MITIGATING OVER-EXPLORATION IN LATENT SPACE OPTIMIZATION USING LES

Anonymous authors
Paper under double-blind review

ABSTRACT

We develop Latent Exploration Score (LES) to mitigate over-exploration in Latent Space Optimization (LSO), a popular method for solving black-box discrete optimization problems. LSO utilizes continuous optimization within the latent space of a Variational Autoencoder (VAE) and is known to be susceptible to over-exploration, which manifests in unrealistic solutions that reduce its practicality. LES leverages the trained decoder's approximation of the data distribution, and can be employed with any VAE decoder–including pretrained ones–without additional training, architectural changes or access to the training data. Our evaluation across five LSO benchmark tasks and twenty-two VAE models demonstrates that LES always enhances the quality of the solutions while maintaining high objective values, leading to improvements over existing solutions in most cases. We believe that new avenues to LSO will be opened by LES' ability to identify out of distribution areas, differentiability, and computational tractability.

1 INTRODUCTION

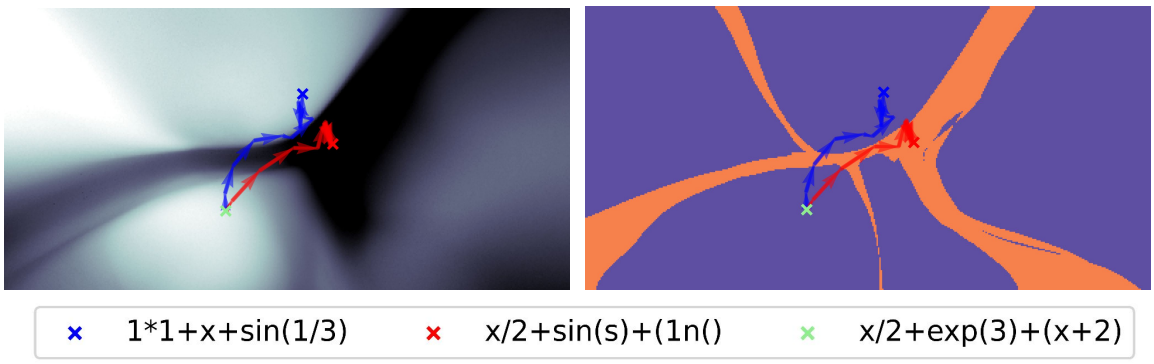

Figure 1: Incorporating LES promotes valid solutions. We consider the task of approximating the expression `1/3 + x + sin(x * x)`, using LSO. Optimization trajectories with (blue) and without (red) LES constraint in the latent space of a VAE are projected onto a two-dimensional subspace that contains the starting point and the endpoints obtained after 10 gradient ascent steps. In the left panel, we show the LES score for latent vectors on the two-dimensional subspace, with darker shades corresponding to lower LES. In the right panel, we show the validity of the decoder outputs for each latent vector, with orange denoting invalid generations. High LES values correlate with valid areas, and incorporating LES in LSO produce an expression that adheres to the grammatical rules of example 1.

Many important tasks in scientific fields, such as small molecule discovery and protein engineering, can be framed as discrete black-box optimization problems. In contrast to conditional sampling-based approaches, including GFlowNet (Bengio et al., 2023) and Diffusion (Corso et al., 2022; Igashov et al., 2024), which are better suited for applications like linker design (Du et al., 2024), optimization is particularly effective when the goal is to improve a specific property, such as enhancing a drug's safety.

Latent Space Optimization (LSO) was recently developed to enhance the sample efficiency of discrete optimization algorithms, such as genetic algorithms, in the black-box setting (Gómez-Bombarelli et al., 2018). LSO transfers the optimization problem to the domain of the latent space of a VAE, which can be efficiently explored using continuous

optimization techniques. However, ensuring that LSO solutions respect the structure of the original space remains a challenge. To illustrate this issue, we first provide some examples of such structures.

Example 1 (Arithmetic expressions)**.** *An expression built up using numbers, arithmetic operators and parentheses is called an arithmetic expression. However, not every sequence of the above elements correspond to a valid expression. For instance, the expression "`sin(x) + x`" is a valid expression while "`sin(xxx`" is not.*

Example 2 (Simplified molecular-input line-entry system (SMILES))**.** *SMILES provides a syntax to describe molecules using short ASCII strings. Atoms are represented by letters (e.g., water:"O"), bonds are represented by symbols (e.g., triple: "#", double: "=", ...), branches are represented in parentheses and cyclic structures are represented by inserting numbers at the beginning and the end. Like the arithmetic expressions case, not every combination of the elements described above corresponds to a valid molecule. For example, while "`C1CCCCC1`" is valid, both "`C1CCCCC2`" and "`C1CCCCC)`" are not.*

Example 3 (Quality filters for molecules)**.** *Chemists often seek molecules that not only optimize desired chemical properties but are also stable and easy to synthesize. This has led to the development of rules such as Lipinski's Rule of Five (RO5, Lipinski et al. (1997)), which helps determine if the bioavailability (i.e., the proportion of a drug or other substance that enters the circulation when introduced into the body) of a given compound meets a certain threshold. For example, RO5 suggests that poor absorption is more likely when the octanol-water partition coefficient (logP) exceeds 5. Similarly, the Pan Assay Interference Compounds (PAINS, Baell & Holloway (2010)) filter helps in identifying false positives in assay screenings. Recently, the `rd_filters` (Walters, 2019) package has curated many such rules and is considered a "high precision, low recall surrogate measure" (Brown et al., 2019). Following Notin et al. (2021) we consider a sample valid if it passes the `rd_filters` quality filters* [1]

Numerous directions have been explored to overcome the challenge of providing valid solutions, including specialized VAE architectures (Kusner et al., 2017; Jin et al., 2018) or robust representations for discrete data (Krenn et al., 2020). Additionally, constrained objectives can be formulated under the assumption that one has access to a function which quantifies the validity of any point in the latent space (Griffiths & Hernández-Lobato, 2020). However, in many realistic scenarios, such as example 3, these solutions may not be directly applicable, as the structure of the sequence space may not be sufficiently well understood. To address this, Notin et al. (2021) proposed using an estimator of the uncertainty of the decoder, based on the variational approximation to a posterior distribution over the VAE parameters, encouraging LSO to respect the sequence space structure (details are provided appendix D). Although this approach proved effective, the non-differentiable nature of the uncertainty score required its integration into LSO through heuristic approaches. Additionally, the computation of the uncertainty score is not exact (i.e., it relies on variational approximation and Monte Carlo sampling) and requires significant amount of time to compute. Therefore, there is a need for robust methods that work across different VAEs and sequence space structures, and can be easily integrated into existing LSO pipelines.

To achieve this goal, we develop LES, a score that can be used as a constraint in LSO optimization to increase the number of solutions that respect a given structure. The distinctive characteristics of LES are differentiability and robustness that allow its easy integration into existing LSO pipelines. Specifically, our contributions are as follows:

- We introduce LES, a score that achieves higher values in regions of the latent space closer to the training data. Our results demonstrate that LES is highly effective at identifying regions that preserve the structure of the sequence space. Although LES' computation scales cubically with the latent dimension, it is up to 80% faster than the current state-of-the-art for identifying out-of-distribution data points in the latent space of generative models for discrete sequences (tables 1 and 2).
- We develop a numerically stable optimization procedure to incorporate LES as a constraint in LSO.
- We evaluate LES-constrained LSO across thirty optimization tasks, including twenty-two VAEs and five benchmark problems, demonstrating its robustness in generating valid solutions and achieving high objective values. Specifically, in 18 out of the 30 LSO experiments, our method either finds the best solution on average or achieves a solution within 1 standard deviation of the best solution across 10 independent runs. This outperforms the six alternative methods we considered by 11% (tables 3 and 8).

2 Background: Latent Space Optimization

LSO is a method for solving black box optimization problems in discrete and structured spaces, such as the space of valid arithmetic expressions. Formally, let $\mathcal{V} \subset \mathbb{R}^{L \times D}$ be a discrete and structured space, represented as a sequence of

[1] We use the Inpharmatica rule set comprised of 91 alerts, which is the default option

L one-hot vectors of dimension D. We represent sequences of length L of categorical variables with D categories. L is set as the maximum sequence length that we are optimizing for, and one of the D categories is used as an "empty" category. For instance, in the case of valid arithmetic expressions, $\mathcal{V}$ would be the set of all sequences that define such expressions. Let $\mathcal{M} : \mathcal{V} \to \mathbb{R}$ be the objective function. LSO aims to solve,

$$\arg\max_{x \in \mathcal{V}} \mathcal{M}(x). \tag{1}$$

In this setting, we assume that evaluations of the objective function ($\mathcal{M}$) are expensive to conduct. For example, the objective may be the binding affinity of a compound to a given protein, measured through a wet lab experiment.

A popular approach to solve eq. (1) is Bayesian Optimization (BO), which utilizes first order optimization of a surrogate model for $\mathcal{M}$. However, since the space is discrete, first order optimization cannot be directly applied.
In an attempt to make BO applicable for solving eq. (1), Gómez-Bombarelli et al. (2018) proposed to transfer the optimization problem into that over a domain of the latent space of a deep generative model and subsequently perform BO in this space. The main idea is to (1) learn a continuous representation of the discrete objects (e.g., using a VAE) and (2) perform BO in the latent space while decoding the solution at each step. Formally, given a pre-trained encoder ($\mathbf{E}_\theta$) and decoder ($\mathbf{G}_\theta$) the initial labelled dataset $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$ is first encoded into the latent space $\mathcal{D}^{\boldsymbol{z}} = \{\boldsymbol{z}_i = \mathbf{E}_\theta(\boldsymbol{x}_i), y_i\}_{i=1}^n$. Using the encoded dataset, an iterative BO procedure is conducted, which we describe in algorithm 1. Most commonly, a Gaussian process is used as the surrogate model for $\mathcal{M}$, and the acquisition function is the expected improvement, defined as (Frazier, 2018)

$$\mathcal{A}_{\hat{f}}(\boldsymbol{z}) = \mathbb{E}_{\hat{f}} \max(\hat{f}(\boldsymbol{z}) - \max_i y_i, 0), \tag{2}$$

where the expectation is with respect to the distribution of the function $\hat{f}$, conditioned on $\mathcal{D}^{\boldsymbol{z}}$.

