Appendix

Table of Contents

A Extended Background

In this section we provide full descriptions of the diffusion processes introduced in Sec. 3.

A.1 Continuous noise diffusion

The forward process is defined by noise variances β. We use the cosine variance schedule from Nichol and Dhariwal [55]. For convenience we further define

$$\alpha_t = 1 - \beta_t, \;\; \bar{\alpha}_t = \prod_i^t \alpha_i$$

The forward process is defined by the conditional distributions

$$\begin{aligned} p(x_t|x_{t-1}) &= \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \\ p(x_t|x_0) &= \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t) I) \\ p(x_t|w) &= \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}U_\theta w, (1-\bar{\alpha}_t) I) \end{aligned}$$

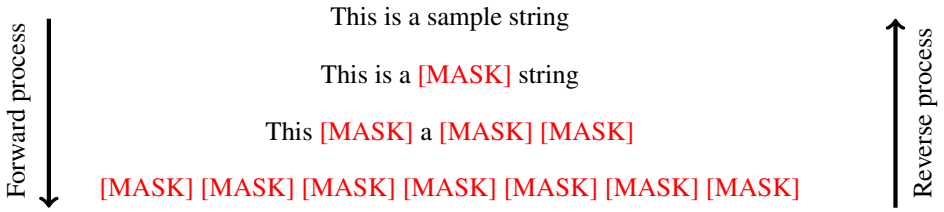

Figure 8: Illustration of a string gradually corrupted by [MASK] tokens.

where U_θ is an embedding matrix. The reverse process is defined by

$$\begin{aligned}
\pi(x) &= \mathcal{N}(0, I) \\
p(x_{t-1}|x_t, x_0) &= \mathcal{N}\left(x_{t-1}; \mu_t, \sigma_t^2 I\right) \\
\mu_t &= \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t \\
\sigma_t^2 &= \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \\
p_\theta(w|x_t) &= \text{Softmax}(\phi_\theta(x_0)) \\
p_\theta(x_{t-1}|x_t) &= \sum_{\hat{w}} p(x_{t-1}|x_t, x_0 = U_\theta \hat{w}) p_\theta(\hat{w}|x_t)
\end{aligned}$$

A.2 Categorical noise diffusion

Following Austin et al. [4] we define the MLM style categorical diffusion using transition matrices

$$[Q_t]_{ij} = \begin{cases} 1 & \text{if } i = j = m \\ \alpha_t & \text{if } j = m, i \neq m \\ 1 - \alpha_t & \text{if } i = j \neq m \end{cases}$$

and $\bar{Q}_t = Q_1 Q_2 ... Q_t$ for noise schedule $\overline{\alpha}_t \in [0, 1]$ (see Figure 8 for an illustration). These transition matrices correspond to categorical conditional distributions

$$\begin{aligned}
p(w_t|w_{t-1}) &= \text{Cat}(w_t; p = w_{t-1} Q_t) \\
p(w_t|w_0) &= \text{Cat}(w_t; p = w_0 \bar{Q}_t)
\end{aligned}$$

The reverse process is defined by

$$\begin{aligned}
\pi(w) &= 1[w = [\text{MASK}]^L] \\
p(w_{t-1}|w_t, w_0) &= \text{Cat}\left(w_{t-1}; p = \frac{w_t Q_t^\top \odot w_0 \bar{Q}_{t-1}^\top}{w_0 \bar{Q}_t w_t^\top}\right) \\
p_\theta(w_0|w_t) &= \text{Softmax}(\phi_\theta(w_t)) \\
p_\theta(w_{t-1}|w_t) &= \sum_{\hat{w}_0} p(w_{t-1}|w_t, \hat{w}_0) p_\theta(\hat{w}_0|w_t)
\end{aligned}$$

B Methodological Details

B.1 Infilling algorithm

We sample infills using the procedure in Algorithm 1. The infill mask P is constructed by setting the index of conserved residue equal to 1, in this case at every residue that is not included in set of CDR regions being infilled. We use the same algorithm to perform the guided infilling in Subsec. 5.2, where it is extended with a guidance Langevin sampling step.

Algorithm 1 Infilling with categorical denoising diffusion model

Inputs: Denoiser $p_\theta(\hat{w}|x_t, t)$, corruption process $p(x_t|x_0)$, infilling mask P, and seed sequence s
Returns: Sample from $\tilde{p}(w) = p_\theta(w|P, s)\exp(f(w))$
$x_T \sim p(x_T)$
$s_T \sim p(s_T|s)$
$x_T \leftarrow (I - P^\top P)x_T + P^\top s_T$
for $t = T, \ldots, 1$ **do**
  $p(x_{t-1}|x_t) \leftarrow \sum_{\hat{w}} p(x_{t-1}|x_t, \hat{w})p_\theta(\hat{w}|x_t, t)$
  $x_t \sim p(x_{t-1}|x_t)$
  $s_t \sim p(s_t|s)$
  $x_t \leftarrow (I - P^\top P)x_t + P^\top s_t$
end
$w \sim p_\theta(w|x_0)$
return w

B.2 Hidden State Langevin Sampling

Design of molecules or images with generative models is often posed as the problem of sampling from a posterior distribution $p(x|a)$ given the unconditional distribution $p(x)$ and attribute model $p(a|x)$. Indeed, reinforcement learning, the design of good actions in an environment, can also be framed as posterior sampling where $p(a|x)$ is the probability that a given state or state-action pair is optimal [46]. Methods that employ posterior sampling of this form are often call "plug-and-play" because $p(a|x)$ and $p(x)$ need not share parameters and therefore users can mix and match different instantiations [54, 17, 34, 26]

The most common way to sample from the posterior $p(x|a) \propto p(a|x)p(x)$ is through Langevin sampling on the unnormalized joint density $\tilde{p}(a, x) = p(a|x)p(x)$, with sampling steps

$$\begin{aligned} x^{i+1} &= x^i + \eta\nabla \log \tilde{p}(a, x) + \sqrt{2\eta}z^i, \qquad z^i \sim \mathcal{N}(0, I) \\ &= x^i + \eta\left(\nabla \log p(a|x) + \nabla \log p(x)\right) + \sqrt{2\eta}z^i, \qquad z^i \sim \mathcal{N}(0, I) \end{aligned}$$

When we work with generative models over continuous random variables that permit a likelihood (e.g. normalizing flows), score function (e.g. diffusions), or energy (e.g. EBMs) $\nabla \log p(x)$ has a natural interpretation and sampling can be performed with essentially vanilla Langevin sampling. In other cases where only a denoising function over continuous variables is available, authors have proposed approximate samplers using an approximation of the score function [54].

