i.e., there is a real difference between $p_t(c|x_0, ..., x_t)$ and $p_t(c|x_t)$. These do however remain strongly correlated, with for instance the dashed and full curves going up and down together, simply with the wrong scale. An important note is that in practice, on real trained models, we find that the flow matching sampler yields to more stable and accurate posterior estimation. We hypothesize that this discrepancy between theory and practice is also caused by the fact that the deterministic ODE shows smoother less stochastic trajectories, such that the posterior information is easier to discern for the trained model. A precise understanding of this phenomenon is however left as future work.

C.3 Validation of the logical operators

To validate the logical operators we compare the generated samples to the exact target distribution of each gate. Comparing full sequences would require far too many samples, so we instead count the number of tokens "A" in the first and in the second half, (n_1, n_2). As all classes are exchangeable within a half, each target (the AND-law, and the mixtures underlying OR and NOT) depends on a sequence only through these counts, such that its exact distribution $T(n_1, n_2)$ is easily computed. We then measure the KL divergence between the empirical distribution of (n_1, n_2) over M generated samples and T. We do this not only for the final samples, but also for the noisy states x_t at intermediate noise levels, such that Fig. 7 shows how the generated distribution approaches the target as denoising progresses.

For the AND gate it is important to also keep the forward kernel in the predictor. The exact data-level predictor for the AND gate is

$$\log p(x_1|x_t) = \sum_{i=1}^{N} \log p(x_1|x_t, c_i) + (1 - N) \log p_t(x_t|x_1) + \text{const}, \tag{82}$$

and not simply the sum of the logits of the conditional denoisers, as each of these already contains the forward kernel $p_t(x_t|x_1)$ once, while the target should contain it only once. For *masking* the

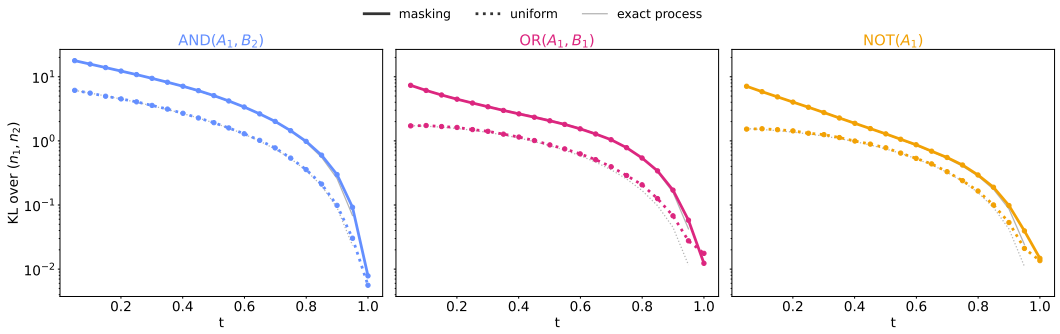


Figure 7: Evolution of the KL divergence between the distribution of (n_1, n_2) of the noisy state x_t and that of the clean target, for AND(A_1, B_2), OR(A_1, B_1) and NOT(A_1), under masking (full) and uniform (dotted) noise. In all cases the derived gates sample from the desired target distribution, illustrated by the KL distance between generated and reference samples going to zero.

additional term does not depend on x_1 and drops out of the softmax. For *uniform* noise however $p_t(x_t^j|x_1^j)$ favors the observed token, such that the term does depend on x_1 and has to be kept. As it only involves the forward kernel it is known exactly, without needing p_1 or the model. Importantly for a CFG-like mixture of conditional and unconditional distributions this factor does not exist as there, the base distribution is multiplied with a ratio of conditional and unconditional distributions. As such, this factor appears for both numerator and denominator and cancels out cleanly. The other two gates, OR and NOT, are weighted averages of the conditional denoisers and require no such correction.

D ADDITIONAL DETAILS REGARDING IMAGE GENERATION

In this section additional details and results on the image modality are provided. First, Appendix D.1 describes the architecture and the training hyperparameters. Second, Appendix D.2 shows ablations on the model performance, namely FID, precision, recall and C2ST as a function of the guidance scale and of the number of sampling steps. Then, Appendix D.3 the posterior estimations is analyzed in detail. Finally, Appendix D.4 contains an in depth discussion of the results obtained using the NOT gate.

D.1 TRAINING DETAILS

Our image models are based on the DiT architecture (Peebles & Xie, 2022), adapted to discrete tokens. Every pixel value is a token from a vocabulary of 256 values (plus a mask token for masking diffusion), such that an MNIST image is a sequence of 784 tokens. The tokens are embedded, processed by transformer blocks and decoded into a categorical distribution over the 256 pixel values for every pixel. The noise level t and the class label are provided through adaptive layer normalization (AdaLN-Zero), where the label embedding has an additional entry for the unconditional case. The model is trained to predict the clean image with a cross-entropy loss, which for masking is only computed on the masked tokens and for uniform noise on all tokens. Noise levels are sampled in a stratified way, using a linear noise schedule. Conditional models are trained with label dropout, while unconditional models are trained separately by always dropping the label. As is typical for diffusion models, an exponential moving average (EMA) of the weights is used for sampling. The hyperparameters, listed in Table 4, were set to standard values (Shi et al., 2024) and not tuned per noise type.

Architecture	DiT with AdaLN-Zero
Parameters	7.6M
Depth / hidden size / heads	6 / 256 / 4
Sequence length / vocabulary	784 / 256 (+ mask token)
Optimizer	AdamW
Learning rate	$4 \cdot 10^{-4}$
Betas / weight decay	(0.9, 0.999) / 0.01
Warmup steps / schedule	1000 / cosine decay
Batch size	512
Training steps	200k
Gradient clipping	1.0
EMA decay	0.9999
Label dropout	0.15
Noise schedule	linear
Noise level sampling	stratified in $[0, 1]$
Loss weighting	none

Table 4: Hyperparameters of the MNIST models. The unconditional models are trained separately with label dropout 1.0, i.e., without ever seeing the label. The cosine decay is defined over a horizon longer than the training run, so that the learning rate only decays to about $3 \cdot 10^{-4}$ at the end of training.

Figure 8 shows the validation loss per noise level during training. In both cases the loss decreases with t, as less information has to be inferred at lower noise levels. The highest noise levels converge first, presumably because they are dominated by uncertainty that cannot be reduced (many images are consistent with a nearly empty input), while the low noise levels keep improving until the end of training. The two noise types are not directly comparable, as masking averages over the masked tokens only. At low noise the loss is only over masked tokens, so the loss stays high, whereas for uniform noise most tokens are already noise-free and thus easy to guess, so the loss goes to zero.

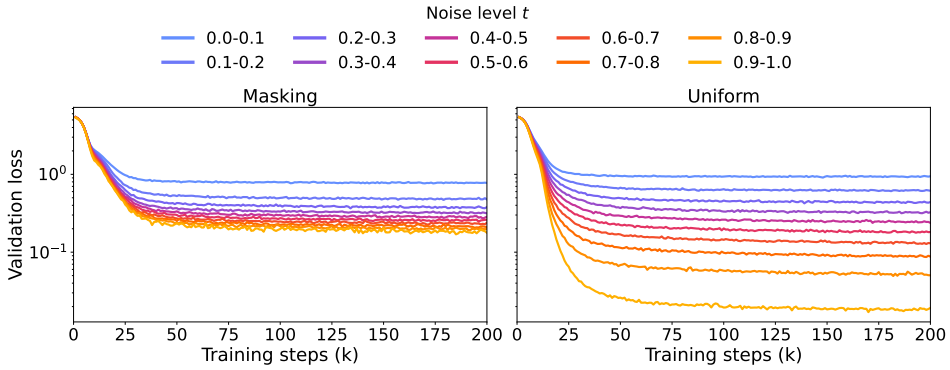


Figure 8: Validation loss during training of the MNIST models, averaged over bins of noise levels $t \in [0, 0.1], [0.1, 0.2], \ldots$, for masking (left) and uniform noise (right).

D.2 Model ablations

We evaluate the generated images with three metrics, all computed on InceptionV3 features of 4096 generated samples compared to the training set. The Fréchet inception distance (FID) is the Fréchet distance between Gaussian fits to the generated and the real features, and measures overall quality and diversity (lower is better). Precision and recall are k-nearest-neighbor based: precision is the fraction of generated samples that lie within the manifold of the real data (fidelity), recall the fraction of real samples that lie within the manifold of the generated data (diversity). The C2ST is the held-out accuracy of a logistic regression classifier that separates real from generated features, where 0.5 means the two are indistinguishable. Figure 9 shows the effect of the guidance scale on these metrics.

Conditional models (scale 1) are clearly better than unconditional ones (scale 0), and unconditional models that are trained separately (stars) are better than the unconditional mode of the jointly trained model.