Algorithm 1 Latent Space Optimization

for $t = 1$ **to** T **do**

1. Fit a surrogate model $\hat{f}$ to the encoded dataset, $\mathcal{D}^{\boldsymbol{z}}$
2. Generate a new batch of query points by optimizing a chosen acquisition function ($\mathcal{A}$)

$$\boldsymbol{z}^{(\text{new})} = \arg\max_{\boldsymbol{z}} \mathcal{A}_{\hat{f}}(\boldsymbol{z}) \tag{3}$$

3. Decode $\boldsymbol{x}^{(\text{new})} = \mathbf{G}_\theta(\boldsymbol{z}^{(\text{new})})$, evaluate the corresponding true objective values ($y^{(\text{new})} = \mathcal{M}(\boldsymbol{x}^{(\text{new})})$) and update $\mathcal{D}^{\boldsymbol{z}}$ with $(\boldsymbol{z}^{(\text{new})}, y^{(\text{new})})$.

Over-exploration in LSO Multiple studies (Notin et al., 2021; Kusner et al., 2017) have found that unconstrained latent space optimization (LSO) often yields solutions that disregard the aforementioned structures. For example, when searching for arithmetic expressions, invalid equations like "$ssin(xxx$" frequently occur. Similarly, many solutions in molecule searches fail to pass basic quality filters (example 3), limiting their practical utility (Maus et al., 2022).

While acquisition functions such as expected improvement (eq. (2)) are designed to balance exploration and exploitation based on the estimated uncertainty from the Gaussian process model for $\mathcal{M}$. The frequent generation of invalid solutions during acquisition optimization, which implies that the estimated uncertainty can be problematic in this setting, underscores the need for additional regularization (Tripp et al., 2020), which we aim to address.

To mitigate over-exploration, we propose adding a penalty to eq. (3). The penalty uses a new score, giving higher values over the latent space valid set, defined as:

Definition 4 (Latent space valid set)**.** *Let $\mathbf{G}_\theta : \mathcal{Z} \to \mathbb{R}^{\mathrm{L}\times\mathrm{D}}$ be a decoder network, and let $\mathcal{V} \subset \mathbb{R}^{\mathrm{L}\times\mathrm{D}}$ be the set of valid sequences, the latent space valid set is defined as*

$$\{\boldsymbol{z}; \mathbf{G}_\theta(\boldsymbol{z}) \in \mathcal{V}\}. \tag{4}$$

The derivation of our score leverages the Continuous Piecewise Affine (CPA) representation of neural networks, which we briefly review below.

Deep generative networks as CPA Following (Humayun et al., 2022; 2021; Balestriero & Baraniuk, 2018), we consider the representation of Deep Generative Networks (DGNs) as Continues Piecewise Affine (CPA) Splines operators.

Let f_θ be any neural network with affine layers and piecewise affine activations then it holds that

$$f_\theta(\boldsymbol{z}) = \sum_{\omega \in \Omega} (\boldsymbol{A}_\omega \boldsymbol{z} + \boldsymbol{b}_\omega) \, 1_{\{\boldsymbol{z} \in \omega\}}, \tag{5}$$

where Ω is the input space partition induced by f_θ, ω is a particular region and the parameters $\boldsymbol{A}_\omega$ and $\boldsymbol{b}_\omega$ defines the affine transformation depending on ω.

In cases where the neural network f_θ is not composed solely of piecewise affine layers and activations, we leverage the result from Daubechies et al. (2022) to assert that eq. (5) either exactly represents f_θ or provides a sufficiently accurate approximation for our practical purposes (Humayun et al., 2022). We therefore argue that all the decoder neural networks included in our study (i.e., GRU, LSTM, and Transformers) can be approximated with high accuracy as continuous piecewise affine (CPA) functions.

3 A Latent Exploration Score to Reduce Over-Exploration in LSO

In this section, we introduce Latent Exploration Score (LES), our new score to reduce over-exploration in LSO. We begin by motivating LES and proceed to formally derive it in section 3.1. In section 3.2, we provide empirical evidence that LES gives higher values in the latent space valid set. The use of LES to regularize or constrain LSO is left for section 4.

Motivation Our goal is to develop a meaningful constraint for optimizing the acquisition function within a latent space of a given VAE. Specifically, we aim to construct a constraint that is a continuous function of $\boldsymbol{z}$ with higher values, indicating that it is more likely that $\boldsymbol{z}$ resides within valid regions of the latent space (definition 4).

Such a score should be higher in regions near training data points, assuming most of VAE training data is valid. To achieve this, we treat the latent space of the VAE as a probability space, i.e. $\boldsymbol{z} \sim p_{\boldsymbol{z}}$, for some prior distribution p (for example standard Gaussian). The prior should reflect our best guess for the distribution of the observed data in the latent space. Solutions are mapped back to sequences by the decoder through a deterministic (we do not consider $\boldsymbol{x}$ to follow a conditional distribution given $\boldsymbol{z}$) transformation of the latent vectors. Therefore, any distribution on the latent space defines a distribution over the space of sequences. Our score uses the density function of the push-forward measure of $\boldsymbol{x} = \mathbf{G}_\theta(\boldsymbol{z})$, which we call the sequence density. Consequentially, our score depends only on the decoder network, not the encoder, and can potentially be applied to other generative models like GANs or diffusion models.

Why use the sequence density function? We argue that for a well-trained decoder network, the density should be higher in areas of the sequence space close to the training data. To see why, consider a decoding model $\mathbf{G}_\theta$ trained on a dataset $\{(\boldsymbol{z}_i, \boldsymbol{x}_i)\}_{i=1}^n$. The average loss (denoted as L) at $\boldsymbol{z}$ is

$$\ell(\mathbf{G}_\theta(\boldsymbol{z})) = \mathbb{E}_{\boldsymbol{x} | \mathbf{E}_\theta(\boldsymbol{x}) = \boldsymbol{z}} L(\mathbf{G}_\theta(\boldsymbol{z}), \boldsymbol{x}). \tag{6}$$

As the training process is designed to minimize the population loss: $\mathbb{E}\ell(\mathbf{G}_\theta(\boldsymbol{z}))$, if successful, we hypothesize that the distribution of $\mathbf{G}_\theta(\boldsymbol{z})$ puts higher weight in the areas where $\ell(\mathbf{G}_\theta(\boldsymbol{z}))$ is low. Since we expect most of the training data to be valid and to achieve low expected loss, the sequence density should put higher weight on the latent space valid set. In, section 3.2 we provide an empirical validation for this hypothesis, for examples 1 to 3. We highlight that this relationship between the valid set and the sequence density depends on how well the decoder fits the data.

3.1 Derivation of LES

Analytical formula for LES DGNs for discrete sequences typically output a matrix of logits, transformed into normalized scores by the softmax function:

$$\mathbf{G}_\theta(\boldsymbol{z}) = \text{Softmax}(\mathbf{L}_\theta(\boldsymbol{z})). \tag{7}$$

$\mathbf{L}_\theta(\boldsymbol{z})$ is a D $\times$ L logits matrix (D - vocabulary size, L - sequence length) and Softmax is the softmax operation applied to every column of $\mathbf{L}_\theta(\boldsymbol{z})$. In order to avoid a violation of the assumption that $\mathbf{G}_\theta$ is bijective, we extend the function's output to include the normalizing constant for each column. We find that parametrizing the output to include the inverse of the normalizing constant (eq. (8)), helps in avoiding numerical instabilities that are caused by the constant being potentially very large. With this formulation, we can now derive the sequence density function.

Theorem 5 (DGN sequence density)**.** *Let*

$$\mathbf{G}_\theta(\boldsymbol{z}) = \left(\boldsymbol{p}_{\boldsymbol{z}}^{(1)}, (c_{\boldsymbol{z}}^{(1)})^{-1}, \ldots, \boldsymbol{p}_{\boldsymbol{z}}^{(L)}, (c_{\boldsymbol{z}}^{(L)})^{-1}\right) \tag{8}$$

$$= \boldsymbol{x}_{\boldsymbol{z}} \tag{9}$$

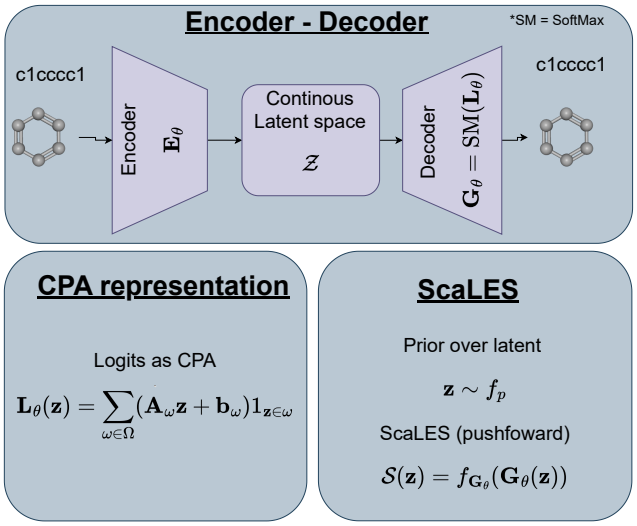

Figure 2: Derivation of LES. The decoder network ($\mathbf{G}_\theta$), which maps from the latent space to the output space, is assumed to be the composition of a softmax operation over a continuous piecewise affine (CPA) spline operator. LES is the density of a random variable ($\boldsymbol{z}$) in the latent space, under the decoder transformation. Calculating LES only requires a pre-trained decoder and LES does not have any hyperparameters.

where $\boldsymbol{p}_{\boldsymbol{z}}^{(i)} = Softmax(\mathbf{L}_\theta(\boldsymbol{z}))_{.i}$ *and* $c_{\boldsymbol{z}}^{(i)} = \sum_{j=1}^{D} e^{\mathbf{L}_\theta(\boldsymbol{z})_{ji}}$. *Assume that* $\mathbf{L}_\theta$ *is bijective and can be expressed as a CPA (eq.* (5)*), and that* $\boldsymbol{z} \sim p_{\boldsymbol{z}}$*, then the density function of* $\mathbf{G}_\theta(\boldsymbol{z})$ *is given by:*

$$f_p(\boldsymbol{z})\sqrt{\det\left(\sum_{i=1}^{\mathrm{L}}(\boldsymbol{A}_i^\dagger)^T(\boldsymbol{B}_i)^T\boldsymbol{B}_i\boldsymbol{A}_i^\dagger\right)} \tag{10}$$

for

$$\boldsymbol{B}_i = \left(diag\left(\frac{1}{(\boldsymbol{p}_{\boldsymbol{z}}^{(i)})_1},\ldots,\frac{1}{(\boldsymbol{p}_{\boldsymbol{z}}^{(i)})_{\mathrm{D}}}\right), -\mathbf{1}\frac{1}{c_{\boldsymbol{z}}^{(i)}}\right)^T \tag{11}$$

$$\boldsymbol{A}_i^\dagger = \left(\boldsymbol{A}_\omega^{(1)},\ldots,\boldsymbol{A}_\omega^{(L)}\right)^\dagger_{(i\cdot\mathrm{D}):(i+1\cdot\mathrm{D}).}, \tag{12}$$

where $\left(\boldsymbol{A}_\omega^{(1)},\ldots,\boldsymbol{A}_\omega^{(L)}\right)^\dagger$ *is the Moore–Penrose inverse of* $\left(\boldsymbol{A}_\omega^{(1)},\ldots,\boldsymbol{A}_\omega^{(L)}\right)$*, and* f_p *is the density function of* $p_{\boldsymbol{z}}$.