When we instead hope to sample from a posterior over discrete random variables constructing an analogy to the score function $\nabla \log p(x)$ is challenging, and prior work adopts a different approach of regularizing the conditional sampling distribution $p(w|a)$ with unconditional sampling $p(w)$ in order to maintain high likelihood [17]. In autoregressive models, $p(w)$ is broken down using the chain rule, $p(w_t|w_{<t})$ and thus the appropriate regularization is

$$\text{KL}(p(w_t|w_{<t}) \;||\; p(w_t|w_{<t}, a)) \tag{6}$$

In our case, the distribution $p(w)$ is factorized by the transition distributions $p(w_t|w_{t-1})$ (or their continuous analogies in token embedding space), and we hope to sample from the perturbed transition

$$\tilde{p}(w_{t-1}|w_t) = p_\theta(w_{t-1}|w_t)\exp(v_\theta(w_t))$$

The correct regularization term in our case is thus

$$\text{KL}(p(w_{t-1}|w_t) \;||\; p(w_{t-1}|w_t, a))$$

To put the pieces together, we first recognize that the denoising model $p_\theta(w_0|w_t)$ can be broken down into an language model head, H_θ, and trunk, T_θ, with

$$\begin{aligned} h_t &= T_\theta(w_t) \\ p_\theta(w_0|w_t) &= H_\theta(w_0|h_t) \end{aligned}$$

We can then perform Langevin sampling on the hidden representations, initializing with h_t, as shown in Algorithm 2. In the experiments above we set $\lambda_3 = 0$, as we saw no noticable benefit from adding additional stochasticity. Importantly, sampling from $p(w_{t-1}|w_t)$ already introduces randomness into the reverse process.

Algorithm 2 Guided diffusion sampler

Inputs: Denoiser $p_\theta(\hat{w}|x_t, t) = [T_\theta, H_\theta]$, value function v_θ, and weights $\lambda_1, \lambda_2, \lambda_3$
Returns: Sample from $\tilde{p}(w) = p(w)\exp(f(w))$
$w_T = [\text{MASK}]^L$
for $t = T, \ldots, 1$ **do**
  $p(w_{t-1}|w_t) \leftarrow \sum_{\hat{w}} p(w_{t-1}|w_t, \hat{w}) p_\theta(\hat{w}|w_t)$
  $h^0 \leftarrow T_\theta(w_t)$
  for $i = 0, \ldots, K-1$ **do**
    $z^i \sim \mathcal{N}(0, I)$
    $p_h \leftarrow \sum_{\hat{w}} p(w_{t-1}|w_t, \hat{w}) H_\theta(\hat{w}|h^i)$
    $h^{i+1} \leftarrow h^i + \lambda_1 \nabla_h v_\theta(h^i) + \lambda_2 \nabla_h \text{KL}(p(w_{t-1}|w_t)||p_h) + \lambda_3 z^i$
  end
  $w_{t-1} \sim H_\theta(h^K)$
end
return w_0

C Infilling / NOS Guidance

All of our diffusion models are train on all paired heavy and light chain sequences from OAS [56] (pOAS) combined with all sequences from SAbDab [25], aligned with ANARCI [24].

C.1 Infilling experiment

For our trained diffusion models, we use Algorithm 1 without guidance, generating P based on the indicated CDRs, using chothia numbering for consistency with DiffAb. For the baselines, we constructed wrapper scripts to convert the chosen CDR ids into each method's native format.

C.2 MCMC comparison

Following Verkuil et al. [78], we construct a Markov chain using uniform random mutations to map a sequence w to a mutated sequence w', using the following Metropolis-Hastings correction:

$$p(\text{accept } w'|w) = \min\left(1, \frac{\exp(-E(w')/T)}{\exp(-E(w)/T)}\right),$$

where $T > 0$ is a temperature hyperparameter. While this method has appealing theoretical properties, obtaining good samples from this Markov chain in practice requires hundreds of thousands of steps of burn-in.

In our experiment (Figure 9), we define the energy, E, by combining sequence level probabilities assigned by IgLM with a beta sheets objective function trained on IgLM's representations. We construct the energy as

$$E(w) = p_{\text{IGLM}}(w) + \lambda v_\theta(w),$$

We tune λ to generate sequences with approximately 40% beta sheets. We also tune the NOS λ parameter (Eq. 4) to produce approximately 40% beta sheets.

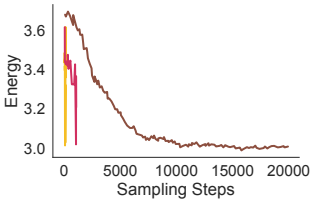

Figure 9: (**left**) Comparing convergence in sampling using a Metropolis Hastings-adjusted MCMC [78] against NOS models. Diffusion models (ours) accelerate sampling by two orders of magnitude while converging to similar energy values.

C.3 PPLM details

In order to generate full (heavy and light chain) optimized antibodies with PPLM and IgLM, we train two separate value function models on IgLM's aggregated hidden representations, one for heavy chain sequences and one for light chain sequences. IgLM uses special tokens for both the chain identity and the species identity of each sequences, and we pass in appropriate corresponding tokens when calculating the hidden representations for each model. To determine the correct species token for each sequence, we use the predicted species returned by ANARCI [24]. Our value function is a

simple one-layer feed-forward neural network trained on top of the mean-aggregated representations for the corresponding chain identity.

To sample using PPLM, we overwrite the forward pass of the huggingface decoder used by IgLM to include a Langevin sampling step over the current hidden representations. We perform K gradient steps to update the current hidden representation $\mathbf{h}'$ by descending on the objective

$$\lambda \mathrm{KL}[p(\hat{w}|\mathbf{h}') \;||\; p(\hat{w}|\mathbf{h})] - v(\mathbf{h}')$$

where h is the original hidden representation output by the model's encoder, and η and λ are the step size and regularization strength respectively. We ran optimization with both vanilla gradient descent and AdaGrad [23] and found AdaGrad to be more robust to poor specifications of the step size. For the results in Sec. 5, we draw samples and present results for all of the hyperparameter settings in Table 1

λ	0, 0.001, 0.01, 0.1, 1.0
η	0.5, 0.8, 1.1, 1.4, 1.7, 2.
K	5, 10
optimizer	SGD, AdaGrad

Table 1: Hyperparameter settings used for PPLM. λ controls the strength of the regularization. Large values prevent sampling values that differ significantly from the unguided model. η controls the size of steps taken in the latent space. Larger step sizes, when not too large, can increase the distance traveled in the latent space and the extent to which sampling can yield samples with high values of the objective.