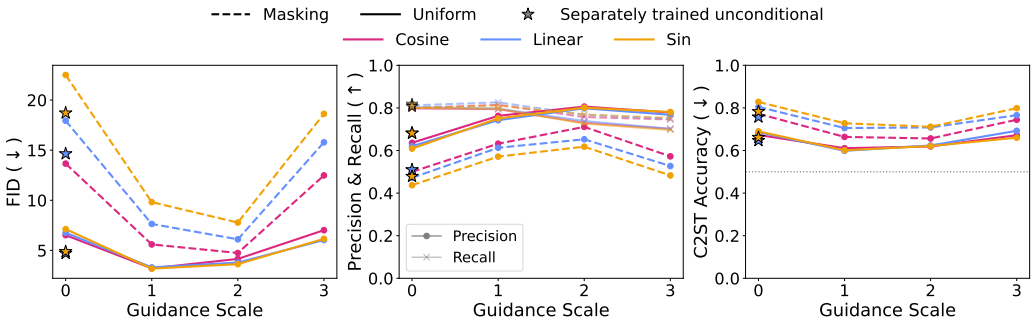


Figure 9: Sweep over the guidance scale for the MNIST-trained model. CFG is implemented as usual in the discrete case, i.e., a linear combination of logits. During inference 128 steps of the specified schedulers were taken. The stars correspond to the separately trained unconditional model.

Figures 10 and 11 show the dependence on the number of sampling steps. The quality increases with the number of steps in all settings. For masking there is a clear ranking of the sampling schedules, with the cosine schedule, which is slow early and fast late, performing best. This is expected: early in the denoising process the tokens have to be placed carefully, while late steps are easy as most of the image is already fixed. For uniform noise the schedule matters less, thanks to the ability to correct earlier mistakes. The differences between schedules shrink as the number of steps grows, as a schedule is only a smarter way to use a given compute budget and not a better model.

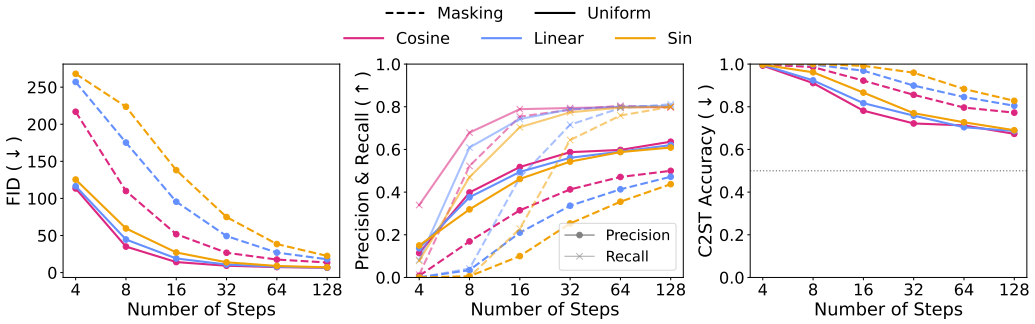


Figure 10: Sweep over the number of sampling steps for the unconditional MNIST-trained model. Masking and uniform models are evaluated under three schedulers; sine (fast at first and then slow), linear (constant sped) and cosine (slow at first and then fast). From left to right, the FID, the Precision and Recall and the C2ST accuracy.

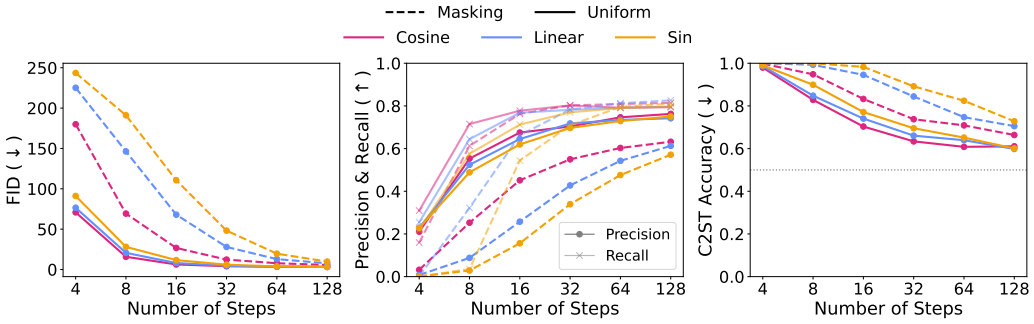


Figure 11: Sweep over the number of sampling steps for the conditional MNIST-trained model. Masking and uniform models are evaluated under three schedulers; sine (fast at first and then slow), linear (constant sped) and cosine (slow at first and then fast). From left to right, the FID, the Precision and Recall and the C2ST accuracy.

Finally, the uniform model outperforms the masking model, especially for a low number of sampling steps. This comparison should be taken with care, as the training hyperparameters were set to standard values and not tuned per method, so they may happen to suit the uniform model better.

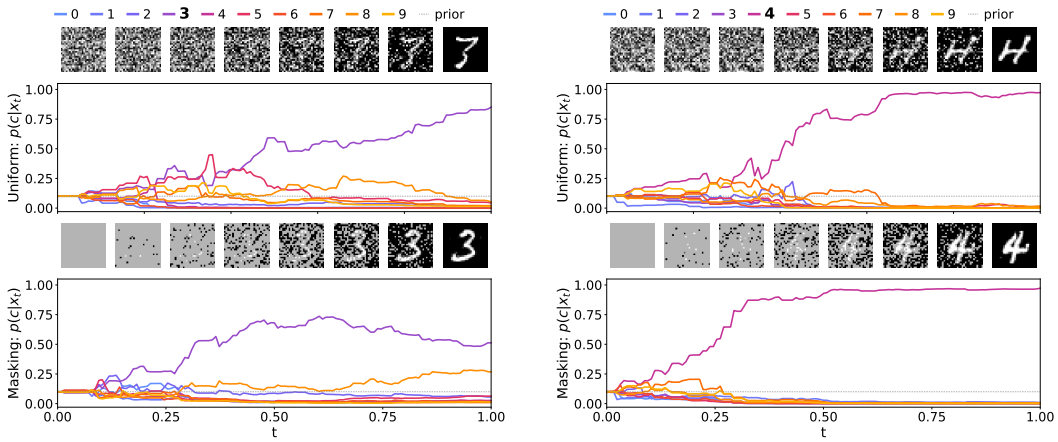
As the cosine scheduler (slow at first, fast at the end) outperforms all others all experiments use it during sampling. Furthermore, for maximal accuracy, all experiments were performed at 128 steps. All metrics in this modality are computed on 4096 samples.

D.3 Posterior Estimation

In the context of image generation, the posterior can be properly normalized by marginalizing over all classes. In practice, we achieve this by tracking the log-posterior for each class and subsequently applying a softmax over all classes. This results in a normalized and directly interpretable posterior distribution. Moreover, considering all classes jointly provides a clearer classification signal, as predictions are compared against one another rather than only against an unconditional prediction. Examples similar to the one presented in the main body in Figure 1b, including an example in which the trajectory is misclassified, are shown in Figure 12. In all examples, the posterior concentrates on the final class more rapidly under masking noise than under uniform noise.

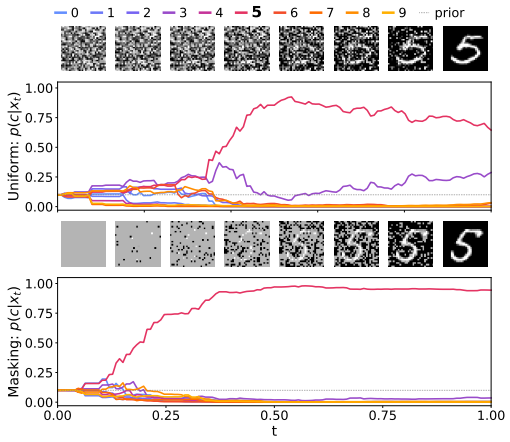
To quantify the effectiveness of the online posterior estimation, we consider two main metrics: one-vs-rest AUC and classification accuracy. Both metrics are evaluated as a function of denoising time, highlighting the main advantage of our approach: the posterior provides a classification signal before generation is complete. Ground-truth labels for the generated samples are obtained using a pretrained ViT classifier (Dosovitskiy et al., 2021). The accuracy is computed as the agreement rate between the top posterior prediction and the class assigned by the ViT. For the AUC, we use the macro-averaged one-vs-rest (OvR) extension of the ROC AUC to the multiclass setting. For each class c, samples assigned class c by the ViT are treated as positives and all remaining samples as negatives, while the posterior probability $p_t(c \mid \mathbf{x}_t)$ is used as the ranking score to compute the corresponding binary ROC AUC. The per-class AUCs are then averaged uniformly over the 10 classes. An AUC of 0.5 corresponds to chance level, which is also the expected value at $t = 0$, when no class-specific evidence has yet been accumulated.