The proof is provided in appendix A. We define LES to be the logarithm of the determinant term,

$$\mathcal{S}(\boldsymbol{z}) = \log\left(\sqrt{\det\left(\sum_{i=1}^{\mathrm{L}}(\boldsymbol{A}_i^\dagger)^T(\boldsymbol{B}_i)^T\boldsymbol{B}_i\boldsymbol{A}_i^\dagger\right)}\right), \tag{13}$$

as the contribution of the prior is negligible in magnitude in all the decoders we study.

Remark: LES can be calculated directly without assuming the decoder logits follow a CPA function (Ben-Israel, 1999). However, using the expessions derived in eq. (13) has two computational benefits for calculating the derivative of LES. First, using Jacobi's formula and observing that $\sum_{i=1}^{\mathrm{L}}(\boldsymbol{A}_i^\dagger)^T(\boldsymbol{B}_i)^T\boldsymbol{B}_i\boldsymbol{A}_i^\dagger$ is a quadatic formula of the softmax probabilites, we can calculate the derivative of LES in closed form. Second, by the CPA assumption, the matrices $\boldsymbol{A}_\omega^{(i)}$ are a constant function of $\boldsymbol{z}$ and therefore $\frac{\partial \boldsymbol{A}_\omega^{(i)}}{\partial \boldsymbol{z}} = 0$. As a results, we avoid the need to calculate the hessian of the decoder when taking the derivative of LES.

Limitations of theorem 5 Our derivation relies on the decoder logits being (i) a CPA operator and (ii) bijective between the latent space and the generated manifold in the ambient space. We argue that (i) is not a restrictive assumption, as approximation theory has already demonstrated that any continuous model can be approximated by a CPA network. Therefore, one always recovers Theorem 5 even when using non-CPA models (see Daubechies et al. (2022) for proof).

On the other hand, (ii) is a stronger assumption that practitioners should be mindful of, as it would invalidate ScaLES as a meaningful metric for comparing different samples. For (ii) to be violated, i.e., for eq. (30) to be incorrect, the Lebesgue measure of the set $C_{\mathbf{G}_\theta} = \{\boldsymbol{z}; |; \exists \boldsymbol{z}^*; \mathbf{G}_\theta(\boldsymbol{z}) = \mathbf{G}_\theta(\boldsymbol{z}^*)\}$ must be larger than 0 (see lemma 7 for proof). This would suggest some degeneracy in the decoder function, where large regions of the latent space map to the same output, resulting in a zero gradient of the decoder with respect to its input. However, we believe this is rare in practice. In our experiments, where we compute the gradient of the decoder with respect to the input to calculate LES, we did not encounter any instances where the gradient was exactly zero.

Although we do not formally validate (ii) (the bijectivity assumption), we argue through our empirical analysis in table 2, conducted across 22 VAEs (including pre-trained models), that (ii) may hold or, at the very least, serve as a reasonable approximation for real-world VAEs.

Lastly, it is crucial to highlight that the ability of LES to detect out-of-distribution data is closely tied to the decoder's capacity to accurately model the data. While theorem 5 remains valid even for a poorly trained decoder under the specified conditions, we do not recommend practitioners to rely on LES in such scenarios.

Computing LES LES is a function of the matrices $\boldsymbol{B}_i$ and $\boldsymbol{A}_\omega$. The matrices $\boldsymbol{B}_i$ are a function of the logits and can be calculated using a single forward run of $\mathbf{G}_\theta$. The matrix $\boldsymbol{A}_\omega$ is equal to the derivative of $\mathbf{L}_\theta$ at $\boldsymbol{z}$, and can therefore can be obtained using automatic differentiation ((Paszke et al., 2017)). To avoid the (pseudo) inversion of the matrix $\boldsymbol{A}_\omega$ we can exploit the inverse function theorem which states that $J\mathbf{G}_\theta^{-1} = (J\mathbf{G}_\theta)^{-1}$ and we can take $\mathcal{S}(\boldsymbol{z}) = -0.5 \log\det((A_\omega \frac{\partial \mathbf{G}_\theta(\mathbf{L}_\theta)}{\partial \mathbf{L}_\theta})(A_\omega \frac{\partial \mathbf{G}_\theta(\mathbf{L}_\theta)}{\partial \mathbf{L}_\theta})^T)$, where $\frac{\partial \mathbf{G}_\theta(\mathbf{L}_\theta)}{\partial \mathbf{L}_\theta}$ (derivative of the softmax w.r.t the logits) admits a simple closed form solution. Ideally, LES can be computed by performing all of the above calculations in parallel using a single forward call to the $\mathbf{G}_\theta$ network. In addition, the computation of the determinant is done via SVD on a square matrix whose dimension is the latent dimension of the decoder (d) with a computational complexity of $\mathcal{O}(d^3)$.

In table 1 we provide the wall clock times for calculating LES for a batch of 20 latent vectors across all architectures and datasets studies in this work. LES is compared with the Bayesian uncertainty score proposed by Notin et al. (2021)(with the default configuration (`num_sampled_models=10, num_sampled_outcomes=40`)), which was previously used to regularize LSO.We also compare with a *Likelihood* score defined as

$$\ell(\boldsymbol{z}) = \max_{\boldsymbol{x}} p_{\mathbf{G}_\theta}(\boldsymbol{X} = \boldsymbol{x} | \boldsymbol{Z} = \boldsymbol{z}) \tag{14}$$

where $p_{\mathbf{G}_\theta}(\boldsymbol{z})$ is the distribution defined by the decoder softmax probabilities, which reflects the likelihood of the most likely $\boldsymbol{x}$ conditioned on the latent vector $\boldsymbol{z}$.

For 9 out of the 10 models, particularly when the latent dimension is larger (e.g., SMILES and SELFIES), LES is computed faster than the Uncertainty score, achieving reductions of up to 85% in some cases. As the Likelihood score requires only a single forward pass of the decoder it offers a more computationally efficient alternative, which comes with some performance trade-offs, as we show in sections 3.2 and 4 and appendix C.1.

3.2 Validating the relationship between LES and valid generation

To assess LES's ability to identify valid regions (as defined in examples 1 to 3), we sample data points in the latent space using the twenty-two VAEs studied in section 4. Specifically, we sample 500 data points from three distributions: train, prior ($\mathcal{N}(\mathbf{0}, \boldsymbol{I})$), and out-of-distribution ($\mathcal{N}(\mathbf{0}, \boldsymbol{I} \cdot 5)$). We decode each data point and determine if the decoded sequence is valid.

Identifying if a point in the latent space decodes into a valid sequence can be viewed as a classification problem, in which the different scores (i.e., LES or the Bayesian uncertainty score) provide (unnormalized) probabilities for a sequence being valid. We measure the performance of these scores using the AUROC metric. Besides LES, the Bayesian uncertainty score and the Likelihood score, we add three additional baseline scores for comparison. The first is the density of a standard Gaussian (**Prior**), which is the distribution the latent vectors are regularized to follow during VAE training. The second is the polarity score (**Polarity**) (Humayun et al., 2022), based only on the derivative of the decoder logits with respect to the latent vector, which shows the gains due to accounting for the softmax non-linearity in the derivation of LES (theorem 5). We also consider the average distance to the closest three data points within a random sample of 1000 points from the training data in the latent space (**Train distances**).

Table 1: Wall clock times in seconds (lower is better) for calculating LES and the Bayesian uncertainty score (Notin et al., 2021) for a sample of 20 latent vectors. In most cases (8 out of 10), LES is faster to compute, with a reduction in compute of up to 85%.

Dataset	Architecture	Latent dim.	LES (ours)	Uncertainty	Likelihood
Expressions	GRU	25	0.730	0.823	0.025
	LSTM	25	0.164	0.857	0.049
	Transformer	25	0.526	0.481	0.029
SMILES	GRU	56	0.663	3.157	0.103
	LSTM	56	0.696	3.990	0.123
	Transformer	56	0.581	0.726	0.185
SELFIES	GRU	75	0.498	1.925	0.064
	LSTM	75	0.525	2.442	0.071
	Transformer	75	0.451	0.583	0.085
	Transformer 25	256	7.422	42.882	0.410
Average	–	–	1.226	5.787	0.114

The results are shown in table 2. LES provides the best performance in 18 out of the 22 VAEs in this analysis, and in all cases provides a clear signal for identifying valid regions, as indicated by AUROC values that are at least 0.75. This is while being much faster to compute than the Uncertainty score (table 1) and without the need to store a potentially large array of latent vectors required by the Train distances approach.

Table 2: AUROC values (higher is better) for identifying valid data points within the latent space, across datasets and decoder architectures. Data points are sampled from the training data, the VAE prior (standard Gaussian) and out of distribution (Gaussian with std of 5). LES achieves the best performance in most cases (18 out of 22) and on average. In addition, LES achieves AUROC values of at least 0.75 in all cases, indicating it can effectively differentiate valid from invalid data points.