One critical difference between controllable autoregressive models and controllable diffusions is the ability to resample previously sampled values. Procedures that allow for resampling are often called "iterative refinement" procedures because they can produce increasingly plausible generations by refining the model's previous output at each step in an iterative procedure. Because there are many potential differences between our NOS models and PPLM, including but not limited to the nature of iterative refinement, we performed an additional experiment to assess the impact of adapting a discrete diffusion to perform autoregressive sampling. Autoregressive models can themselves be thought of as diffusions with an idiosyncratic corruption process that masks out all tokens to the right of the last sampled token. As in our discrete corruption process, the prior is also a sequence of all mask tokens. Using this insight, we can run our trained discrete diffusions in autoregressive mode by contriving the sampling noise schedule to be autoregressive and recover an approximation of the timestep post-hoc from the percentage of masks at each step in autoregressive sampling.

Figure 10 shows the difference in objective values and likelihood for samples obtained by running the model in typical diffusion mode (iterative refinement) or in contrived autoregressive mode. We can see that on the beta sheets objective, iterative refinement has a noticeable positive impact on the objective values of the sample. This effect is also present in the SASA objective, but to a much more limited extent. We speculate that the iterative refinement facet of NOS is helpful for outperforming other methods but not completely sufficient.

C.4 Model Architecture and Training

The gaussian and categorical diffusions are trained with the bert-small transformer backbone introduced by Bhargava et al. [8]. We use a cosine noise schedule for both diffusions and train for 100 epochs with a batch size of 64, optimizing with AdamW using an initial learning rate of 5e-3 with a linear warmup. The value function is a feed-forward neural network with one hidden layer. The value function is trained jointly with the denoiser by alternating optimization steps, with 5 steps on the generative objective for each step on the discriminative objective. We train the models for 100 epochs in total.

C.5 Hyperparameter settings

For each guided sampling experiment with NOS, we sample using many different hyperparameter combinations in order to generate both conservative and aggressive optimization of the value function.

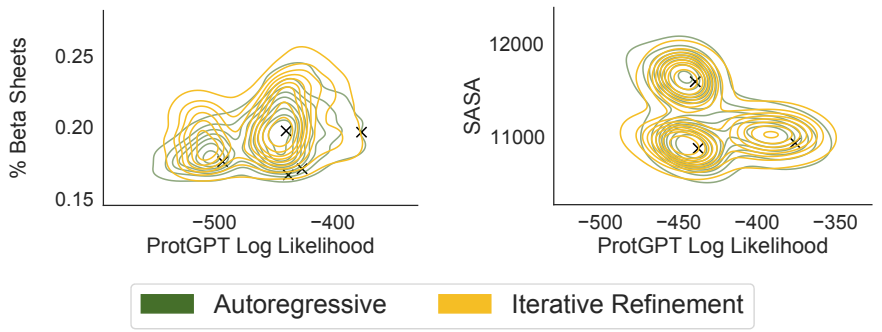

Figure 10: We compare samples from running our guided discrete diffusion (NOS-D) with diffusion style sampling versus autoregressive style sampling. We find that using an iterative refinement procedure does lead to consistent improvements in the objective value, though not to an extent that would suggest iterative refinement is sufficient for strong sampling performance.

The full hyperparameter settings for both objectives (beta sheets and SASA) and both corruption types (NOS-D and NOS-C) are shown in Table 2. In Table 2, there is an additional hyperparameter, "guidance layer", which we did not discuss at length in the main text of the paper. This parameter dictates whether we perform guidance in the first layer of the neural network (the token embeddings), as is standard in continuous diffusion models for discrete sequences, or the final layer of the neural network (the layer before the final linear head). In either case, we can use the same gradient descent objective and corruption process in each case and need only change the variable we propagate gradient updates to. Table 2 shows the hyperparameters used in the just Figure 5.

To aid intuition for the effects of each hyperparameter, we show the sample densities that result from each combination of λ and η in Table 2 when guiding in the first (Figure 11) and last (Figure 12) layer of the NOS-D and NOS-C models. We see that the most important parameter is λ, which controls how far samples tend to move from the seeds. We can also observe that guiding in the first hidden state tends to perform better when sampling with NOS-C, while guiding in the final hidden state tends to perform better with NOS-D.

DiGress comparison DiGress [79] is built on top of a model with one-hot encodings and discrete corruptions. The guided sampling procedure can be described as follows (using the notation from our submission): At each denoising step t, we use the one-hot encodings as a continuous variable and construct a perturbation distribution from a learned discriminative model $\hat{v} = v_\theta(w_t)$,

$$p_\theta(\hat{v}|w_{t-1}) \propto \exp(-\lambda \langle \nabla_{w_t} v_\theta(w_t), w_{t-1} \rangle)$$

We then sample the next value from the base diffusion transition $p_\theta(w_{t-1}|w_t)$ perturbed with $p_\theta(\hat{v}|w_{t-1})$,

$$w_{t-1} \sim p_\theta(w_{t-1}|w_t) p_\theta(\hat{v}|w_{t-1})$$

The key details for guided sampling can be found in the DiGress code repo, where we see that the guided distribution is the normalized product of the original denoising distribution and the softmax of the gradients scaled with λ.

On a theoretical level, this guidance has noticeably different properties from NOS. For large λ, the perturbation $p(w_t|\hat{v})$ collapses to a one-hot on the token index with the largest gradient value. For small values of λ, $p(w_t|\hat{v})$ becomes a uniform distribution. Therefore λ interpolates $p(w_{t-1}|w_t)$ between the original unguided distribution and a one-hot in the max gradient direction. NOS also reduces to unguided infilling when $\lambda = 0$, but $\lambda > 0$ only modulates the direction of the gradient update. The distance between the guided and unguided distribution is controlled by the number of langevin steps and the step size hyperparameter η. Digress amounts to a single update step applied directly to the output token probabilities using a continuous relaxation of the one-hot encoded input, whereas NOS performs a sequence of local updates to hidden states that are actually continuous.