The evolution of both metrics throughout denoising, for the three modalities evaluated in this work, is shown in Fig. 13. For a given noise schedule, the masking variant consistently accumulates class information more rapidly, which is expected from its inability to modify tokens once they have been unmasked. It is important to emphasize that we do not claim that our posterior provides a perfect classifier in this setting. Rather, its primary advantage is that it provides a proxy for the final class before generation is complete. As shown by both the AUC and accuracy curves, the classification signal largely saturates after approximately half of the generation process. This saturation is likely related to the out-of-distribution nature of the conditional models at low noise levels. As denoising progresses, the conditional marginals increasingly separate, such that evaluations of a model conditioned on one class become less meaningful for states belonging to another class. For example,

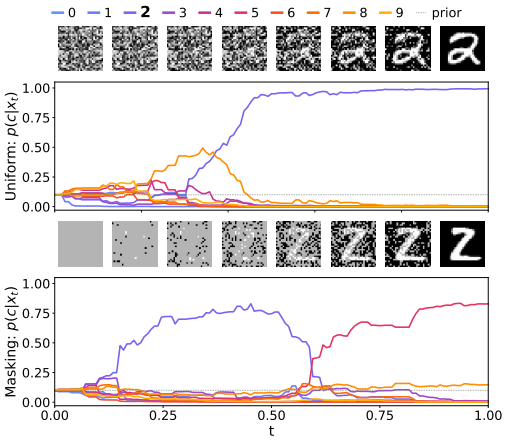


(a) Successful example

(b) Successful example



(c) Successful example



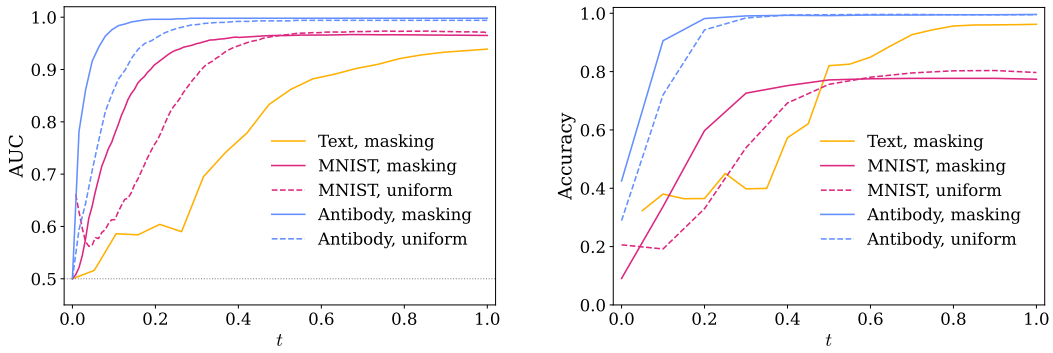
(d) Unsuccessful example

Figure 12: Illustration of the online posterior evaluation. In (a-c), trajectories for which the online posterior estimate recovers the final class for both masking and uniform noise are shown. In (d), an example is shown for which the masking model does not recover the correct class label. All shown trajectories use $\tau = 0.1$, a value taken from a calibration run on the NOT gate.

a model trained exclusively on "2"s cannot be expected to accurately predict the transition probabilities required to transform a partially denoised "4" into what it knows. This limitation occurs primarily once the sample has already been resolved, at which point the posterior has little additional utility. The posterior most informative precisely during the earlier stages of denoising, when class information is still emerging and an early estimate is most useful.

D.4 LOGICAL OPERATORS: NOT GATE

For the NOT gate, avoiding the generation of a given digit is a trade-off between image quality and safety. We track quality with the FID with respect to reference data that excludes the unwanted digit, and safety with the violation rate, i.e., the fraction of generated images that a ViT classifier assigns to the unwanted digit. As a static baseline we use negative prompting, a standard method from continuous diffusion, which is CFG with a negative guidance scale $-\lambda$. To stay consistent with the discrete literature, CFG is implemented as usual by an addition of logits. Figure 14 shows FID and violation rate for a sweep over λ. A very small scale already suffices to push away from the unwanted digit, while larger scales quickly harm the image quality.



(a) AUC (b) Accuracy

Figure 13: Evaluation of the online posterior estimation for different modalities during denoising. In (a), the OvR AUC. In (b), the accuracy.

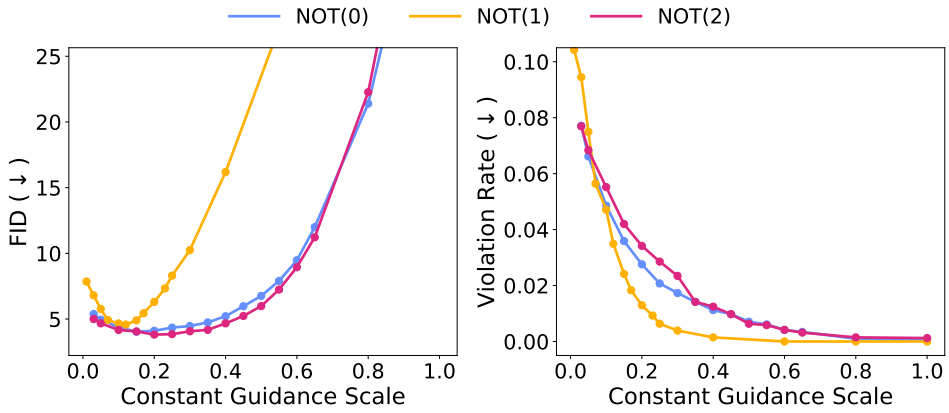


Figure 14: Static negative guidance (negative prompting): FID and violation rate as a function of the constant guidance scale λ, for NOT(0), NOT(1) and NOT(2).

Our method has two hyperparameters: the initial guidance scale s_0, which mathematically is a parameter of the prior, and the inverse temperature τ with which the posterior updates are rescaled. In this setting the results were of sufficient quality that the offset parameter δ was not needed. We perform a grid search over both hyperparameters and analyze both metrics (Figure 15). A larger s_0 increases the strength of the guidance, whereas a larger τ makes it fade faster. For small τ (≤ 0.05) the trade-off is almost independent of τ, while for larger values the guidance fades too quickly and more violations remain. We therefore fix $\tau = 0.01$ in all further experiments.

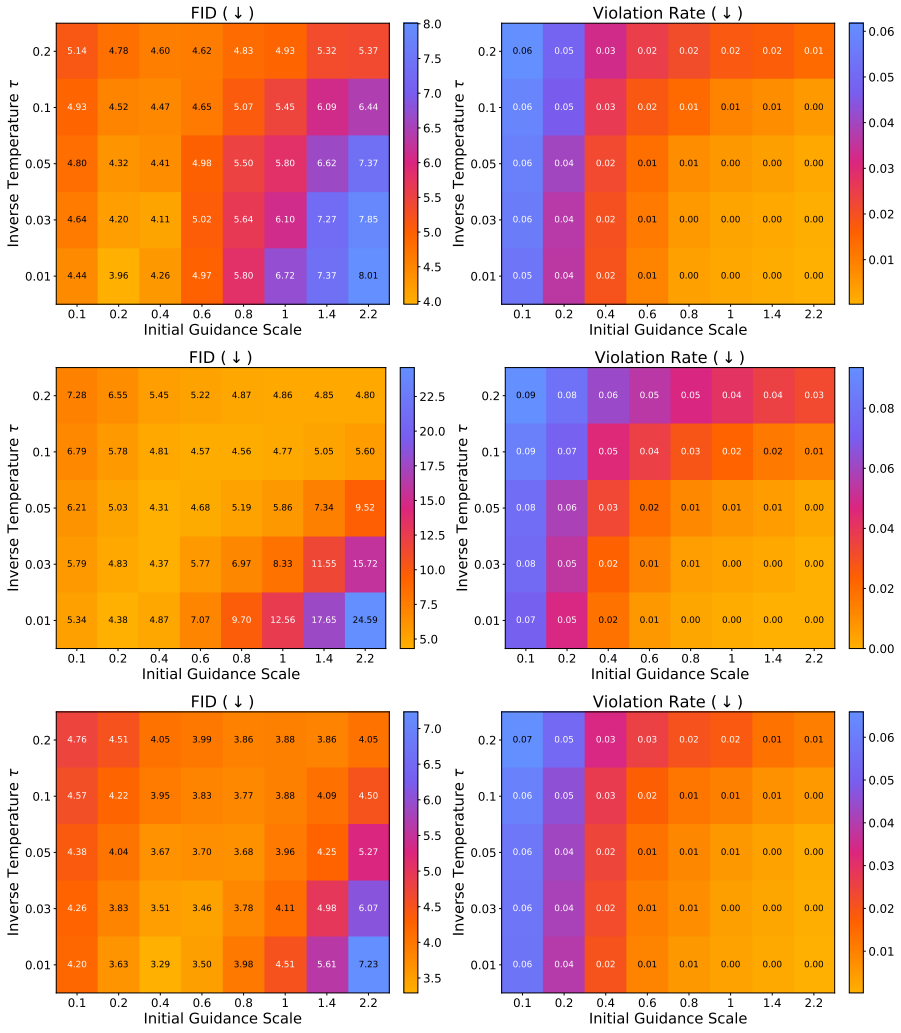


Figure 15: FID and violation rate of the dynamic NOT gate as a function of the initial guidance scale and the inverse temperature τ, for NOT(0), NOT(1) and NOT(2) (top to bottom). Amber is best, blue is worst.

Figure 16 shows FID and violation rate as a function of s_0. The initial guidance scale is an intuitive way of adding more guidance, and results in a trade-off profile very similar to that of the static baseline.