Dataset	Architecture	β	LES (ours)	Polarity (17)	Prior	Uncertainty	Train distances	Likelihood
SMILES	GRU	0.05	0.93	0.42	0.09	0.85	0.85	**0.94**
		0.1	**0.94**	0.72	0.12	0.84	0.86	**0.94**
		1.0	0.91	0.35	0.16	0.87	0.80	**0.92**
	LSTM	0.05	**0.99**	0.93	0.07	**0.99**	0.91	0.98
		0.1	0.89	0.67	0.21	0.89	0.81	**0.9**
		1.0	**0.97**	0.76	0.12	0.95	0.85	0.96
	Transformer	0.05	**0.93**	0.89	0.14	0.84	0.77	0.92
		0.1	**0.94**	0.91	0.14	0.87	0.86	0.93
		1.0	**0.97**	0.93	0.10	0.89	0.85	0.95
Expressions	GRU	0.05	**0.96**	0.89	0.38	**0.96**	0.67	0.88
		0.1	**0.94**	0.86	0.42	**0.94**	0.71	0.8
		1.0	**0.94**	0.80	0.57	**0.94**	0.75	0.89
	LSTM	0.05	**0.96**	0.86	0.38	0.90	0.67	**0.96**
		0.1	**0.95**	0.83	0.37	0.91	0.66	**0.95**
		1.0	**0.95**	0.79	0.56	0.91	0.72	0.91
	Transformer	0.05	**0.91**	0.79	0.43	0.86	0.71	0.90
		0.1	**0.91**	0.79	0.53	0.87	0.78	0.89
		1.0	0.86	0.61	0.70	**0.92**	0.89	**0.92**
SELFIES	Transformer	0.05	**1.0**	0.99	0.02	0.99	0.62	0.97
		0.1	**0.99**	0.98	0.03	0.96	0.81	0.98
		1.0	**0.95**	0.94	0.06	0.85	0.72	0.93
SELFIES 25	Transformer	–	**0.75**	0.39	0.69	0.33	0.69	0.70
	Average		**0.93**	0.78	0.29	0.88	0.77	0.91

4 LES-CONSTRAINED LSO

Our investigation in section 3.2 shows that LES is a robust score that obtains higher values in the latent space valid set (definition 4). Furthermore, LES is differentiable, which means it can easily be used to constrain any optimization problem. Therefore, we propose adding an explicit constraint to eq. (3), encouraging the solution to achieve a high LES value. We modify algorithm 1 by penalizing step (2) as follows:

$$\boldsymbol{z}^{\text{new}} = \arg\max_{\boldsymbol{z}} \mathcal{A}(\hat{f}(\boldsymbol{z})) + \lambda \mathcal{S}(\boldsymbol{z}). \tag{15}$$

4.1 EXPERIMENTAL SETUP

VAE models To evaluate the effectiveness of LES as a general purpose regularization method for LSO, we trained twenty-two VAEs, focusing on varying the decoder architectures and the β parameter, which controls the trade-off between reconstruction loss and alignment with the prior (KL divergence term). All models use a convolutional encoder based on the architecture proposed by Kusner et al. (2017), and were trained for 300 epochs using the Adam optimizer (Kingma, 2014) with a learning rate of 0.001 and batch size of 256.

The VAEs for the Expressions dataset, sourced from Kusner et al. (2017), were trained on 80k data points with a latent dimension of 25. The SMILES VAEs were trained on the ZINC250k dataset, consisting of approximately 250k drug-like molecules in SMILES format. Following Kusner et al. (2017) and Notin et al. (2021), a latent dimension of 56 was used. For the SELFIES VAEs, we used a subset of approximately 200k molecules from the ZINC250k dataset that passed a set of quality filters (Walters, 2019), using the SELFIES representation (Krenn et al., 2020), with a latent dimension of 75. Additionally, the pre-trained VAE by Maus et al. (2022) had a latent dimension of 256.

LSO setup Building on the setup described in Notin et al. (2021), we begin our experiments by training a single-task Gaussian Process on an initial dataset, where samples are encoded into the latent space and paired with their true objective values. For the Expressions and SMILES VAEs, we start with 500 data points, while for the SELFIES VAEs, we use a larger sample of 1500 data points. Across all experiments, we generate 500 candidate solutions per problem, using a batched approach that generates 20 solutions per batch.

For the SELFIES VAEs (both from Maus et al. (2022) and those trained by us), we employ a deep kernel that reduces the latent space to 12 dimensions before fitting the Gaussian Process. To mitigate vanishing gradients, we use log expected improvement (Ament et al., 2024) as our acquisition function, which is sequentially maximized during the optimization process.

Optimization tasks The Expressions dataset consists of arithmetic expressions that are functions of a single variable (e.g., $\sin(x)$, $1 + x * x$). Our objective is to find an expression that approximates `1/3 + x + sin(x * x)`, as described by Kusner et al. (2017). The optimization target is defined as $\mathcal{M}(\boldsymbol{x}) = -\log(1+\text{MSE}(\boldsymbol{x}))$, where $\text{MSE}(\boldsymbol{x})$ is the mean-squared error between the expression $\boldsymbol{x}$ and `1/3 + x + sin(x * x)`, evaluated over the range -10 to 10 using a grid of 1000 equally spaced points.

For the SMILES dataset, our goal is to maximize the octanol-water partition coefficient, which is calculated using the prediction model developed by Wildman & Crippen (1999). In the case of the SELFIES dataset, following Maus et al. (2022), we focus on three objectives: Perindopril MPO, Ranolazine MPO, and Zaleplon MPO, all of which are part of the Guacamol benchmarks (Brown et al., 2019). While the SELFIES syntax is 100% robust, we consider only solutions that pass a set of quality filters for evaluation (see example 3 for more details).

LES-constrained LSO For ease of implementation and numerical stability, when applying LES regularization, we adopt a simple optimization procedure in which the acquisition function is optimized using 10 steps of normalized gradient ascent. This is because we empirically find that the norm of the derivative of the constraint (i.e., $\mathcal{S}(\boldsymbol{z})$) is typically much larger than the norm of the derivative of the acquisition function. As a result, using the gradient ascent update rule $\boldsymbol{z}^{(i+1)} = \partial\mathcal{A}(\hat{f}(\boldsymbol{z}^{(i)})) + \lambda\partial\mathcal{S}(\boldsymbol{z}^{(i)})$ leads to a numerically unstable optimization process. To address this issue, we propose the following update rule:

$$\boldsymbol{z}^{(i+1)} = \frac{\partial\mathcal{A}(\hat{f}(\boldsymbol{z}^{(i)}))}{\|\partial\mathcal{A}(\hat{f}(\boldsymbol{z}^{(i)}))\|_2} + \lambda\frac{\partial\mathcal{S}_\rho(\boldsymbol{z}^{(i)})}{\|\partial\mathcal{S}_\rho(\boldsymbol{z}^{(i)})\|_2}. \tag{16}$$

We set $\lambda = 0.05$ for the Expressions dataset, $\lambda = 0.1$ for the SELFIES models we trained, and $\lambda = 0.5$ for both the SMILES dataset and the pre-trained SELFIES VAE. This is because, without regularization, the SMILES and

pre-trained SELFIES models tend to produce a lower percentage of valid solutions (table 10). Based on our findings, we suggest $\lambda = 0.5$ as a reasonable default value, while leaving the exploration of an optimal, potentially adaptive, choice for future work.

Table 3: Average of the top 20 solutions found during LSO (higher is better) across datasets and decoder architectures. We **bold** the best method and underline the second-best. The average ranking for each method (lower is better) is provided, along with the number of times each method is within one standard deviation of the best. LES achieves the highest value in most experiments (20 out of 30) and outperforms other methods in terms of both the average ranking and the frequency of being within one standard deviation of the best result.

	Architecture	β	LES	LSO (L-BFGS)	UC	LSO (GA)	Prior	TuRBO	Likelihood
Expressions	GRU	0.05	-1.43 (0.05)	-1.72 (0.07)	-1.5 (0.05)	-1.25 (0.06)	-1.28 (0.05)	-2.1 (0.11)	**-1.15 (0.09)**
		0.1	-1.34 (0.08)	-1.93 (0.09)	-2.01 (0.12)	-1.21 (0.09)	**-1.18 (0.11)**	-2.14 (0.18)	-1.31 (0.08)
		1	**-0.84 (0.02)**	-1.97 (0.07)	-0.91 (0.03)	**-0.84 (0.02)**	-0.89 (0.05)	-1.34 (0.05)	-0.88 (0.03)
	LSTM	0.05	-1.06 (0.06)	-1.78 (0.09)	-1.53 (0.06)	**-0.98 (0.05)**	-1.02 (0.03)	-1.59 (0.13)	-1.0 (0.07)
		0.1	-0.8 (0.03)	-1.39 (0.08)	-1.09 (0.04)	-0.79 (0.03)	-0.79 (0.03)	-1.32 (0.13)	**-0.75 (0.03)**
		1	**-1.79 (0.02)**	-2.04 (0.04)	-2.02 (0.03)	-1.81 (0.02)	-1.83 (0.02)	-2.22 (0.08)	-1.81 (0.03)
	Transformer	0.05	-1.0 (0.04)	-2.93 (0.13)	-1.64 (0.08)	-1.03 (0.04)	-0.99 (0.05)	-2.16 (0.09)	**-0.93 (0.05)**
		0.1	**-0.77 (0.02)**	-2.69 (0.25)	-1.79 (0.08)	**-0.77 (0.04)**	-0.78 (0.03)	-1.39 (0.12)	-0.81 (0.03)
		1	**-1.36 (0.09)**	-2.41 (0.09)	-1.52 (0.09)	-1.5 (0.07)	-1.41 (0.08)	-1.77 (0.12)	-1.45 (0.12)
SELFIES	Transformer (pdop)	0.05	**0.37 (0.0)**	0.21 (0.01)	0.36 (0.0)	**0.37 (0.0)**	**0.37 (0.0)**	0.04 (0.01)	**0.37 (0.0)**
		0.1	**0.36 (0.0)**	0.19 (0.01)	0.35 (0.0)	**0.36 (0.0)**	0.34 (0.0)	0.01 (0.0)	**0.36 (0.0)**
		1	**0.35 (0.0)**	0.25 (0.01)	0.31 (0.01)	0.33 (0.0)	0.28 (0.01)	0.23 (0.02)	0.34 (0.0)
	Transformer (rano)	0.05	0.21 (0.0)	0.03 (0.0)	**0.22 (0.0)**	0.21 (0.0)	0.21 (0.0)	0.06 (0.0)	0.21 (0.0)
		0.1	0.21 (0.0)	–	**0.23 (0.01)**	0.21 (0.0)	0.22 (0.0)	0.03 (0.01)	0.22 (0.0)
		1	0.21 (0.0)	0.07 (0.0)	**0.22 (0.01)**	0.19 (0.0)	0.2 (0.0)	–	0.21 (0.0)
	Transformer (zale)	0.05	**0.33 (0.0)**	0.13 (0.01)	0.32 (0.0)	**0.33 (0.0)**	0.32 (0.0)	0.04 (0.01)	**0.33 (0.0)**
		0.1	**0.34 (0.0)**	0.06 (0.0)	0.31 (0.01)	0.32 (0.01)	0.31 (0.0)	0.04 (0.01)	**0.34 (0.0)**
		1	**0.31 (0.0)**	0.18 (0.02)	0.26 (0.01)	0.29 (0.01)	0.23 (0.01)	0.17 (0.02)	0.3 (0.01)
SELFIES 25	Transformer (pdop)	1	0.39 (0.01)	0.33 (0.01)	0.38 (0.0)	0.38 (0.0)	**0.4 (0.0)**	**0.4 (0.01)**	0.37 (0.0)
	Transformer (rano)	1	**0.27 (0.01)**	0.12 (0.01)	0.26 (0.01)	0.26 (0.0)	**0.27 (0.0)**	0.15 (0.01)	0.24 (0.0)
	Transformer (zale)	1	0.29 (0.01)	0.29 (0.01)	0.3 (0.01)	0.31 (0.0)	0.3 (0.01)	**0.36 (0.01)**	0.24 (0.01)
SMILES	GRU	0.05	**2.31 (0.04)**	0.52 (0.12)	2.18 (0.05)	2.21 (0.05)	2.2 (0.04)	0.82 (0.15)	**2.31 (0.03)**
		0.1	**2.3 (0.05)**	0.12 (0.14)	1.68 (0.04)	2.09 (0.07)	1.93 (0.06)	0.41 (0.19)	2.12 (0.07)
		1	**1.64 (0.14)**	–	–	1.19 (0.2)	0.58 (0.28)	-0.16 (0.0)	1.49 (0.14)
	LSTM	0.05	**2.33 (0.03)**	0.66 (0.12)	2.02 (0.03)	2.22 (0.05)	2.13 (0.05)	1.54 (0.41)	2.28 (0.03)
		0.1	**1.57 (0.1)**	–	–	0.8 (0.12)	0.9 (0.09)	–	1.43 (0.12)
		1	**1.94 (0.1)**	0.7 (0.2)	0.95 (0.24)	1.14 (0.21)	0.81 (0.26)	0.52 (0.26)	1.67 (0.13)
	Transformer	0.05	**2.26 (0.04)**	0.42 (0.15)	2.04 (0.05)	2.22 (0.03)	2.24 (0.03)	0.97 (0.19)	2.25 (0.03)
		0.1	**2.26 (0.03)**	0.61 (0.15)	2.08 (0.04)	2.21 (0.03)	2.17 (0.03)	0.62 (0.14)	2.23 (0.02)
		1	**2.17 (0.05)**	0.23 (0.12)	1.32 (0.09)	1.98 (0.06)	1.82 (0.06)	0.48 (0.18)	2.16 (0.05)
	Average rank		**1.83**	6.47	4.4	2.97	3.5	6.03	2.8
	# within 1 std of best		**21**	0	4	8	8	2	14