In our comparison, the embeddings and corruptions of each model are chosen to be:

1. NOS-C [Gaussian corruptions + learned embeddings]

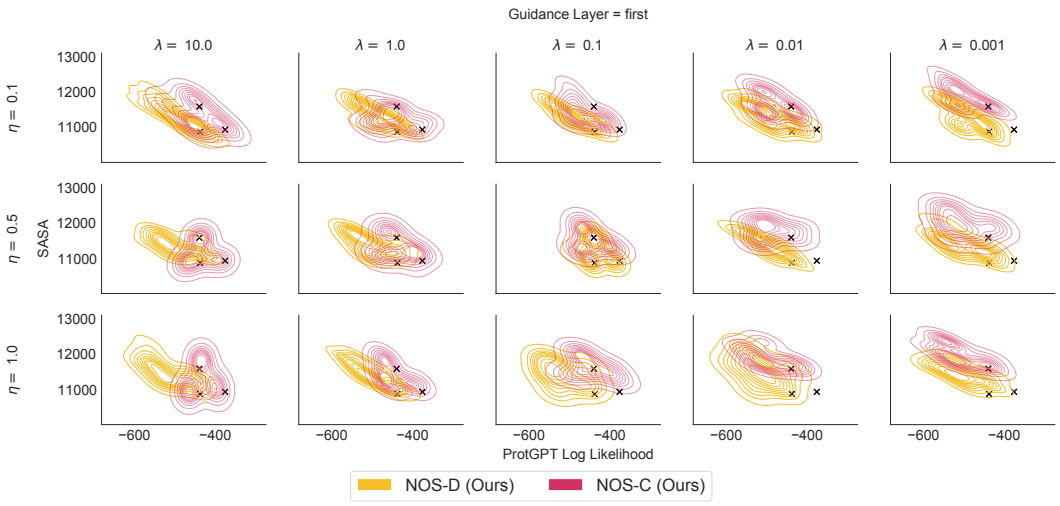

Figure 11: Density plots for every combination of the regularization (λ) and step-size (η) parameter, when performing guidance in the first layer (token embeddings) of the neural network denoiser. We observe that lambda has the strongest effect on trading off fitness under the objective with likelihood or closeness to the seed sequences.

2. NOS-D [Discrete (mask) corruptions + learned embeddings]
3. DiGress [Discrete (mask) corruptions + fixed one-hot encodings]

All models use the same backbone transformer and regression heads, facilitating an apples-to-apples comparison. For DiGress, we perform sampling for large range of scaling values $\lambda \in \{1e5, 3e4, 1e4, 3e3, 1e3, 3e2, 1e2, 3e1, 1e1, 1e0, 1e-1, 1e-2, 1e-3\}$. For each model, λ modulates the degree to which the model prefers greedy sampling from the value function gradient.

λ	0.001, 0.01, 0.1, 1.0, 10.0
η	0.1, 0.5, 1.0
K	5, 10
guidance layer	first, last
optimizer	SGD, AdaGrad

Table 2: NOS guided sampling hyperparameter settings. λ controls the regularization strength, constraining the plausibility of samples, η, when chosen effectively, can effect the degree of optimization that takes place on the hidden states. The guidance layer is the layer in the neural network over which guidance is applied, the first being the token embeddings and the last being the final representations before the linear head. The same values are used for both NOS-D and NOS-C.

λ	0, 0.001, 0.01, 0.1, 1.0, 10.0
η	1.0
K	10
optimizer	AdaGrad

Table 3: Hyperparameter settings used in Sec. 5. The guidance layer for NOS-D is final, and the guidance layer for NOS-C is last.

C.6 Density plots

Because pareto fronts present only a partial view of sampling outcomes (focusing on the best case outcomes along each axis), we also include sample density plots to confirm that our methods consistently yield samples with better trade-off between likelihood and fitness. Figure 13 shows

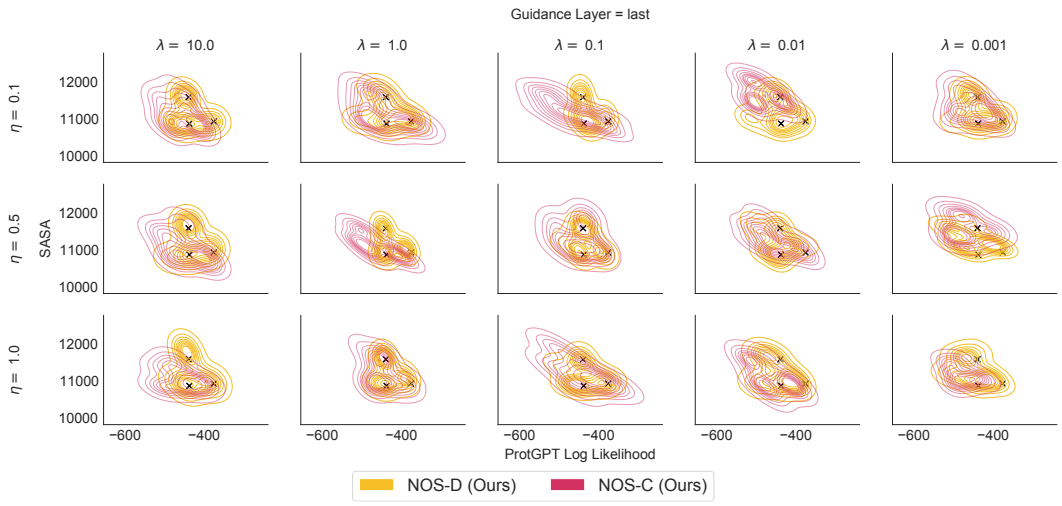

Figure 12: Density plots for every combination of the regularization (λ) and step-size (η) parameter, when performing guidance in the last layer (pre-logits layer) of the neural network denoiser. NOS-C and NOS-D exhibit quite different performance as a function of guiding the first or final hidden representation.