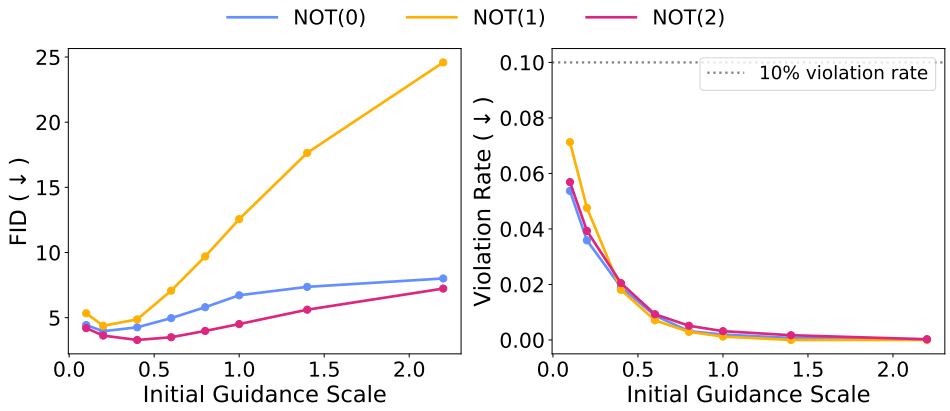


Figure 16: Dynamic negative guidance with $\tau = 0.01$: FID and violation rate as a function of the initial guidance scale, for NOT(0), NOT(1) and NOT(2).

To compare both methods, Figure 17 (a copy of the figure in the main body) plots FID against violation rate. The dynamic approach outperforms the static baseline, at the cost of one additional hyperparameter.

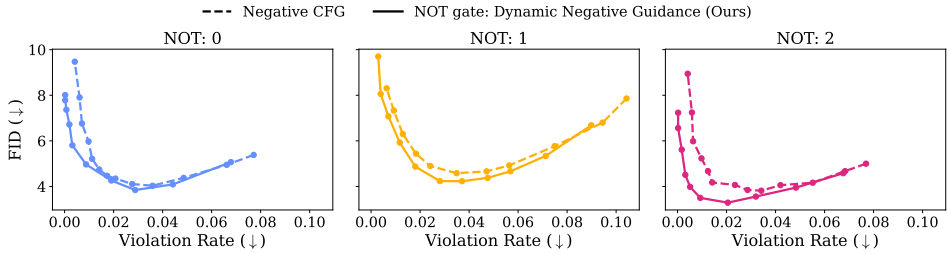


Figure 17: FID against violation rate for dynamic negative guidance (full) and the static baseline (dashed), for NOT(0), NOT(1) and NOT(2).

Interestingly, the profile of the guidance scale is very consistent (Figure 18). The guidance is mainly needed at the start of the generation and then decays roughly exponentially, where the trajectory only determines how fast. This suggests once more that for discrete image generation the first pixels are decisive, not only for the quality, as reflected by the choice of a cosine scheduler, but also for the class.

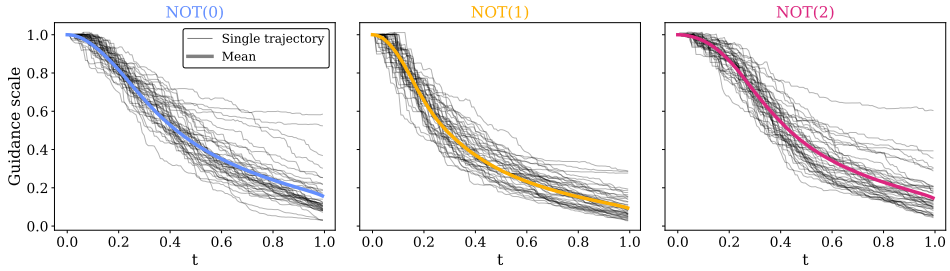


Figure 18: Guidance scale of the dynamic NOT gate over the sampling process ($\tau = 0.01$, initial guidance scale 1): single trajectories (black) and their mean.

All results are shown for the uniform model. The masking model shows qualitatively similar behavior.

E ADDITIONAL DETAILS REGARDING TEXT EVALUATION

In this appendix additional details and results on the text modality are provided. First, Appendix E.1 describes the experimental setup for the text modality. Second, Appendix E.2 analyzes the posterior estimation in more detail. Finally, Appendix E.3 contains an in depth discussion of the results obtained using the NOT gate.

E.1 EXPERIMENTAL SETUP DETAILS FOR TEXT EVALUATION

In our analysis on the text modality, we use the Cognac (Chen et al., 2022) dataset and benchmark. For the posterior evaluation, we use the dataset as a set of prompts to verify whether the answer is in a certain class. For the NOT gate evaluation, we use the benchmark, where the (guided) model needs to generate text about one topic (e.g., beverages), but it cannot generate anything related to a subcategory (e.g., wine), the class. An example of a prompt in this dataset is given in Figure 19. There is an instruction to generate about a specific topic, some examples and finally some formatting instructions. We extended the formatting instructions, as the DLM otherwise failed often to format correctly. Note that this is the prompt unconditional to the class, so it does not mention that class.

> Generate examples that are under the category of beverage.
> == red wine is a type of wine made from dark colored grape varieties. ==
> == pinot noir (french: [pino nwaʁ]) is a red wine grape variety of the species vitis vinifera. ==
> == turkish coffee is a style of coffee prepared in a cezve using very finely ground coffee beans without filtering. ==
> Now keep generating.
> Only output the continuation in == ... == format, no other text.
> Your continuation must be entirely original.

Figure 19: Neutral, unconditional prompt

Answer evaluation. The evaluation uses WordNet (Miller, 1995) to evaluate the generated text. It leads to three possible outcomes: “Class”, “Other classes” and “Invalid”. First, it checks whether the model generated something related to the topic at all, by finding literal matches of WordNet leaf hyponyms (i.e., words that fall under the topic). This method showed some limitations, as literal string matching on the semantically incomplete WordNet had quite some false negatives. To alleviate this issue, we introduce LLM as a judge with the LLM being Qwen3 8B (Yang et al., 2025). We find that the combination of both (so either a WordNet match or LLM as a judge match) works best for a fair evaluation. This is because WordNet catches niche words, while the LLM catches non-literal matches. As an example, we take the dev-set ID 31, with topic snack food and class (subtopic) sandwich. WordNet has only four leaves that would be counted on topic, but not in the class: chip, corn chip, potato skin, tortilla chip. While catching niche snacks such as potato skin, it is clear that the category is broader. A generated sample “== A chocolate bar is a popular snack food, typically made from cocoa and sugar. ==” was counted off topic by WordNet, but on topic by the LLM judge. As such, both methods truly complement each other. To check whether the model not only generated something on topic, but also in the specified class (subtopic), again WordNet is used with string matching. For the class detection, we found much less false negatives and adding LLM as a judge the same way increased the false positives strongly. Hence, we keep the WordNet only approach for class detection. These two checks create three possible outcomes for generated text. The first one is “Class”, which means the text is on topic and the specified class was generated. The next is “Other classes”, where the text is on topic, but not the specified class. The last option is “Invalid”, where the model fails to generate something on topic at all. As the DLMs also showed some failure modes (such as not respecting the == == format), we count degenerate output also under the class of Invalid, no matter the Wordnet matches.

Prompting the model. As DLM, we choose the popular LLaDA 8B model (Nie et al., 2025). This model is prompted in three different ways. The unconditional prompt, as shown in Figure 19, does

not mention the constraint at all. The negative prompt adds an instruction to surely mention the class, as shown in Figure 20. We call this the negative prompt, because for the NOT gate, we will guide away from this prompt and thus negatively guide on it. The positive prompt has the opposite instruction, where it is asked to not mention the subcategory.

We would like to highlight that classical CFG, also in text, intends to strengthen the adherence to the entire prompt, by contrasting with the unconditional prompt filled with only [MASK] tokens. However, for our guidance setup, we argue that the unconditional model is the one that sees the neutral prompt. Just like in image generation, the model already "knows" about the task that it should generate natural images, or like our MNIST model knows it should generate numbers between 0-9, the diffusion model that generates inside the data distribution for our task, is the one that is prompted with a neutral prompt that explains the task at hand. As such, we do not use p_{MASK} as the basis of all models, but rather the neutrally prompted $p_{\text{neutral prompt}}$. As this is our base model, our conditional models also build further on this prompt, only adding to the base prompt.

Posterior evaluation setup. For the posterior evaluation, we combine the three prompts in the main paper. That is: the neutral prompt is used to generate the text, while the posterior of the class is calculated on the negative versus the positive prompt. We use the positive prompt here as it gives an even better posterior signal than other prompt combinations, as shown in Appendix E.2.

For the NOT gate in text, we explore two different guidance setups and two baselines, with all setups being shown in Table 5. The first setup is that of the NOT gate in probability space, as outlined in the theory of Section 3.2. This model only uses the neutral and negative prompt, so it does not use the "power" of the prompted NOT. In the second setup, negative CFG, we naively apply the well known logit-space CFG, but with a negative sign to guide away from the negative prompt. As baselines, we have the unconditional model, which is unaware of any constraints. Any samples that are already correct under this model, were at no risk of violation in the first place, As such the unconditional model sets a worst case lower bound. The second baseline model is the positive prompted model. This model is interesting to compare with, as it could be argued that this captures a very rough upper bound of how much the model "knows" not to generate.

NOT gate (prob-space)	$p = p_{\text{neutral prompt}} - \lambda_t \cdot (p_{\text{neg prompt}} - p_{\text{neutral prompt}})$
Negative CFG (logit-space)	$l = l_{\text{neutral prompt}} - \lambda \cdot (l_{\text{neg prompt}} - l_{\text{neutral prompt}})$
Unconditional	$p = p_{\text{neutral prompt}}$
Positive prompted	$p = p_{\text{pos prompt}}$

Table 5: Contrastive signals used in this paper.