Benchmark methods We compare our LES-constrained method (**LES**) with five alternative approaches for optimizing the acquisition function. First, we evaluate a non-regularized version of eq. (16) , where $\lambda = 0$ (**LSO (GA)**), and a two regularization methods that uses the prior density (i.e., ℓ_2 regularization) and likelihood (eq. (14)) instead of

LES (**Prior**, and **Likelihood** respectively), using similar λ values described above($\lambda = 0.05$ for Expressions, $\lambda = 0.1$ for SELFIES VAEs trained by us, and $\lambda = 0.5$ for SMILES and pre-trained SELFIES VAE).

Additionally, we consider optimizing the acquisition function using the Limited-memory BFGS method within a symmetric hypercube centered at 0 (**LSO (L-BFGS)**), which is the default approach in the `BoTorch` package (Balandat et al., 2020). Since recent state-of-the-art LSO pipelines (Maus et al., 2022; Lee et al., 2024) have utilized trust regions centered around the best observed value (Eriksson et al., 2019), we also compare with this approach (**TuRBO**). Lastly, we implement the Uncertainty Censoring (**UC**) method proposed by Notin et al. (2021), which suggests early stopping of the optimization when the estimated uncertainty exceeds a certain threshold. For this threshold, we use the 99th percentile of the observed uncertainty values in the training data, as recommended by Notin et al. (2021).

Hyperparameters We calibrate the step size, which affects **LES**, **UC**, **Prior**, **Likelihood**, and **LSO (GA)**, to ensure that our gradient ascent procedure (with $\lambda = 0$) consistently improves the acquisition function values across different initializations. The same step size is applied to all models within the same dataset: Expressions = 0.8, SMILES = 0.003, SELFIES = 0.03, and SELFIES pre-trained = 0.3. The LSO (L-BFGS) method has a single hyperparameter, the facet length, which is set to 5. For TuRBO, there are three primary hyperparameters: the initial length, which we set to 0.8, along with the success and failure tolerances, determining when to expand or shrink the trust region, set at 10 and 2, respectively. An ablation study for these choices is provided in appendix B.

4.2 Results

Optimization results The experimental results, presenting the average across 10 independent LSO runs for the top 20 and best solutions found, are summarized in tables 3 and 8 respectively. In both cases, LES achieves the average best performance most frequently (20 and 14 out of 30 times, respectively). Furthermore, LES outperforms other methods in both the average ranking across LSO tasks and the frequency with which it falls within one standard deviation of the best-performing method. These findings demonstrate that using LES as a regularization technique generally enhances optimization performance.

In table 10 we present the percentage of valid solutions found by each method across datasets and decoder architectures. LES improves upon the non-regularized version of gradient ascent by 7% on average and upon TuRBO and LSO (L-BFGS) by 24% and 35% on average respectively. While UC achieves a 3% higher percentage of valid solutions on average, we show in table 9 that the number of valid solutions can be increased by setting a higher λ value. However, in our experiments, this did not improve the optimization performance.

5 Discussion

We proposed LES to mitigate over-exploration in latent space optimization (LSO). LES is differentiable and fully parallelizable. Extensive evaluations demonstrate that incorporating LES as a penalty in LSO consistently enhances solution quality and objective outcomes. Moreover, LES outperforms alternative regularization techniques, proving to be the most robust across diverse datasets and varying definitions of validity. In addition, LES has only a single hyperparameter (the regularization strength), and we observe empirically that deploying LES can provide significant performance gains–without any instability or having to resort to implementation tricks. We therefore believe LES offers a powerful and robust approach for discovering more realistic solutions, particularly when the criteria for realism are difficult to define or validate.

While LES is fully parallelizable, it requires the calculation of the derivative of the decoder as well as the determinant of the change-of-variables term, which can be computationally expensive. This step can become a bottleneck when the size of the output and the latent dimension are both large. It is left for future work to develop a fast approximation for this operation in order to enable the use of LES in applications involving large generative models, such as large language models.

REFERENCES

Sebastian Ament, Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Unexpected improvements to expected improvement for bayesian optimization. *Advances in Neural Information Processing Systems*, 36, 2024.

Jonathan B Baell and Georgina A Holloway. New substructure filters for removal of pan assay interference compounds (pains) from screening libraries and for their exclusion in bioassays. *Journal of medicinal chemistry*, 53(7):2719–2740, 2010.

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020. URL `http://arxiv.org/abs/1910.06403`.

Randall Balestriero and Richard Baraniuk. Mad max: Affine spline insights into deep learning. *arXiv preprint arXiv:1805.06576*, 2018.

Adi Ben-Israel. The change-of-variables formula using matrix volume. *SIAM Journal on Matrix Analysis and Applications*, 21(1):300–312, 1999.

Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *The Journal of Machine Learning Research*, 24(1):10006–10060, 2023.

Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3):1096–1108, 2019.

Gabriele Corso, Hannes Stärk, Bowen Jing, Regina Barzilay, and Tommi Jaakkola. Diffdock: Diffusion steps, twists, and turns for molecular docking. *arXiv preprint arXiv:2210.01776*, 2022.

Ingrid Daubechies, Ronald DeVore, Nadav Dym, Shira Faigenbaum-Golovin, Shahar Z Kovalsky, Kung-Chin Lin, Josiah Park, Guergana Petrova, and Barak Sober. Neural network approximation of refinable functions. *IEEE Transactions on Information Theory*, 69(1):482–495, 2022.

Yuanqi Du, Arian R Jamasb, Jeff Guo, Tianfan Fu, Charles Harris, Yingheng Wang, Chenru Duan, Pietro Liò, Philippe Schwaller, and Tom L Blundell. Machine learning-aided generative molecular design. *Nature Machine Intelligence*, pp. 1–16, 2024.

David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable global optimization via local bayesian optimization. *Advances in neural information processing systems*, 32, 2019.

Peter I Frazier. Bayesian optimization. In *Recent advances in optimization and modeling of contemporary problems*, pp. 255–278. Informs, 2018.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

Ryan-Rhys Griffiths and José Miguel Hernández-Lobato. Constrained bayesian optimization for automatic chemical design using variational autoencoders. *Chemical science*, 11(2):577–586, 2020.

Ahmed Imtiaz Humayun, Randall Balestriero, and Richard Baraniuk. Magnet: Uniform sampling from deep generative network manifolds without retraining. In *International Conference on Learning Representations*, 2021.

Ahmed Imtiaz Humayun, Randall Balestriero, and Richard Baraniuk. Polarity sampling: Quality and diversity control of pre-trained generative networks via singular values. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10641–10650, 2022.

Ilia Igashov, Hannes Stärk, Clément Vignac, Arne Schneuing, Victor Garcia Satorras, Pascal Frossard, Max Welling, Michael Bronstein, and Bruno Correia. Equivariant 3d-conditional diffusion model for molecular linker design. *Nature Machine Intelligence*, pp. 1–11, 2024.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pp. 2323–2332. PMLR, 2018.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. Self-referencing embedded strings (selfies): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4):045024, 2020.

Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *International conference on machine learning*, pp. 1945–1954. PMLR, 2017.

Seunghun Lee, Jaewon Chu, Sihyeon Kim, Juyeon Ko, and Hyunwoo J Kim. Advancing bayesian optimization via learning correlated latent space. *Advances in Neural Information Processing Systems*, 36, 2024.

Christopher A Lipinski, Franco Lombardo, Beryl W Dominy, and Paul J Feeney. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Advanced drug delivery reviews*, 23(1-3):3–25, 1997.

Natalie Maus, Haydn Jones, Juston Moore, Matt J Kusner, John Bradshaw, and Jacob Gardner. Local latent space bayesian optimization over structured inputs. *Advances in neural information processing systems*, 35:34505–34518, 2022.