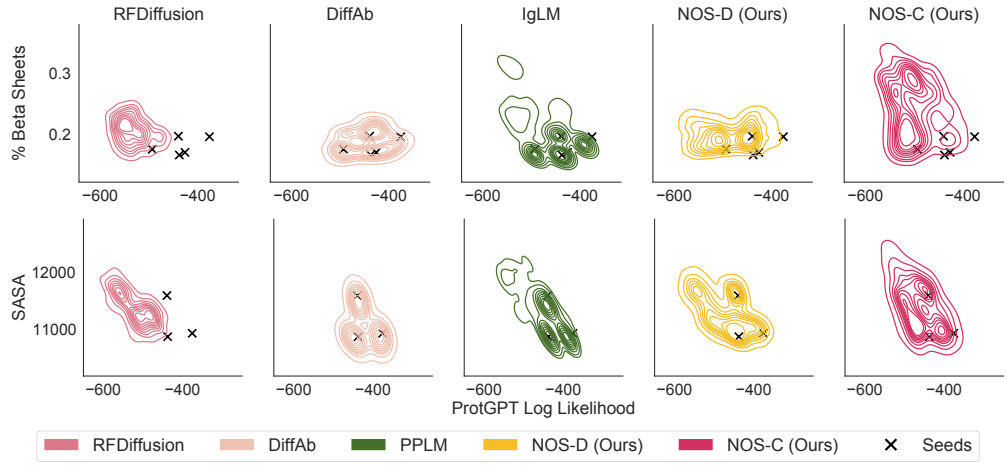

Figure 13: We compare sample densities for the methods presenting in Sec. 5, in order to augment the limitations of simply showing pareto fronts. We see that NOS-C and NOS-D can both consistently generate samples with favorable trade-offs while other methods tend to radically decrease likelihood with little benefit to the value function or be relatively limited to the neighborhood around the seed sequences.

density plots for NOS and baselines when optimizing each of the two objectives (percentage of beta sheets and SASA). We find that DiffAb and IgLM samples tend to cluster around the starting seeds, while RFDiffusion samples tend to generate more diverse samples under the objective, but often with much lower likelihood than the seed sequences. By contrast, both NOS methods consistently improve values of the objective without sacrificing likelihoods.

D LaMBO-2

D.1 Intro to Multi-Objective Bayesian Optimization

When there are multiple objectives of interest, a single best (i.e. strictly dominant) sequence $\mathbf{x}^*$ may not exist. Suppose there are k objectives, $f : \mathcal{X} \to \mathbb{R}^k$. The goal of multi-objective optimization

Algorithm 3 LaMBO-2: one guided discrete diffusion step

Inputs: Seed sequence w_0, edit budget projection P, diffusion timestep t, corruption function $c(w, t)$, constraint function $u(w)$, encoder $g_\theta(w)$, value function $v_\theta(h)$, decoder $d_\theta(h)$, regularization strength λ, SGLD step-size η and temperature τ.
Returns: Best feasible sample from SGLD chain with distribution $p'(\mathbf{x}) \propto p(\mathbf{x}) \exp(f \circ g(\mathbf{x}))$

$w^*, v^* = w_0, v_\theta \circ g(w_0)$ (initialize optimal solution)
$w'_0 = c(w_0, t)$ (apply diffusion noise)
$h'_0 = g_\theta(w'_0)$ (initialize hidden state)
for $i = 1, \ldots, I$ **do**
  $\texttt{loss} = \lambda \mathrm{KL}[d_\theta(h'_{i-1}) || d_\theta(h'_0)] - (1 - \lambda) v_\theta(h'_{i-1})$
  $h'_i = h'_{i-1} - P(\eta \nabla_{h'} \texttt{loss} + \sqrt{2\eta\tau}\varepsilon), \quad \varepsilon \sim \mathcal{N}(\mathbf{0}, I)$ (projected SGLD step)
  $w_i \sim d_\theta(h'_i)$ (decode hidden state)
  if $v^* < v_\theta \circ g_\theta(w_i)$ & $u(w_i)$ **then**
    $w^* \leftarrow w_i$
    $v^* \leftarrow v_\theta \circ g_\theta(w_i)$
  end
end
return w^*, v^*

(MOO) is to identify the set of *Pareto-optimal* (i.e. non-dominated) solutions such that improving one objective within the set leads to worsening another. We say that $\mathbf{x}$ dominates $\mathbf{x}'$, or $f(\mathbf{x}) \succ f(\mathbf{x}')$, if $f_j(\mathbf{x}) \geqslant f_j(\mathbf{x}')$ for all $j \in \{1, \ldots, m\}$ and $f_j(\mathbf{x}) > f_j(\mathbf{x}')$ for some j. The set of non-dominated solutions $\mathscr{X}^*$ is defined in terms of the Pareto frontier (PF) $\mathcal{P}^*$,

$$\mathscr{X}^\star = \{\mathbf{x} : f(\mathbf{x}) \in \mathcal{P}^\star\}, \quad \text{where } \mathcal{P}^\star = \{f(\mathbf{x}) \,:\, \mathbf{x} \in \mathcal{X},\ \nexists\, \mathbf{x}' \in \mathcal{X} \text{ s.t. } f(\mathbf{x}') \succ f(\mathbf{x})\}. \tag{7}$$

MOO algorithms typically aim to identify a finite approximation to $\mathscr{X}^\star$ (which may be infinitely large), within a reasonable number of iterations. One way to measure the quality of an approximate PF $\mathcal{P}$ is to compute the hypervolume $\mathrm{HV}(\mathcal{P}|\mathbf{r}_{\text{ref}})$ of the polytope bounded by $\mathcal{P} \cup \{\mathbf{r}_{\text{ref}}\}$, where $\mathbf{r}_{\text{ref}} \in \mathbb{R}^m$ is a user-specified *reference point*.

$$u_{\text{EHVI}}(\mathbf{x}, f, \mathcal{D}) = \mathrm{HVI}(\mathcal{P}', \mathcal{P}|\mathbf{r}_{\text{ref}}) = [\mathrm{HV}(\mathcal{P}'|\mathbf{r}_{\text{ref}}) - \mathrm{HV}(\mathcal{P}|\mathbf{r}_{\text{ref}})]_+, \tag{8}$$

where $\mathcal{P}' = \mathcal{P} \cup \{\hat{f}(\mathbf{x})\}$ [27, 28, 18]. To decide where to query f next, we search for $\mathrm{argmax}_{\mathbf{x}} \mathbb{E}[u_{\text{EHVI}}(\mathbf{x}, f, \mathcal{D})]$, where the expectation is w.r.t. $p(f|\mathcal{D})$.

D.2 Discrete EHVI

Although expression yield and binding affinity are both continuous measurements, we chose to discretize them and model them as classification with a softmax likelihood (See Appendix D.4). As a result we needed an extension of EHVI for discrete outcomes. Informally, EHVI is simply computing the HVI for different realizations of f and marginalizing f using $p(f|\mathcal{D})$. Instead of taking f to be the latent function of some regression $y = f(w) + \varepsilon$. $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, we instead take f to be the logits of a categorical distribution, $p(y = i|w, \mathcal{D}) = \int \texttt{softmax}_i(f(w)) p(f|\mathcal{D}) df$.