Generate examples that are under the category of beverage.
== red wine is a type of wine made from dark colored grape varieties. ==
== pinot noir (french: [pino nwaʁ]) is a red wine grape variety of the species vitis vinifera. ==
== turkish coffee is a style of coffee prepared in a cezve using very finely ground coffee beans without filtering. ==
Now keep generating and include things in the category of wine.
Only output the continuation in == ... == format, no other text.
Your continuation must be entirely original.

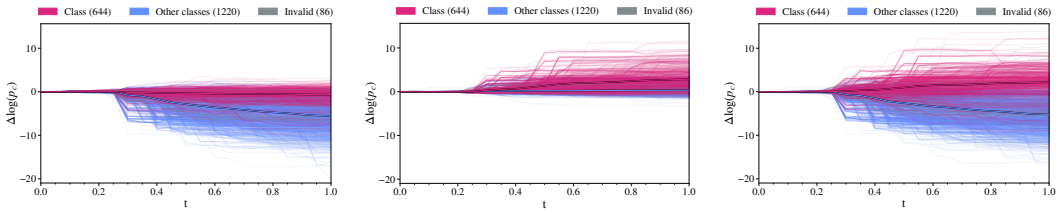
Figure 20: Negative prompt

E.2 POSTERIOR ESTIMATION

In this appendix, the posterior is analyzed deeper in the following ways: we analyze the impact of the prompt types to compare during posterior estimation, we determine classification accuracy in a few-class scenario and we give more examples on high impact tokens for some prompts.

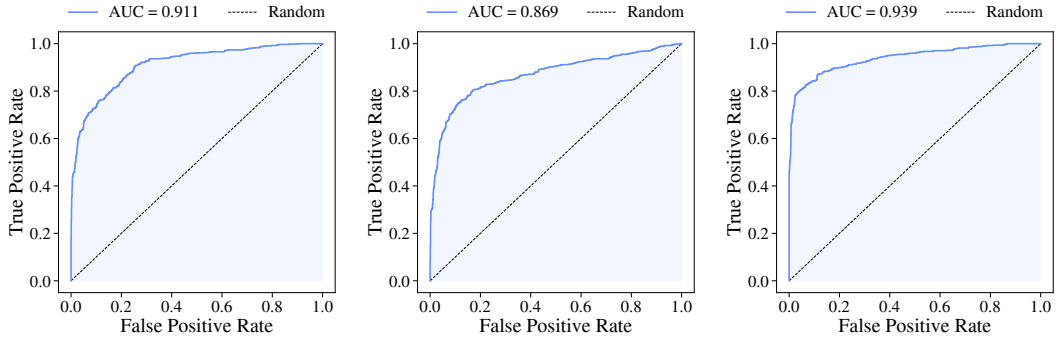
Our posterior approximation can be applied to any two models that give probabilities on the token being generated. This allows for three different couples of models in the case of "NOT" classification: negative vs positive prompt, negative vs neutral and neutral vs positive prompt. The posterior approximations for each of these runs are given in Figure 21. We note some interesting behavior, for example in Figure 21a, the posterior approximation does not move much for violated samples, but it goes down much for correct samples. We interpret it as follows: when the unconditional model was going to violate anyways, adding the negative part of the prompt does not change much. However, if the unconditional prompt wants to generate a correct sample, the negative part of the prompt adds a strong signal. Conversely, we see the opposite in Figure 21b, where the violations increase the posterior approximation and the correct samples stay rather flat. Adding both signals together by directly comparing negative vs positive, we see in Figure 21c, that violations go up and correct samples go down.

Do note in the typical NOT gate context (in other modalities), only the negative vs neutral signal is present and this is also the one used in NOT gate experiments. However, for the sake of finding the best posterior approximation, we find that negative vs positive is best at classifying samples, as shown in Figure 22. In more detail: negative vs positive yields the strongest classifier via the posterior approximation, with an ROC of 0.94. The other approximations perform significantly worse with p=0.05 in a bootstrap test. In conclusion, the best posterior approximation for classification is the negative vs positive setup.



(a) Negative vs neutral (b) Neutral vs positive (c) Negative vs positive

Figure 21: Posterior trajectories with different prompt types.



(a) Negative vs neutral (b) Neutral vs positive (c) Negative vs positive

Figure 22: ROC curves for classification of correct vs violated using different prompt types.

While it is hard in free-text generation to define a fixed set of non-overlapping classes, we attempt to construct such a few-class setup, in order to determine the classification accuracy in such tasks. To this end, we take 1000 samples from the unconditional model per prompt and we select four prompts with reasonable variety (i.e., not the same topic being generated 99% of the time). Then, the majority topics are selected per prompt, and these are defined as the classes. We use the softmax over the posteriors to determine class probability and classify the sample as the highest-probability class. Figure 23 shows a single sample being generated, while Table 6 shows that the posterior estimations yield more than 95% accuracy at the end of generation, while Figure 13b shows the accuracy during generation.

In Table 7 we extend the list of intuitive tokens tokens with high impact on the posterior from the main paper.

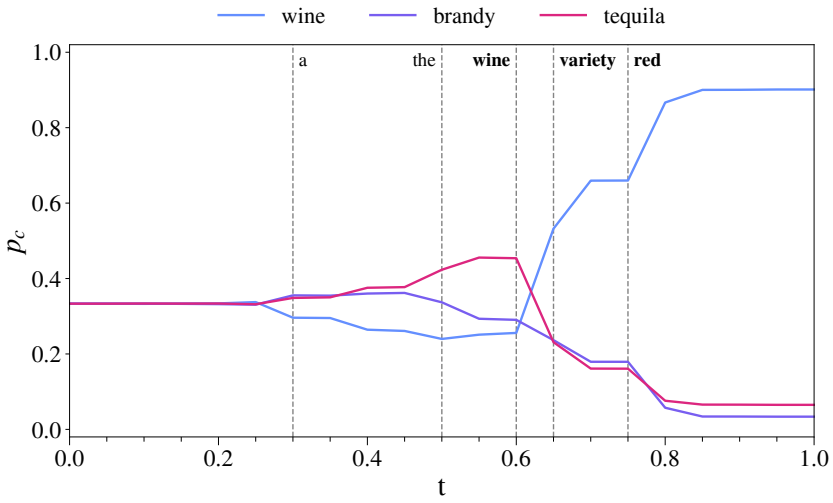


Figure 23: The posterior of multiple classes during generation of the prompt with ID 76 in the development set of Cognac. The answer of the model is "== Cabernet Sauvignon is a type of red wine made from the Cabernet Sauvignon grape variety. ==". Some tokens are highlighted with a dashed line at the time step of generation. The blue line, which represents the wine class, clearly jumps on the words "wine", "variety" and "red" being generated. On the other hand, the posteriors do not move much upon "a" or "the" being generated.

ID	Topic	Classes (%)	Accuracy
76	alcohol	wine 57%, brandy 18%, tequila 26%	99.0%
276	beverage	wine 55%, beer 28%, liqueur 17%	95.1%
353	liquor	gin 65%, brandy 23%, rum 12%	95.2%
418	domestic animal	cat 16%, dog 84%	95.6%

Table 6: Accuracy of the posterior estimation, classifying which (manually defined) subtopic was generated using argmax of the posterior. The IDs refer to the Cognac dataset's development set.

Topic	Sub-topic (class c)		1st Token	1st $\Delta \log p_c$	2nd Token	2nd $\Delta \log p_c$	3rd Token	3rd $\Delta \log p_c$
sweet	candy	+	candy	+2.8	lollipop	+0.8	are	+0.5
		−	marzipan	-2.1	cookies	-0.7	sweet	-0.6
dairy product	cheese	+	cheddar	+2.4	Swiss	+1.4	cheese	+0.5
		−	yogurt	-6.8	cream	-0.5	product	-0.5
dairy product	milk	+	milk	+0.8	with	+0.4	whole	+0.4
		−	whey	-0.7	cheese	-0.7	liquid	-0.6
alcohol	wine	+	wine	+2.9	red	+0.6	Cabernet	+0.5
		−	absinthe	-5.5	tequila	-4.6	liqueur	-4.2
beverage	wine	+	wine	+2.3	cabernet	+1.3	red	+1.1
		−	syrup	-5.5	ginger	-3.9	espresso	-3.2
liquor	gin	+	gin	+1.5	pink liqueur	+0.8	floral liqueur	+0.8
		−	vodka	-5.3	almonds	-4.8	coffee	-3.8
domestic animal	dog	+	dog	+4.0	-sized	+2.0	coat	+0.9
		−	cat	-3.4	coon	-0.5	breed	-0.2

Table 7: Tokens pushing most towards (+) and away from (−) the specified class, as estimated by our posterior estimation. (Sub)words in gray are context to put the token (black) in context for the reader. Each of there gray contexts were the actual context for that token for 100% of the samples with that token.

E.3 Logical operators: NOT gate

The results of the NOT gate in language on the Cognac benchmark are shown in Table 8. Firstly, note that the unconditional model already answers correctly, i.e., any class other than the undesired class, 70% of the times. This means that the model never intended to violate the constraint anyway in many cases. With this in mind, the positive prompted model, i.e., the one prompted not to violate the constraint, scores only 79%. Interestingly, both negative CFG and the NOT gate yield very similar results. So while the NOT gate is more grounded in the theory, we would recommend negative CFG to practitioners, as there are not three hyperparameters (grid search shown in Figure 24) but just one hyperparameter to tune. The results of the NOT gate are also very similar to the positive prompted model. This validates that the not gate can work as well as prompting the model. In the case of language, this is rather silly, as we can just prompt the model, but this comparison serves more as a proof of concept in the one modality where we can actually compare with a prompted model. So these results are complementary to the other modalities, where simple prompting is not available.