Pascal Notin, José Miguel Hernández-Lobato, and Yarin Gal. Improving black-box optimization in vae latent space using decoder uncertainty. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 802–814. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/06fe1c234519f6812fc4c1baae25d6af-Paper.pdf.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Austin Tripp, Erik Daxberger, and José Miguel Hernández-Lobato. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. *Advances in Neural Information Processing Systems*, 33:11259–11272, 2020.

P. Walters. rd filters, 2019. URL https://github.com/PatWalters/rd_filters. Accessed: January 14, 2019.

Scott A Wildman and Gordon M Crippen. Prediction of physicochemical parameters by atomic contributions. *Journal of chemical information and computer sciences*, 39(5):868–873, 1999.

A PROOFS

Lemma 6. *Let f_θ be a DGN as defined in eq.* (8) *and assume that f_θ can be expressed as a CPA (eq.* (5)*) and is inevitable, then*

$$Jf_\theta^{-1}(\boldsymbol{x}) = \left(\begin{bmatrix} \boldsymbol{B}_1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \boldsymbol{B}_{\mathrm{L}} \end{bmatrix} \boldsymbol{A}_\omega^\dagger \right)^T, \tag{17}$$

where $\boldsymbol{A}_\omega^\dagger$ is the Moore–Penrose inverse of the slope matrix, at the knot whose image constrains $\boldsymbol{x}$, and

$$\boldsymbol{B}_i = \left(diag \left(\frac{1}{(\boldsymbol{p}_{\boldsymbol{z}}^{(i)})_1}, \ldots, \frac{1}{(\boldsymbol{p}_{\boldsymbol{z}}^{(i)})_{\mathrm{D}}} \right), -\mathbf{1} \frac{1}{c_{\boldsymbol{z}}^{(i)}} \right)^T. \tag{18}$$

Proof. First we write

$$f_\theta(\boldsymbol{z}) = \mathrm{Softmax}_+(\ell_\theta(\boldsymbol{z})), \tag{19}$$

Where $\mathrm{Softmax}_+$ is the extension of the column wise Softmax function to include the normalizing constants. Specifically, for L by D $\ell_\theta(\boldsymbol{z})$ matrix, we have

$$\mathrm{Softmax}_+(\ell_\theta(\boldsymbol{z})) = \left(\boldsymbol{p}_{\boldsymbol{z}}^{(1)}, (c_{\boldsymbol{z}}^{(1)})^{-1}, \ldots, \boldsymbol{p}_{\boldsymbol{z}}^{(L)}, (c_{\boldsymbol{z}}^{(L)})^{-1} \right) = \boldsymbol{x}_{\boldsymbol{z}}, \tag{20}$$

with $\boldsymbol{p}_{\boldsymbol{z}}^{(i)} = (\frac{e^{\ell_\theta(\boldsymbol{z})_{1i}}}{c_{\boldsymbol{z}}^{(i)}})$, and $c_{\boldsymbol{z}}^{(i)} = \sum_{j=1}^{\mathrm{D}} \exp(\ell_\theta(\boldsymbol{z})_{ji})$.
Next,

$$f_\theta^{(-1)}(\boldsymbol{x}) = \ell_\theta^{-1}(\mathrm{Softmax}_+^{-1}(\boldsymbol{x})) \tag{21}$$

A direct calculation yields,

$$\mathrm{Softmax}_+^{-1}(\boldsymbol{x}) = \left(\log(\boldsymbol{p}_{\boldsymbol{z}}^{(1)}) + \log(c_{\boldsymbol{z}}^{(1)}), \ldots, \log(\boldsymbol{p}_{\boldsymbol{z}}^{(L)}) + \log(c_{\boldsymbol{z}}^{(L)}) \right). \tag{22}$$

As we assume ℓ_θ is bijective and can be written as

$$\ell_\theta(\boldsymbol{z}) = \sum_{\omega \in \Omega} (\boldsymbol{A}_\omega \boldsymbol{z} + \boldsymbol{b}_\omega) \, 1_{\boldsymbol{z} \in \omega}, \tag{23}$$

we have that

$$\ell_\theta^{-1}(\mathrm{Softmax}_+^{-1}(\boldsymbol{x})) = (\mathrm{Softmax}_+^{-1}(\boldsymbol{x}) - \boldsymbol{b}_\omega) \boldsymbol{A}_\omega^\dagger. \tag{24}$$

Lastly, as

$$\frac{\partial \mathrm{Softmax}_+^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}} = \begin{bmatrix} \boldsymbol{B}_1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \boldsymbol{B}_{\mathrm{L}} \end{bmatrix}, \tag{25}$$

for

$$\boldsymbol{B}_i = \left(\mathrm{diag} \left(\frac{1}{(\boldsymbol{p}_{\boldsymbol{z}}^{(i)})_1}, \ldots, \frac{1}{(\boldsymbol{p}_{\boldsymbol{z}}^{(i)})_{\mathrm{D}}} \right), -\mathbf{1} \frac{1}{c_{\boldsymbol{z}}^{(i)}} \right)^T. \tag{26}$$

we obtain the final result. □

Proof of theorem 5. First, we note that by our invertibility assumption we have that $\mathbb{P}(\boldsymbol{x} \in W) = \mathbb{P}(\boldsymbol{z} \in f_\theta^{(-1)}(W))$. We then proceed with a direct calculation

$$\begin{aligned}
\mathbb{P}(\boldsymbol{x} \in W) &= \mathbb{P}(\boldsymbol{z} \in f_\theta^{(-1)}(W)) && (27)\\
&= \sum_{\omega \in \Omega} \mathbb{P}(\boldsymbol{z} \in (f_\theta^{(-1)}(W) \cap \omega)) && (28)\\
&= \sum_{\omega \in \Omega} \int_{f_\theta^{(-1)}(W) \cap \omega} f_{\boldsymbol{z}}(\boldsymbol{z}) d\boldsymbol{z} && (29)\\
&= \sum_{\omega \in \Omega} \int_{W \cap f_\theta(\omega)} f_{\boldsymbol{z}}(f_\theta^{(-1)}(\boldsymbol{x})) \sqrt{\det\left(J f_\theta^{(-1)}(\boldsymbol{x}) J f_\theta^{(-1)}(\boldsymbol{x})^T\right)} d\boldsymbol{x} && (30)\\
&= \int_W \sum_{\omega \in \Omega} f_{\boldsymbol{z}}(f_\theta^{(-1)}(\boldsymbol{x})) \sqrt{\det\left(J f_\theta^{(-1)}(\boldsymbol{x}) J f_\theta^{(-1)}(\boldsymbol{x})^T\right)} 1_{\{\boldsymbol{x} \in f_\theta(\omega)\}} d\boldsymbol{x}. && (31)
\end{aligned}$$

Using lemma 6, we get that the volume element is

$$\begin{aligned}
J f_\theta^{(-1)}(\boldsymbol{x}) J f_\theta^{(-1)}(\boldsymbol{x})^T = &\left(\begin{bmatrix} \boldsymbol{B}_1 & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{B}_{\mathrm{L}} \end{bmatrix} \boldsymbol{A}_\omega^\dagger \right)^T \left(\begin{bmatrix} \boldsymbol{B}_1 & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{B}_{\mathrm{L}} \end{bmatrix} \boldsymbol{A}_\omega^\dagger \right) && (32)\\
&\left((\boldsymbol{A}_\omega^\dagger)^T \begin{bmatrix} \boldsymbol{B}_1^T & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{B}_{\mathrm{L}}^T, \end{bmatrix} \right) \left(\begin{bmatrix} \boldsymbol{B}_1 & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{B}_{\mathrm{L}} \end{bmatrix} \boldsymbol{A}_\omega^\dagger \right) && (33)\\
&= \sum_{i=1}^{\mathrm{L}} (\boldsymbol{A}_i^\dagger)^T (\boldsymbol{B}_i)^T \boldsymbol{B}_i \boldsymbol{A}_i^\dagger, && (34)
\end{aligned}$$

where $\boldsymbol{A}_i^\dagger = \left(\boldsymbol{A}_\omega^{(1)}, \ldots, \boldsymbol{A}_\omega^{(L)} \right)^\dagger_{(i \cdot \mathrm{D}):(i+1 \cdot \mathrm{D}).}$. □

Lemma 7. *Let $f : \mathcal{Z} \to \mathcal{X}$ be a function and define $f^\dagger : \mathcal{X} \to \mathcal{Z}$ as $f^\dagger(x) \in \{z : f(z) = x\}$. Let μ be the Lebesgue measure and assume that $\mu(\{z; \exists z' \text{ s.t. } f(z) = f(z')\}) = 0$, then for every $B \subseteq \mathcal{Z}$ we have*

$$\mu(\{z; f(z) \in B\}) = \mu(f^\dagger(B)) \tag{35}$$

Proof. We proceed with direct calculation

$$\mu(\{z; f(z) \in B\}) \leq \mu(f^\dagger(B)) + \mu(\{z; \exists z' \text{ s.t. } f(z) = f(z')\}) \tag{36}$$

Now assume that $\mu(\{z; \exists z' \text{ s.t. } f(z) = f(z')\}) = 0$, we have that $\mu(\{z; f(z) \in B\}) \leq \mu(f^\dagger(B))$. The other direction follows immediately from the definition of $f^\dagger$. □

lemma 7 implies that eq. (30) can still hold under the assumption that $\mu(\{z; \exists z' \text{ s.t. } f_\theta(z) = f_\theta(z')\}) = 0$.

Table 4: Ablation study for the initial length, success tolerance and failure tolerance for the TuRBO method. Average across 10 independent runs, of the top 20 best values across datasets, architectures. Column names indicate the length/success/fail values. Results from the main paper are in bold.