Let $\mathbf{y} = [y_1 \ \cdots \ y_k]^\top$. Given a set of baseline points $\mathcal{B} \subset \mathcal{A}^L$ we define $\mathcal{P}$ (Eq. 8) using the posterior mean $\hat{\mathbf{y}}(w) = \mathbb{E}[\mathbf{y}|w, \mathcal{D}]$, $w \in \mathcal{B}$. We model $y_1, \ldots, y_k$ as conditionally independent given some shared hidden state $h = g_d(w)$, so $p(\mathbf{y}|h, \mathcal{D})$ factorizes nicely. Finally we define $\mathcal{P}' = \mathcal{P} \cup \{\mathbf{y}\}$ and take the expectation of Eq. 8 w.r.t. $p(\mathbf{y}|h, \mathcal{D})$. Since $p(\mathbf{y}|h, \mathcal{D})$ is discrete and factorizes, we can marginalize in closed form when $K_1 \times \cdots \times K_k$ is not too large, where K_i is the number of classes corresponding to the discretization of the original continuous f_i.

D.3 Architecture and Hyperparameters

The inputs of the LaMBO-2 model for antibody design are the variable heavy (VH) and variable light (VL) regions of the antibody sequence as determined by Aho alignment with ANARCI, as well as the (unaligned) antigen sequence. Note that the concatenation of the antigen to the input makes the samples from the generative head conditional on the antigen as well as the unmasked portion of the antibody sequence. The LaMBO-2 model jointly predicts antigen-conditional categorical

token distributions for corrupted positions and discriminative distributions over protein properties. Discriminative predictions that should not depend on the antigen are made invariant through data augmentation with random antigen sequences. See Algorithm 3 for an overview of a single guided diffusion step with LaMBO-2.

Model Architecture: our architecture for this experiment is inspired by the one proposed by Stanton et al. [73]. In particular we jointly a train an encoder shared between a generative discrete diffusion head and discriminative heads which predict expression and affinity. Rather than use a deep kernel GP, we simply ensemble 10 heads for each discriminative task to obtain uncertainty estimates. Like Stanton et al. [73] for this experiment we use 1D CNN residual blocks (kernel width 9), with layer normalization and sinusoidal position embeddings. The shared encoder was comprised of 4 residual blocks, and each task head was comprised of 2 residual blocks followed by a linear layer, with the exception of the generative head which was just a linear layer on top of the shared embeddings. Note that in future work self-attention layers could be used instead of CNN layers, as was the case for the pOAS experiments in Sec. 5. We set the embedding dimension to 32, and the latent channel dimension to 256.

Training Hyperparameters: The LaMBO-2 model is both a jointly trained generative and discriminative model, as well as a true multi-task model, which is necessary since measurements for various protein properties are often missing from a substantial fraction of rows in real-world datasets. We trained for 500K gradient updates using the Adam optimizer with $\eta =$ 1e-3, $\beta_0 = 0.99, \beta_1 = 0.999$. At each gradient step we randomly sampled a task head and task minibatch (batch-size 121) and updated the corresponding weights (including shared weights). We used a linear learning rate warmup over 10K gradient updates, and decayed the learning rate to 1e-6 with a cosine schedule. We did not regularize with weight decay or dropout.

Generation Hyperparameters: to generate the designs in Figure 7, we sampled 1K designs from a pool of seed antibody sequences hand-selected by domain experts. For each seed we set the total edit budget shared between chains to $B = 16$. In this experiment each infilling method took 16 diffusion steps, using an inverse linear noise schedule $\overline{\alpha}_t = 1/(1+t)$. Although the models were trained with a standard cosine noise schedule, we found the inverse linear schedule gave better results in terms of sample acquisition value at generation time. Within each diffusion step we took 64 Langevin steps, with noise scale $\tau =$ 1e-2. For guided infills with uniformly distributed edit positions we set $\tau =$ 1e6. For guided infills *with* saliency-informed edit position selection we set $\tau = 0.1$. We set $\lambda = 0.5$ to balance the tradeoff of sequence likelihood and value during guidance.

Generation Constraints: in addition to the edit budget locality constraint, our LaMBO-2 designs were also constrained to meet certain sequence liabilities constraints:

- **Canonical Cysteine Conservation:** there are specific conserved cysteine residues in antibody sequences which play a crucial role in the formation of disulfide bridges. Disulfide bridges are covalent bonds formed between two cysteine residues through oxidation of their sulfur atoms. These bridges contribute to the overall structural stability and integrity of antibodies.
- **No Unpaired Cysteines:** odd numbers of cysteines within individual chains (i.e. unpaired cysteines) are generally undesirable since they can lead to non-native disulfide bonds between different antibody molecules, which may disrupt assembly, folding, or function.
- **No Glycosylation Motifs:** A glycosylation motif is a specific amino acid sequence within a protein that serves as a recognition site for the attachment of sugar molecules. The presence of a glycosylation motif in a protein can affect its stability, solubility, activity, and function. The addition of sugar molecules can alter the protein's conformation, change its interactions with other proteins or molecules, and affect its trafficking and localization within the cell.

D.4 Training Data, Class Imbalance, and Label Smoothing

Training Data: the expression task heads were trained on a dataset of 10K linear transfection expression measurements, which was subsequently augmented to 160K rows by pairing the same measurements with different random antigens to teach the model to ignore the antigen sequence when predicting expression. The binding task heads were trained on a dataset of 10K SPR affinity measurements for various antigens, which was then augmented to 12K rows by pairing binders with different random antigens and imputing a non-binding label. This augmentation is important for

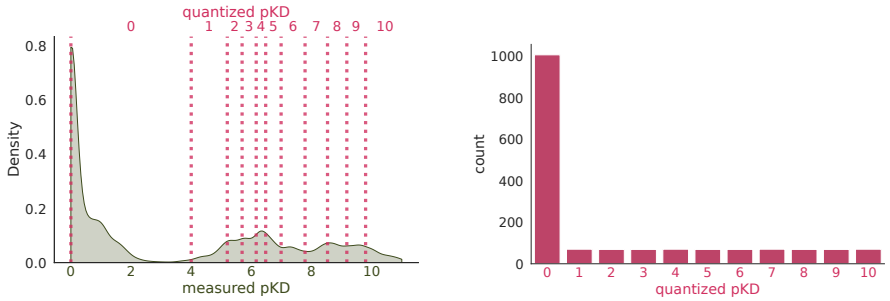

Figure 14: An illustration of using quantization to address heavily imbalanced data. On the right we show the original marginal label distribution in green, and the discretization boundaries as dotted lines. The boundaries are defined by a minimal level of affinity to be considered a binder ($\text{pKD} = 4$), and pKD deciles computed from the remaining measurements.

training a pan-target affinity model, since experimental measurements of affinity to off-target antigens are uncommon. Note that the expression and affinity data only partially overlapped, necessitating the multi-task architecture described in Appendix D.3. The generative diffusion head was trained only on binding antibody-antigen pairs in the SPR binding data.