Method	Correct (Other classes)	Violated (Class)	Invalid
Unconditional	0.70 ± 0.03	0.23 ± 0.03	0.07 ± 0.02
Prompted "NOT"	0.79 ± 0.03	0.10 ± 0.02	0.11 ± 0.02
Negative CFG (logit-space)	0.81 ± 0.02	0.06 ± 0.01	0.13 ± 0.02
NOT gate (prob-space)	0.79 ± 0.03	0.08 ± 0.02	0.14 ± 0.02

Table 8: Results of the NOT on the Cognac benchmark

F Additional details regarding antibody sequence generation

In this section additional details and results on antibody sequence generation are provided. First, Appendix F.1 introduces the task and the dataset for readers without a background in biology. Second, Appendix F.2 describes the architecture and the training hyperparameters. Third, Appendix F.3 presents ablations on the model performance and introduces the metrics used. Finally, Appendix F.5 covers the posterior estimation and the results obtained with the NOT gate. The structure deliberately

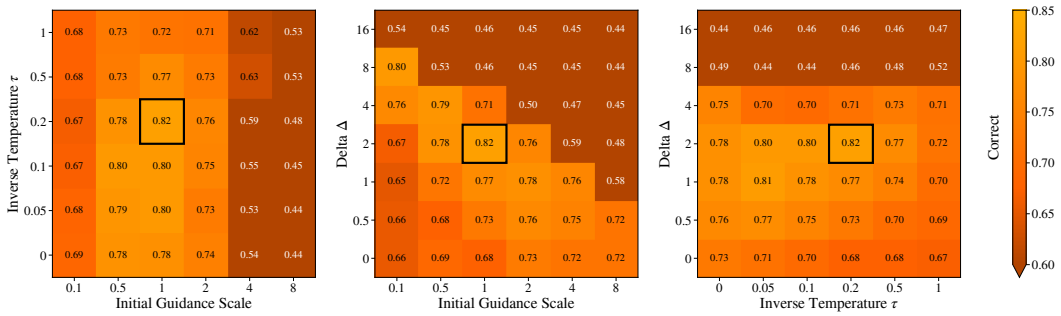


Figure 24: Grid search on the development set of the Cognac benchmark. The optimal hyperparameters that were used for the final experiment are indicated with a bold box. We found that the text modality needs the δ from Equation (10). We use Δ to indicate the cumulative δ over all steps. The first heatmap shows that τ is a little less sensitive to tune than the initial guidance scale. The second heatmap shows a downwards diagonal ridge, which seems to indicate that high initial guidance combined with high delta can make the guidance scale explode and yield poor results. The third heatmap again shows less sensitivity for τ, and larger sensitivity for Δ.

mirrors that of the appendix on images (Sections D.1 and D.2), which precedes this one, so that a reader familiar with it can focus on the differences, mainly the metrics used to evaluate the generated sequences.

F.1 Dataset and background

This work is aimed at a broad machine learning audience, which might not be familiar with proteins, let alone antibodies. In this section we therefore give the reader the minimum needed to understand the specificities of the task. Interested readers are referred to the protein language model literature, for instance Lin et al. (2023); Fournier et al. (2024), and to Shuai et al. (2023) for antibodies in particular.

Antibodies. Antibodies are Y-shaped proteins that are a key part of our immune system. Like every protein, an antibody is defined by its sequence of amino acids, of which 20 are commonly used and each is denoted by a letter. An antibody consists of two identical heavy chains and two identical light chains. At the tip of each chain sits a variable domain of roughly 120 residues, which recognises the target of the antibody, the antigen. This domain is built from four framework regions (FR1-4), which form a conserved structural scaffold, interleaved with three complementarity-determining regions (CDR1-3). The CDRs are hypervariable loops that make direct contact with the antigen and therefore determine what the antibody binds to. The CDR3 of the heavy chain (CDRH3) is the most diverse of them. During an immune response, the CDRs are fine-tuned towards the antigen in a process of somatic hypermutation and selection called affinity maturation.

V genes. The variable domain of the heavy chain is encoded by recombining a V, a D and a J gene segment. The V gene encodes the largest part of the domain, namely the framework regions FR1-3 and CDR1-2, whereas CDR3 lies at the junction of the three segments and is therefore the most variable. The human heavy-chain V genes are commonly grouped into seven families, IGHV1 to IGHV7, whose members are highly similar in sequence. As the germline-encoded parts of a sequence are only mildly altered by mutations, the family of an antibody can be determined very reliably by aligning it to a database of germline genes. This makes the V-gene family a convenient class label for a conditional generative model: it is discrete and it can be assigned to every sequence. It should also be noted that the distribution of V-genes in humans is highly unbalanced, with IGHV3 accounting for 44% of human antibodies, whereas IGHV6 and IGHV7 make up only about 0.6% and 0.5%, respectively (see Table 9).

Dataset. The Observed Antibody Space (OAS) (Olsen et al., 2022) is the largest and most widely used database of antibody sequences. As our goal is not to obtain a state-of-the-art antibody model, but to demonstrate the flexibility of our approach, we use an older partition of the database, namely

the OAS95 set of 559M sequences clustered at 95% identity, that was used to train IgLM (Shuai et al., 2023). IgLM is an autoregressive antibody language model, which we use to evaluate whether the generated sequences look reasonable (Appendix F.3). To verify our theoretical claims, however, a conditional dataset is required. We therefore keep the human heavy chains of OAS95 that are complete variable domains (starting with the typical first residues of FR1 and without ambiguous residues), and classify them by V-gene family using the publicly available IgBLAST tool (Ye et al., 2013) with the IMGT database of human germline genes (**?**). Exact duplicates are removed. This leaves 55.4M sequences, of which 2% are held out for validation. The sequences are 122 residues long on average, and 99% of them are between 113 and 132 residues long. As the family is mainly determined by the germline-encoded regions, this classification is essentially unambiguous.

V-gene family	IGHV1	IGHV2	IGHV3	IGHV4	IGHV5	IGHV6	IGHV7
Share of sequences (%)	24.9	2.5	44.1	25.3	2.2	0.6	0.5

Table 9: Class balance of the antibody dataset (human heavy chains of OAS95, measured on the held-out validation split of 1.1M sequences). The conditional model is trained on these seven classes.

F.2 Training details

Architecture. The antibody models are based on the AMPLIFY protein language model (Fournier et al., 2024), a bidirectional Transformer encoder with rotary position embeddings and SwiGLU feed-forward layers. As for images, we add adaptive layer normalization (AdaLN-Zero) as conditioning mechanism (Peebles & Xie, 2022): the noise level t and the V-gene family are embedded and summed into a single conditioning vector that modulates every block, where the label embedding has an additional entry for the unconditional case. The vocabulary has 27 tokens: 21 residue tokens (the 20 amino acids and the ambiguity code B) and 6 special tokens, of which one is the mask token used for masking diffusion. Every sequence is enclosed by a begin and an end token and padded to a fixed length of 256 tokens, well above the longest sequences. These special tokens are never corrupted and are excluded from the loss. As a consequence, the model does not have to determine the length of a sequence itself: when sampling, the padding pattern of a randomly chosen real training sequence is used as starting point, so that the lengths follow the empirical length distribution independently of the class. The resulting models have 50M parameters, of which 33.6M belong to the AMPLIFY backbone and 16.6M to the AdaLN modulation layers.

Training. As for images, the model predicts the clean sequence with a cross-entropy loss, which for masking is computed on the masked residues only and for uniform noise on all residues. We use a linear noise schedule with stratified sampling of the noise level. Conditional models are trained with label dropout, so that the same network also acts as unconditional model when the label is replaced by the unconditional token. In contrast to the images, no separately trained unconditional model is used, and no exponential moving average of the weights is kept. The hyperparameters in Table 10 were not tuned per noise type. Apart from the noise type, the masking and uniform models only differ in their number of training steps, 200k and 400k, which corresponds to roughly 0.9 and 1.9 epochs of the 54.3M training sequences.

Architecture	AMPLIFY with AdaLN-Zero
Parameters	50M
Depth / hidden size / heads	12 / 480 / 20
Sequence length / vocabulary	256 / 27 (incl. mask token)
Optimizer	AdamW
Learning rate	$1 \cdot 10^{-4}$
Betas / weight decay	(0.9, 0.95) / 0.01
Warmup steps / schedule	1000 / cosine decay
Batch size	256 (2 GPUs $\times$ 128)
Training steps	200k (masking) / 400k (uniform)
Gradient clipping	1.0
Numerical precision	bf16
EMA	none
Label dropout	0.15
Noise schedule	linear
Noise level sampling	stratified in $[0, 1]$
Loss weighting	none

Table 10: Hyperparameters of the antibody models.