	Architecture	β	0.8/10/10	0.8/10/2	0.8/2/10	0.8/2/2	1.6/10/10	1.6/10/2	1.6/2/10	1.6/2/2
Expressions	GRU	0.05	-2.03 (0.12)	**-2.1 (0.11)**	-2.0 (0.09)	-2.0 (0.11)	-1.93 (0.09)	-1.92 (0.08)	-1.91 (0.06)	-1.93 (0.07)
		0.1	-2.18 (0.15)	**-2.14 (0.18)**	-2.11 (0.2)	-2.14 (0.2)	-1.98 (0.13)	-2.05 (0.12)	-2.0 (0.17)	-2.09 (0.22)
		1	-1.39 (0.12)	**-1.34 (0.05)**	-1.56 (0.11)	-1.61 (0.11)	-1.36 (0.08)	-1.42 (0.11)	-1.45 (0.12)	-1.38 (0.12)
	LSTM	0.05	-1.45 (0.09)	**-1.59 (0.13)**	-1.63 (0.12)	-1.49 (0.1)	-1.46 (0.08)	-1.52 (0.08)	-1.51 (0.12)	-1.31 (0.07)
		0.1	-1.31 (0.12)	**-1.32 (0.13)**	-1.37 (0.11)	-1.31 (0.11)	-1.31 (0.1)	-1.36 (0.13)	-1.36 (0.11)	-1.35 (0.09)
		1	-2.16 (0.06)	**-2.22 (0.08)**	-2.23 (0.08)	-2.23 (0.08)	-2.13 (0.07)	-2.16 (0.06)	-2.15 (0.05)	-2.08 (0.06)
	Transformer	0.05	-2.07 (0.12)	**-2.16 (0.09)**	-2.03 (0.17)	-2.15 (0.14)	-1.96 (0.13)	-1.74 (0.13)	-2.09 (0.1)	-1.79 (0.11)
		0.1	-1.45 (0.13)	**-1.39 (0.12)**	-1.56 (0.13)	-1.38 (0.13)	-1.32 (0.11)	-1.43 (0.17)	-1.4 (0.1)	-1.33 (0.12)
		1	-1.96 (0.11)	**-1.77 (0.12)**	-1.81 (0.11)	-1.85 (0.09)	-1.85 (0.11)	-1.76 (0.08)	-1.78 (0.08)	-1.75 (0.09)
SELFIES	Transformer (pdop)	0.05	0.02 (0.01)	**0.04 (0.01)**	0.04 (0.01)	0.04 (0.01)	0.09 (0.01)	0.09 (0.03)	0.08 (0.01)	0.09 (0.02)
		0.1	0.01 (0.0)	**0.01 (0.0)**	0.02 (0.01)	0.02 (0.01)	0.03 (0.01)	0.08 (0.03)	0.04 (0.02)	0.05 (0.02)
		1	0.24 (0.01)	**0.23 (0.02)**	0.21 (0.02)	0.24 (0.01)	0.27 (0.01)	0.26 (0.01)	0.28 (0.01)	0.26 (0.01)
	Transformer (rano)	0.05	0.06 (0.01)	**0.06 (0.0)**	0.06 (0.01)	0.06 (0.01)	0.04 (0.01)	0.02 (0.0)	0.04 (0.0)	0.03 (0.01)
		0.1	0.03 (0.01)	**0.03 (0.01)**	0.03 (0.0)	0.02 (0.0)	–	–	–	–
		1	–	**–**	0.08 (0.0)	–	0.05 (0.01)	0.07 (0.01)	0.07 (0.0)	–
	Transformer (zale)	0.05	0.02 (0.01)	**0.04 (0.01)**	0.02 (0.01)	0.04 (0.01)	0.07 (0.01)	0.07 (0.02)	0.06 (0.01)	0.06 (0.01)
		0.1	0.04 (0.01)	**0.04 (0.01)**	0.04 (0.01)	0.03 (0.01)	0.01 (0.0)	0.02 (0.0)	0.09 (0.0)	0.04 (0.0)
		1	0.16 (0.02)	**0.17 (0.02)**	0.16 (0.02)	0.17 (0.02)	0.18 (0.02)	0.16 (0.02)	0.18 (0.02)	0.19 (0.02)
SELFIES 25	Transformer (pdop)	1	0.4 (0.01)	**0.4 (0.01)**	0.39 (0.02)	0.37 (0.02)	0.35 (0.0)	0.36 (0.01)	0.35 (0.01)	0.35 (0.01)
	Transformer (rano)	1	0.14 (0.01)	**0.15 (0.01)**	0.16 (0.01)	0.16 (0.01)	0.15 (0.01)	0.13 (0.01)	0.14 (0.01)	0.16 (0.01)
	Transformer (zale)	1	0.36 (0.01)	**0.36 (0.01)**	0.36 (0.01)	0.36 (0.01)	0.3 (0.01)	0.29 (0.01)	0.29 (0.01)	0.3 (0.01)
SMILES	GRU	0.05	0.89 (0.19)	**0.82 (0.15)**	0.64 (0.2)	0.89 (0.14)	1.0 (0.14)	0.98 (0.19)	0.7 (0.24)	1.07 (0.19)
		0.1	0.4 (0.22)	**0.41 (0.19)**	0.63 (0.26)	0.55 (0.14)	0.83 (0.13)	0.96 (0.33)	0.69 (0.18)	0.75 (0.18)
		1	1.44 (0.0)	**-0.16 (0.0)**	–	–	0.14 (0.65)	–	–	1.14 (0.0)
	LSTM	0.05	1.18 (0.25)	**1.54 (0.41)**	1.1 (0.15)	1.1 (0.36)	0.86 (0.27)	1.05 (0.21)	1.07 (0.29)	1.22 (0.27)
		0.1	–	**–**	–	–	0.21 (0.0)	–	–	–
		1	0.61 (0.16)	**0.52 (0.26)**	0.48 (0.13)	0.7 (0.23)	0.67 (0.23)	1.13 (0.16)	0.97 (0.28)	0.88 (0.23)
	Transformer	0.05	0.88 (0.18)	**0.97 (0.19)**	0.92 (0.14)	0.67 (0.15)	0.93 (0.17)	1.16 (0.1)	1.09 (0.11)	0.97 (0.24)
		0.1	0.42 (0.12)	**0.62 (0.14)**	0.44 (0.16)	0.73 (0.14)	0.91 (0.1)	0.6 (0.14)	0.81 (0.14)	0.7 (0.14)
		1	0.72 (0.18)	**0.48 (0.18)**	0.61 (0.15)	0.47 (0.16)	0.99 (0.12)	0.74 (0.15)	0.83 (0.16)	0.68 (0.15)

Table 5: Ablation study for the initial length, success tolerance and failure tolerance for the TuRBO method. Average across 10 independent runs, of the best value across datasets, architectures. Column names indicate the length/success/fail values. Results from the main paper are in bold.

	Architecture	β	0.8/10/10	0.8/10/2	0.8/2/10	0.8/2/2	1.6/10/10	1.6/10/2	1.6/2/10	1.6/2/2
Expressions	GRU	0.05	-0.65 (0.11)	**-0.73 (0.12)**	-0.71 (0.09)	-0.61 (0.1)	-0.65 (0.12)	-0.59 (0.06)	-0.62 (0.11)	-0.72 (0.13)
		0.1	-0.56 (0.05)	**-0.61 (0.07)**	-0.59 (0.07)	-0.71 (0.1)	-0.54 (0.04)	-0.58 (0.05)	-0.62 (0.04)	-0.63 (0.04)
		1	-0.56 (0.05)	**-0.54 (0.05)**	-0.54 (0.06)	-0.6 (0.08)	-0.56 (0.06)	-0.61 (0.05)	-0.6 (0.04)	-0.59 (0.05)
	LSTM	0.05	-0.46 (0.03)	**-0.43 (0.02)**	-0.43 (0.02)	-0.38 (0.02)	-0.43 (0.02)	-0.43 (0.02)	-0.47 (0.04)	-0.4 (0.01)
		0.1	-0.38 (0.04)	**-0.39 (0.0)**	-0.41 (0.01)	-0.42 (0.02)	-0.42 (0.02)	-0.42 (0.02)	-0.42 (0.02)	-0.42 (0.02)
		1	-0.96 (0.09)	**-0.86 (0.0)**	-0.86 (0.0)	-1.01 (0.11)	-0.88 (0.01)	-0.98 (0.06)	-0.92 (0.04)	-0.88 (0.01)
	Transformer	0.05	-0.39 (0.04)	**-0.44 (0.02)**	-0.39 (0.04)	-0.4 (0.05)	-0.44 (0.02)	-0.39 (0.04)	-0.38 (0.04)	-0.42 (0.02)
		0.1	-0.38 (0.04)	**-0.41 (0.02)**	-0.42 (0.02)	-0.42 (0.02)	-0.39 (0.04)	-0.41 (0.01)	-0.41 (0.02)	-0.37 (0.04)
		1	-0.67 (0.1)	**-0.58 (0.1)**	-0.52 (0.08)	-0.62 (0.06)	-0.66 (0.11)	-0.62 (0.07)	-0.56 (0.05)	-0.54 (0.05)
SELFIES	Transformer (pdop)	0.05	0.13 (0.02)	**0.15 (0.03)**	0.17 (0.03)	0.18 (0.03)	0.23 (0.03)	0.17 (0.04)	0.22 (0.04)	0.2 (0.04)
		0.1	0.08 (0.02)	**0.09 (0.03)**	0.1 (0.03)	0.09 (0.03)	0.1 (0.03)	0.12 (0.04)	0.1 (0.03)	0.14 (0.04)
		1	0.36 (0.02)	**0.31 (0.03)**	0.36 (0.02)	0.34 (0.02)	0.38 (0.01)	0.38 (0.0)	0.4 (0.01)	0.36 (0.02)
	Transformer (rano)	0.05	0.17 (0.02)	**0.2 (0.02)**	0.2 (0.02)	0.18 (0.02)	0.11 (0.02)	0.08 (0.01)	0.12 (0.02)	0.13 (0.02)
		0.1	0.09 (0.02)	**0.1 (0.01)**	0.11 (0.02)	0.1 (0.02)	0.06 (0.01)	0.05 (0.01)	0.04 (0.01)	0.06 (0.01)
		1	0.07 (0.02)	**0.05 (0.02)**	0.11 (0.03)	0.07 (0.02)	0.16 (0.02)	0.16 (0.03)	0.11 (0.02)	0.12 (0.02)
	Transformer (zale)	0.05	0.11 (0.02)	**0.15 (0.03)**	0.12 (0.02)	0.16 (0.03)	0.22 (0.03)	0.23 (0.03)	0.21 (0.04)	0.19 (0.03)
		0.1	0.14 (0.01)	**0.16 (0.01)**	0.15 (0.02)	0.14 (0.01)	0.1 (0.02)	0.12 (0.02)	0.13 (0.02)	0.14 (0.03)
		1	0.36 (0.02)	**0.31 (0.03)**	0.38 (0.02)	0.36 (0.02)	0.38 (0.01)	0.34 (0.02)	0.31 (0.03)	0.38 (0.02)
SELFIES 25	Transformer (pdop)	1	0.5 (0.01)	**0.49 (0.02)**	0.49 (0.01)	0.47 (0.01)	0.45 (0.01)	0.46 (0.01)	0.45 (0.01)	0.46 (0.01)
	Transformer (rano)	1	0.34 (0.01)	**0.31 (0.02)**	0.32 (0.02)	0.32 (0.01)	0.31 (0.01)	0.32 (0.01)	0.33 (0.01)	0.33 (0.01)
	Transformer (zale)	1	0.47 (0.02)	**0.49 (0.01)**	0.48 (0.03)	0.47 (0.01)	0.42 (0.01)	0.4 (0.01)	0.41 (0.01)	0.43 (0.01)
SMILES	GRU	0.05	2.47 (0.22)	**2.47 (0.22)**	2.42 (0.21)	2.24 (0.17)	2.35 (0.23)	2.42 (0.28)	1.97 (0.18)	2.25 (0.31)
		0.1	1.76 (0.3)	**2.57 (0.31)**	2.45 (0.26)	2.07 (0.34)	2.24 (0.28)	2.26 (0.33)	2.24 (0.31)	1.92 (0.24)
		1	2.43 (0.32)	**2.48 (0.29)**	2.76 (0.24)	2.17 (0.49)	2.43 (0.32)	2.35 (0.37)	1.4 (0.52)	2.04 (0.33)
	LSTM	0.05	2.65 (0.32)	**2.73 (0.33)**	2.89 (0.26)	3.02 (0.34)	2.64 (0.3)	2.91 (0.29)	2.77 (0.23)	2.68 (0.37)
		0.1	1.97 (0.37)	**1.78 (0.43)**	1.98 (0.4)	2.16 (0.29)	2.36 (0.41)	1.87 (0.39)	2.34 (0.33)	1.73 (0.41)
		1	3.06 (0.2)	**2.71 (0.34)**	2.65 (0.16)	2.83 (0.25)	2.75 (0.28)	3.49 (0.26)	2.68 (0.31)	3.16 (0.35)
	Transformer	0.05	2.47 (0.23)	**2.88 (0.24)**	2.42 (0.17)	2.67 (0.27)	2.91 (0.3)	2.25 (0.2)	2.43 (0.24)	2.48 (0.29)
		0.1	3.03 (0.3)	**2.15 (0.18)**	2.51 (0.25)	2.41 (0.24)	2.28 (0.12)	2.28 (0.22)	2.53 (0.26)	2.46 (0.16)
		1	2.37 (0.21)	**2.25 (0.18)**	2.16 (0.17)	2.21 (0.25)	2.46 (0.2)	2.11 (0.12)	2.31 (0.16)	2.71 (0.32)