We did not pretrain our LaMBO-2 models. It is likely that performance could be improved with the right pretraining corpus, however it is unclear if datasets like pOAS are particularly useful for pretraining antibody design models since most do not report antigen sequences and may not have the right level of variability. In any case, it is very encouraging to see positive real-world results before scaling in earnest.

Label Discretization. As noted above, biological data tends to be very imbalanced, and historical experimental data even more so since there are strong selection effects imposed by the scientists collecting the data. We chose to discretize continuous properties like expression yield and binding affinity, making it easier to correct for class imbalance by upsampling minority classes. In Figure 14 we illustrate our discretization scheme. Any antibody-antigen pair with $-\log(\text{KD})$ (pKD) less than 4 was assigned to the non-binding class 0. Then binders were assigned to classes 1 - 10 based on which pKD decile (computed from binders only) they resided in. One consequence of this scheme is increasing any objective value by one unit corresponds to moving up one decile in the empirical label distribution.

Training Discriminators on Noisy Inputs: the benefits of discretization are not limited to addressing class imbalance. Working with discretized labels also allowed a simple approach to training the discriminator on corrupted inputs inspired by label smoothing [76]. We train the discriminators with the same noise schedule as the diffusion model and the usual cross-entropy loss, using modified labels

$$\overline{\mathbf{y}}_t = \overline{\alpha}_t * \mathbf{y} + (1 - \overline{\alpha}_t)/K * \mathbf{1},$$

where $\mathbf{y}$ is the one-hot encoded label and K is the number of classes. Informally, as $\overline{\alpha}_t \to 0$ the discriminator reverts to a uniform prior since the inputs are not distinguishable. Training on corrupted inputs avoids evaluating the value gradient on out-of-distribution inputs during generation, and causes the strength of the value gradient to grow as the diffusion progresses and the samples become more defined.

D.5 Baselining LaMBO-2 Against Unguided Sequence and Structure-Based Diversification:

Structure-Based Diversification We have shown that we can effectively optimize antibodies for predicted yield and affinity, and our method performs well compared to unguided sequence-based infilling methods. We expand our evaluation for this task to include unguided infilling with DiffAb and RFDiffusion of CDRs H2 and H3 of hu4D5 (i.e. the seed), a publicly released therapeutic antibody that is ideally suited for structure-based methods since we have a ground truth crystal structure of hu4D5 docked with its target ERBB2. While it is not feasible to validate the resulting designs *in vitro* during the author response period, we can compare the AntiBERTy naturalness scores and the acquisition value (log expected hypervolume improvement or log-EHVI) of the designs relative to our guided infills (Fig. 15). To summarize, unguided structure-based infilling produces

high likelihood samples, but even when conditioned on the antigen the distribution shift toward better predicted function is very slight.

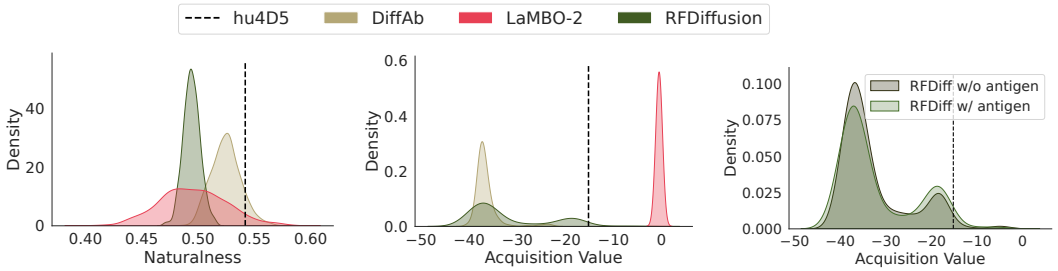

Figure 15: **(left)** we find that structure-based infills, particularly from DiffAb, tend to score consistently well on naturalness. Guided infilling produces a much wider range of scores, but the mode is very close to that of RFDiffusion. **(middle)** as assessed by the same model used to guide towards higher yield and binding affinity. The guided infills have very high acquisition value, since they were explicitly optimized for that outcome. Given 1024 samples each, DiffAb failed to produce any sequences of higher expected value than the seed, and RFDiffusion produced only 7 marginally improved designs. We also took the opportunity to assess the sensitivity of RFDiffusion to the antigen by comparing infills generated using the antibody stucture only **(right)**. While the effect is not large, antigen information does produce a small shift in the distribution of acquisition values to the right.

Sequence Diversification This *in silico* evaluation compares two variants of LaMBO-2 (one using NOS-C, the other NOS-D) against a competing method, walk-jump sampling (WJS), an unguided smoothed discrete sampling algorithm proposed by Frey et al. [30]. Each method generated 1K designs from the same set of seeds, and all methods were restricted to $B = 8$ edits. LaMBO-2 chose all edit positions automatically along the entire antibody sequence, whereas WJS was given manually selected edit positions restricted to CDRs only. In the left two panels of Figure 7 we compare the predicted expression yield, predicted binding affinity, and naturalness of the antibody designs, using the metric proposed by . Comparing the Pareto frontiers obtained from each set of designs, we see that while WJS excels at generating "natural" antibodies, it struggles to generate designs at the higher end of the objective range. Conversely LaMBO-2 designs (particularly those generated with NOS-C) have high predicted objective value but also lower naturalness scores. LaMBO-2 designs generated with NOS-D strike a balance between the two extremes.

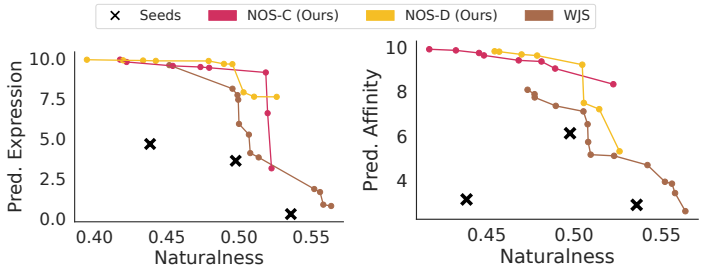

Figure 16: We evaluate LaMBO-2 in the context of real-world antibody lead optimization. LaMBO-2 can use either NOS-C or NOS-D to generate design libraries with higher predicted objective value than the unguided sampling baseline WJS [30], however intensive optimization comes at the cost of reduced naturalness (panels **left** and **center**).