Training curves. Training progress is shown per noise level in Figure 25. As expected, lower noise levels (closer to $t = 1$) show a lower loss, particularly with uniform noise. This is beause when training a masked model the loss is only computed on the still masked tokes, while for the uniform model the loss is always computed on all tokens. In the latter, most tokens at low noise levels are already correct and should simply be returned back as output, a luxury the masking model can never have. This distinction also renders the comparison of loss curves between masking and uniform cases irrelevant. An interesting note in this setting is that the loss decreases very rapidly in the beginning and then only progresses very slowly. We hypothesize that this is due to the dual nature of antibody sequences, very easy framework regions, which are learnt fast and then highly unpredictable CDRs on which progress is slow. For the V-gene conditional task discussed in this paper the first part is most relevant, which is why our models are never evaluated on CDRs.

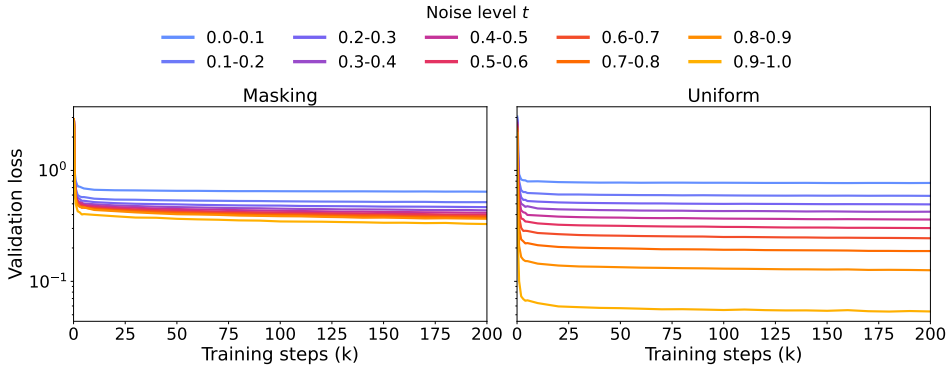


Figure 25: Validation loss during training of the antibody models, averaged over bins of noise levels $t \in [0, 0.1], [0.1, 0.2], \ldots$, for masking (left) and uniform noise (right).

F.3 Model ablations

Metrics. The image metrics (FID, precision, recall and C2ST) rely on features of a pretrained image classifier. For antibodies we replace them by their protein counterparts, and add a measure of how natural the sequences are. All metrics are computed on 4096 generated sequences, whose class labels are drawn from the empirical class frequencies of the training set (Table 9), such that rare families remain rare, and whose lengths are set as explained above.

- **IgLM log-likelihood.** IgLM (Shuai et al., 2023) is an autoregressive language model trained on natural antibodies, conditioned on the chain type and the species (here heavy and human). We report the average log-likelihood per residue of the generated sequences under IgLM, which measures how antibody-like they are (higher is better). For reference, real training sequences reach -0.57 and random amino acid sequences of the same lengths -3.80. Figure 26 shows the gap between generated and real sequences, which should go to zero.
- **FPD.** The Fréchet Protein Distance is the protein counterpart of the FID: the Fréchet distance between Gaussian fits to the ESM-2 embeddings (Lin et al., 2023) of generated and real sequences. It measures quality and diversity together (lower is better). We compare to 20000 random training sequences.
- **C2ST.** The classifier two-sample test is the held-out accuracy of a logistic regression that separates real from generated embeddings (4096 of each, 80/20 split), where 0.5 is the best possible value, as then generated samples are indistinguishable from real ones.

Additionally, we report the conditioning accuracy: the fraction of generated sequences that IgBLAST, the same tool that produced the training labels, assigns to the requested family. This is the counterpart of the classifier-based accuracy used for images.

These metrics are computed for both noise types as a function of the number of sampling steps. This is shown in Figure 26 where the unconditional model is compared to a conditional model with class labels from the true data priors. As expected the quality of all samples increases as the number of inference steps increases. At low number of sampling steps the conditional model outperforms the unconditional one, which can be attributed to the less diverse distribution the model is sampling from. In essence the conditioning signal is often a nearly deterministic signal for portions of the sequence, by using a conditional model, these sections will never be confused, which is not true for the unconditional model, particularly if multiple tokens are flipped simultaneously. Another interesting note is that all metrics seem to saturate at 32 steps, again indicating the strong determinism of the task at hand. In this regime, at every step around 4 tokens are flipped per step and still this is sufficient for perfect samples.

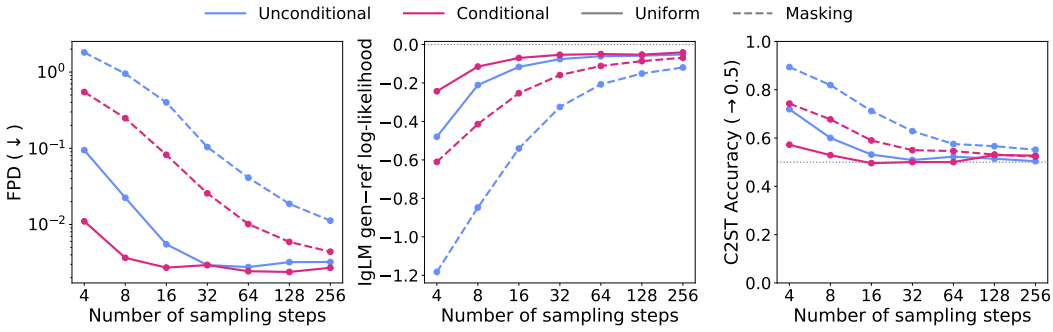


Figure 26: Sweep over the number of sampling steps for the uniform-noise antibody model, for the unconditional (label dropped) and the conditional model, with a linear sampling schedule and 4096 sequences per point. The uniform model in full and the masking model in dashed lines. Shown are, from left to right, the FPD, the gap between the IgLM log-likelihood of the generated and of real sequences, and the C2ST accuracy. Class labels for the conditional model are drawn from the empirical class frequencies.

All results are shown for the uniform-noise model, sampled with the linear schedule that was also used for training.

F.4 Posterior estimation

In this context the V-gene classes are so well separated that the online posterior's task is relatively easy, a few token flips is usually enough to determine the class fully. This is highlighted by the

perfect accuracy and AUC scores the method obtains, not only at the end point, but very early on Figure 13. We see this as a significant proof, that when a strong classification signal is available, the proposed online posterior distinguishes it reliably.

F.5 LOGICAL OPERATORS: NOT GATE

Metrics For the NOT gate we use similar metrics to those used to perform the ablations on the model in Appendix F.3. In particular, the IgLM log-likelihood is used as a quality metric to verify sequence quality on an individual level. To measure distributional coverage, the FPD is again used with ESM2 embeddings, this time computed between generated samples and reference data which do not include the undesired class. The idea is that should a sequence of the unwanted class be generated, the FPD measured with respect to that set should increase as well. Further, as always, we measure safety or removal capacity of the method with a simple violation rate. This value is obtained by classifying samples using IgBLAST and computing the fraction of samples from the undesired class. All metrics are computed on 4096 samples.

Methods To highlight the severe class imbalances present in the data, samples are first generated unconditionally. This gives a first worst-case scenario baseline. Then similarly to the image case, a static logit negative guidance version implemented. This is essentially standard discrete CFG with a negative guidance scale, that therefore pushes away from a certain condition instead of reinforcing it. Finally, the dynamic negative guidance approach, i.e. the NOT gate, is ablated. A key point of this method is that a properly scaled prior-weighted initial guidance scale is chosen. In other words, for a single hyperparameter setting a majority class receives more guidance than a minority class.

Challenge Due to the severe class imbalance present in the data it should be intuitively clear that finding one setting that works best for all cases is no easy task. Majority class simply require much more guidance to be removed properly when compared to minority classes. This is particularly true in static approaches, such as standard CFG, where a fixed guidance scale is chosen for all classes regardless of their prevalence. Our dynamic approach and its prior-weighted initial guidance scale does not suffer the same flaw. By design the guidance scale start off at a multiple its theoretical value, i.e., at $r\frac{p(c_i)}{1-p(c_i)}$ with the factor r defined by the value of the initial guidance scale s_0 through $r = (N-1)s_0$, see Appendix G for more details. This means that a single choice of hyperparameter will automatically apply more guidance to majority classes than to minority ones simply because the initial value is prior-weighted. This is an extra quality of our NOT gate. A single hyperparameter setting can be set, and work equally well at removing any class regardless of how imbalanced the dataset really is.

Results As outlined above, finding a single static guidance scale for negative CFG, that is able to remove all classes regardless of their prevalence is impossible. This is shown in Figure 27. On the right the violation rate, decreasing for all classes as the negative guidance scale is increased, is shown. On the left, the FPD is shown, here a very different story emerges. Majority classes follow the usual trade-off, first decreasing and then increasing indicating that guidance is starting to become too strong. For the minority on the other hand every guidance scale value seems to be too strong in terms of FPD. This makes sense: the FPD is a distributional metric and in this static case guidance affects all trajectories just as much, and does not only apply to the targeted trajectories. One cannot hope the method to require the same hyperparameter for minority and majority classes.

On the other hand, our NOT gate with dynamic negative guidance, that has a prior-weighted initial guidance is shown in Figure 28. Here, we again can observe that the violation rate goes down as more guidance is applied, but crucially because guidance is not applied equally across classes, minority classes suffer much less from guidance. We view this prior weighted initial point as a key advantage of our theoretically grounded approach.

Similarly to how the two are compared in images, a Pareto curve comparing safety (violation rate) and diversity/quality (FPD) is shown in Figure 29. To highlight the differences between majority and minority classes the two Pareto fronts are plotted. Similarly to what was obtained on images, the dynamic approach outperforms the static one. Even more so if a single global setting (one fixed set of hyperparameters) is desired.

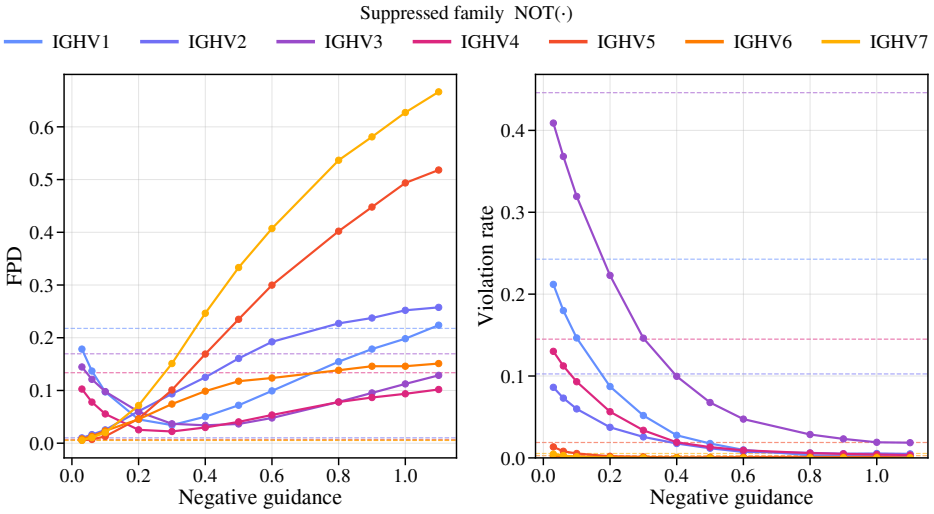


Figure 27: Negative CFG with static negative logit guidance. On the left, the Frechet Protein Distance measuring distributional coverage and sample quality, and on the right, the violation rate. Notice the strong differences between majority (IGHV1-4) and minority classes (IGHV5-7).

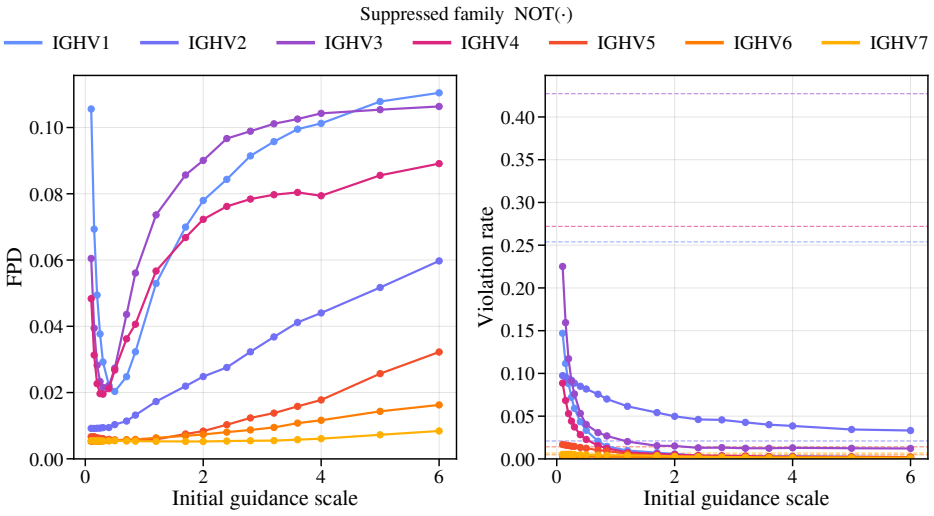


Figure 28: NOT gate with dynamic negative guidance. On the left, the Frechet Protein Distance measuring distributional coverage and sample quality, and on the rate the violation rate. Here, thanks to the prior weighted initial guidance scale, minority classes are much less harmed when applying guidance.

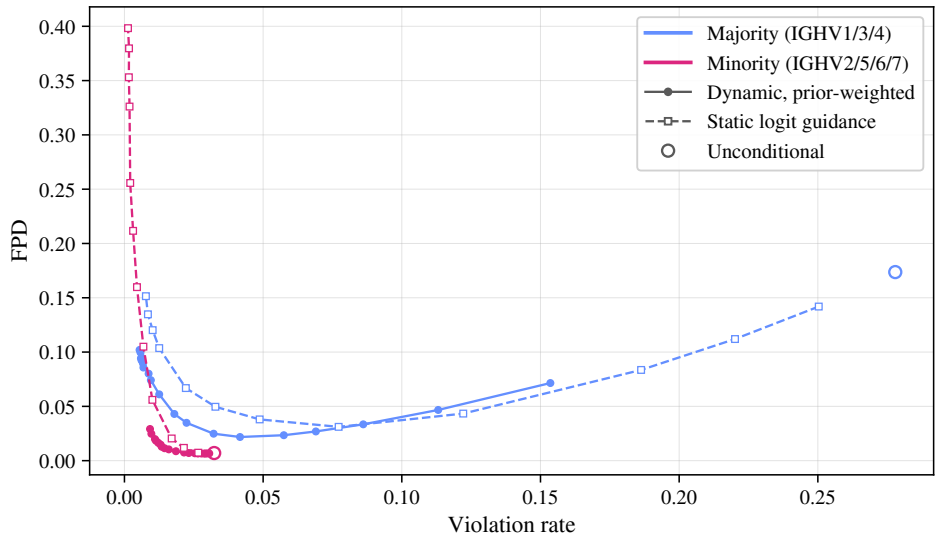


Figure 29: Pareto front between quality (FPD) and safety (violation rate) for majority and minority classes. Negative CFG with static logit guidance is compared to the NOT gate with dynamic negative guidance.

G Posterior estimation hyperparameters

The posterior estimation, when used as formulated by the theory, tends to be stochastic and overconfident, which is why prior work in the continuous domain introduced the two regularizing hyperparameters τ and δ (Koulischer et al., 2025b). We also use an initial guidance scale to determine the strength of guidance with which generation starts.

By setting $\tau < 1$ in Equation (10), strong jumps in the posterior evaluation are avoided. All grid searches across modalities resulted in a value of τ that satisfies this intuition.

The second hyperparameter δ gives a constant push upwards to the posterior. This is most important when interfering during generation with the NOT gate, as there the precise calibration of the posterior value is key in contrast to the AUC/Accuracy experiments that are mainly impacted by the correct ranking of classes. In practice we find that the posterior most often decreases, particularly for the NOT gate, this is shown for text in Figure 21a, even before guidance and Figure 18 after guidance for images. We argue this is because at best the conditional model agrees with the neutral model, yielding an update $\log p(x_t \mid x_{t-1}, c) - \log p(x_t \mid x_{t-1})$ around 0. At worst the models disagree and we pick the token that the unconditional agrees most with, which means $\log p(x_t \mid x_{t-1})$ is relatively high and the difference will be strongly negative. So the posterior going up is a very rare case in practice, though it is clear intuitively that when the conditional model agrees, the posterior is going up. Thus, the δ gives a permanent upward push to actually increase the posterior when the models agree.

The third hyperparameter is the initial guidance scale s_0, that works on the prior probability $p(c \mid x_0)$, which is again important for the NOT gate. The idea of the initial guidance scale is to control the baseline strength of guidance, with three different cases merged in one formula. For this, we scale the priors in guidance scale-space, as if they were uniform: The guidance scale for a uniform prior would start at $\frac{p(c_i)}{1-p(c_i)}$, which is $\frac{1}{N-1}$, and thus we would need a scaling factor r of $s_0 \cdot (N-1)$ in order to achieve initial guidance of s_0. For all three cases, we take the real prior, convert it into guidance scale-space with $\frac{p(c_i)}{1-p(c_i)}$, multiply with r and invert back to probability-space. The first case is that of uniform priors, such as MNIST. In that case, the whole formula just boils down do using exactly s_0 as the strength for all classes. In the second case, we consider the prior to be unknown, for example in the text modality. As the prior is unknown and we want to start with a fixed guidance strength, the approach reduces to the uniform case with a flat guidance scale s_0. The

third case is that of known but non-uniform priors. There, we again convert the prior to the guidance scale-space: $\frac{p(c_i)}{1-p(c_i)}$, we scale it by a factor r and then invert back into probability space. This way we scale all guidance strength equally, and the strength per class is still informed by its prior, rather than using a flat scale. We chose this parametrization as we found it easy to work with and sweep, but there is no fundamental reason to choose this parametrization over another.

H REPRODUCTION DETAILS

We make our code available on https://anonymous.4open.science/r/Online-Class-Posteriors-for-Interpretable-and-Controllable-Discrete-Diffusion-E94D. TEMPORARY NOTE TO REVIEWERS: the code is shared in its current state. Cleanup towards better readability is ongoing, particularly for the image and antibody experiments, and as such the github repository might still change. However, the functionality will remain exactly the same.

All text and image experiments ran on NVIDIA A40 GPUs and 4 virtual cores of a Intel Xeon Silver 4310 CPU. For antibodies the experiments where performed on NVIDIA GeForce RTX 4090 GPUs.