D.6 Are Saliency Maps Reliable?

There is substantial controversy regarding the reliability of input-gradient-based feature attribution methods, specifically related to their ability to consistently highlight ground truth task-discriminative features and ignore irrelevant features. For example, Hooker et al. [41] claim that random attribution is competitive with input-gradient methods, and Casper et al. [11] claim that gradient-free attribution outperforms input-gradient competitors. On the other hand, many papers claim that specific types

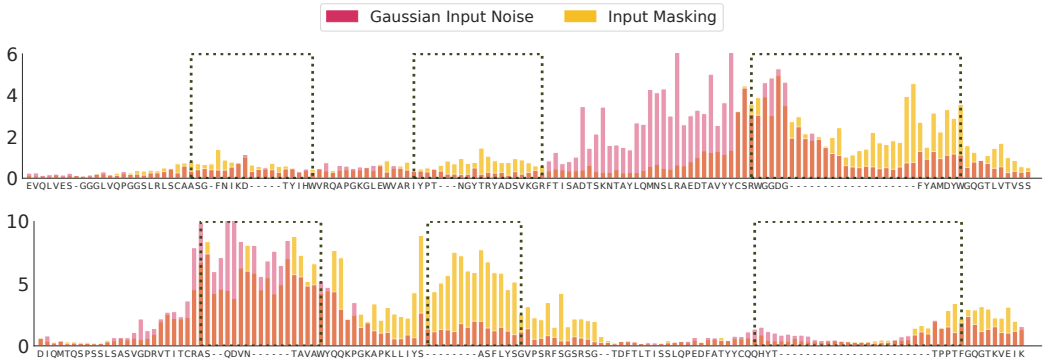

Figure 17: Binding affinity feature attributions for hu4D5 produced by independent models trained with different input corruptions. While the attributions do not match exactly, there is substantial agreement on the importance of CDRH3 (top panel) and CDRL1. Some importance is also assigned to various framework regions, which could be related to the fitness of different antibody germlines. We emphasize that these models were trained solely on aligned sequences, with no additional positional information.

of regularization can improve the performance of input-gradient attribution, including adversarial training [66], mask denoising [6], and model curvature penalties [72].

A thorough investigation of these claims is beyond the scope of this work, however we have found that saliency maps produced by independent models trained with different corruption processes seem to consistently highlight specific regions of the antibody sequence (Figure 17). It is also worth noting that most of the related literature evaluates feature attribution in the offline setting. In LaMBO-2 feature attributions are used online to intervene on the data collection process (specifically where to introduce changes in the antibody sequences). If LaMBO-2 changes a position that does not affect function it is reasonable to conjecture that input-gradient attributions would adjust accordingly after the model is retrained for the next round. Further investigation into feature attribution in decision-making contexts (as opposed to *post hoc* interpretability) is an exciting direction for future work.

D.7 Wetlab Validation

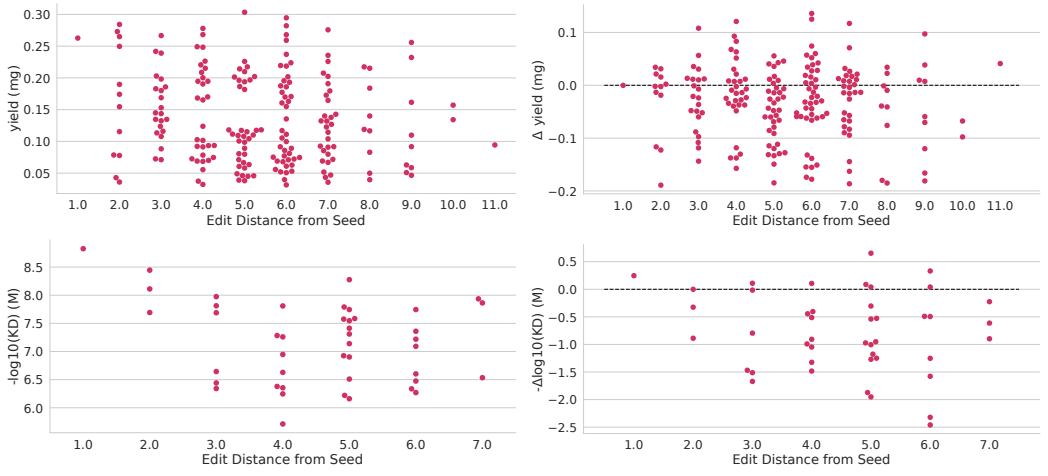

Figure 18: Here we show the experimentally validated yield **(top)** for all expressing designs and affinity **(bottom)** for all binding designs as a function of edit-distance from the original seed. In the right column we show the absolute measurement, and the left column shows the change relative to a seed measurement in the same batch.

In this section we briefly summarize the experimental procedures used to validate LaMBO-2 designs *in vitro*. Designed antibody sequences from LaMBO-2 were expressed and purified, and surface plasmon resonance (SPR) measurements were used to determine binding affinity. See Figure 18 for a plot of design binding affinity vs. edit distance from seed antibody.

Plasmid Construction and Antibody Production: synthesized DNA of antibody variable domains (Twist Biosciences) were cloned into mammalian expression vectors using Gibson assembly. The whole vector was amplified using PrimeStar Max polymerase (Takeda). PCR products were transfected transiently in 1mL Expi293 cell culture. Expression lasted 7 days before harvest. Antibodies were affinity purified over a MAb Select SuRe resin (Cytiva), and their concentration was measured by optical density at 280nM.

Binding Affinity Measurements: affinity of the antibodies towards their target antigen was measured by surface plasmon resonance (SPR) at 37 °C on a Biacore 8K instrument (Cytiva) in HBS-EP+ buffer (10 mM Hepes, pH 7.4, 150 mM NaCl, 0.3mM EDTA and 0.05% vol/vol Surfactant P20). Antibodies were captured on a Protein A chip and their target antigen were injected for 5 minutes and allowed to dissociate for 10 minutes at 30ul/min. The surface was regenerated between cycles with 10 mM glycine pH 1.5. Affinity constants were obtained using Biacore Insight (Cytiva) using a 1:1 binding kinetics model.