B ABLATION STUDIES

Table 6: Ablation study for the facet length parameter of LSO (L-BFGS) method. Average across 10 independent runs of the average top 20 values across datasets, architectures, and bound methods are displayed for facet lengths of size 1, 5 and 10. Results from the main paper are bold.

	Architecture	β	1	5	10
Expressions	GRU	0.05	-1.79 (0.08)	**-1.72 (0.07)**	-1.72 (0.07)
		0.1	-1.73 (0.11)	**-1.93 (0.09)**	-1.89 (0.11)
		1	-1.94 (0.07)	**-1.97 (0.07)**	-2.03 (0.09)
	LSTM	0.05	-1.89 (0.09)	**-1.78 (0.09)**	-1.93 (0.06)
		0.1	-1.29 (0.06)	**-1.39 (0.08)**	-1.37 (0.06)
		1	-2.04 (0.05)	**-2.04 (0.04)**	-2.04 (0.05)
	Transformer	0.05	-3.11 (0.14)	**-2.93 (0.13)**	-3.02 (0.09)
		0.1	-3.19 (0.28)	**-2.69 (0.25)**	-2.93 (0.22)
		1	-2.44 (0.11)	**-2.41 (0.09)**	-2.28 (0.11)
SELFIES	Transformer (pdop)	0.05	0.22 (0.01)	**0.21 (0.01)**	0.25 (0.01)
		0.1	0.25 (0.01)	**0.19 (0.01)**	0.22 (0.03)
		1	0.26 (0.01)	**0.25 (0.01)**	0.27 (0.01)
	Transformer (rano)	0.05	0.04 (0.0)	**0.03 (0.0)**	–
		0.1	–	**–**	–
		1	0.09 (0.02)	**0.07 (0.0)**	0.08 (0.01)
	Transformer (zale)	0.05	0.12 (0.02)	**0.13 (0.01)**	0.11 (0.01)
		0.1	0.08 (0.01)	**0.06 (0.0)**	0.07 (0.0)
		1	0.18 (0.02)	**0.18 (0.02)**	0.19 (0.02)
SELFIES 25	Transformer (pdop)	1	0.33 (0.01)	**0.33 (0.01)**	0.34 (0.01)
	Transformer (rano)	1	0.15 (0.01)	**0.12 (0.01)**	0.14 (0.01)
	Transformer (zale)	1	0.29 (0.01)	**0.29 (0.01)**	0.27 (0.01)
SMILES	GRU	0.05	0.65 (0.12)	**0.52 (0.12)**	0.52 (0.13)
		0.1	0.52 (0.14)	**0.12 (0.14)**	0.36 (0.15)
		1	–	**–**	–
	LSTM	0.05	0.67 (0.15)	**0.66 (0.12)**	0.56 (0.2)
		0.1	–	**–**	–
		1	0.49 (0.16)	**0.7 (0.2)**	0.55 (0.15)
	Transformer	0.05	0.52 (0.18)	**0.42 (0.15)**	0.55 (0.12)
		0.1	0.63 (0.18)	**0.61 (0.15)**	0.56 (0.17)
		1	0.16 (0.15)	**0.23 (0.12)**	0.39 (0.13)

Table 7: Ablation study for the facet length parameter of LSO (L-BFGS) method. Average across 10 independent runs, of the best value across datasets, architectures, and bound methods are displayed for facet lengths of size 1, 5 and 10. Results from the main paper are bold.

	Architecture	β	1	5	10
Expressions	GRU	0.05	-0.57 (0.07)	**-0.56 (0.04)**	-0.56 (0.07)
		0.1	-0.53 (0.04)	**-0.46 (0.02)**	-0.46 (0.02)
		1	-0.68 (0.05)	**-0.77 (0.02)**	-0.73 (0.04)
	LSTM	0.05	-0.57 (0.11)	**-0.56 (0.04)**	-0.67 (0.09)
		0.1	-0.4 (0.05)	**-0.44 (0.04)**	-0.47 (0.04)
		1	-1.01 (0.1)	**-1.02 (0.07)**	-0.86 (0.0)
	Transformer	0.05	-1.06 (0.13)	**-0.8 (0.1)**	-1.11 (0.18)
		0.1	-0.8 (0.14)	**-0.65 (0.1)**	-0.69 (0.09)
		1	-0.85 (0.1)	**-0.82 (0.1)**	-0.78 (0.12)
SELFIES	Transformer (pdop)	0.05	0.36 (0.01)	**0.36 (0.02)**	0.34 (0.03)
		0.1	0.29 (0.03)	**0.34 (0.02)**	0.27 (0.03)
		1	0.39 (0.01)	**0.36 (0.02)**	0.35 (0.03)
	Transformer (rano)	0.05	0.06 (0.01)	**0.07 (0.01)**	0.08 (0.01)
		0.1	0.08 (0.02)	**0.08 (0.02)**	0.07 (0.01)
		1	0.16 (0.02)	**0.16 (0.02)**	0.16 (0.02)
	Transformer (zale)	0.05	0.27 (0.03)	**0.27 (0.03)**	0.26 (0.03)
		0.1	0.15 (0.04)	**0.19 (0.03)**	0.18 (0.04)
		1	0.38 (0.01)	**0.38 (0.02)**	0.39 (0.01)
SELFIES 25	Transformer (pdop)	1	0.46 (0.01)	**0.43 (0.01)**	0.46 (0.01)
	Transformer (rano)	1	0.31 (0.02)	**0.3 (0.01)**	0.31 (0.02)
	Transformer (zale)	1	0.41 (0.0)	**0.4 (0.01)**	0.41 (0.02)
SMILES	GRU	0.05	2.06 (0.3)	**1.71 (0.24)**	2.02 (0.22)
		0.1	2.11 (0.28)	**1.74 (0.2)**	1.94 (0.22)
		1	2.4 (0.31)	**2.1 (0.32)**	2.34 (0.31)
	LSTM	0.05	2.31 (0.21)	**2.32 (0.19)**	2.15 (0.18)
		0.1	1.74 (0.5)	**1.85 (0.34)**	1.47 (0.49)
		1	2.8 (0.25)	**3.09 (0.16)**	2.42 (0.3)
	Transformer	0.05	1.82 (0.2)	**1.79 (0.13)**	1.74 (0.17)
		0.1	2.19 (0.16)	**2.24 (0.1)**	2.09 (0.17)
		1	1.72 (0.18)	**2.0 (0.14)**	1.96 (0.14)

C ADDITIONAL EXPERIMENTAL RESULTS

Table 8: Average of the top solution found during LSO (higher is better), across datasets and decoder architectures. We **bold** the best method and underline the second-best. The average ranking for each method (lower is better) is provided, along with the number of times each method is within one standard deviation of the best. LES and Prior achieve the highest value most frequently (14 out of 30) and outperforms other methods both in terms of the average ranking and the frequency of being within one standard deviation of the best result.

	Architecture	β	LES	LSO (L-BFGS)	UC	LSO (GA)	Prior	TuRBO	Likelihood
Expressions	GRU	0.05	-0.55 (0.04)	-0.56 (0.04)	-0.59 (0.04)	-0.45 (0.05)	**-0.4 (0.07)**	-0.73 (0.12)	**-0.4 (0.07)**
		0.1	-0.45 (0.03)	-0.46 (0.02)	-0.47 (0.05)	-0.43 (0.02)	**-0.37 (0.03)**	-0.61 (0.07)	-0.43 (0.02)
		1	-0.47 (0.03)	-0.77 (0.02)	-0.51 (0.04)	-0.46 (0.02)	-0.47 (0.03)	-0.54 (0.05)	**-0.43 (0.01)**
	LSTM	0.05	-0.43 (0.02)	-0.56 (0.04)	-0.52 (0.05)	-0.43 (0.01)	-0.41 (0.01)	-0.43 (0.02)	**-0.4 (0.01)**
		0.1	**-0.32 (0.05)**	-0.44 (0.04)	-0.39 (0.04)	-0.38 (0.02)	-0.4 (0.01)	-0.39 (0.0)	**-0.32 (0.04)**
		1	**-0.86 (0.0)**	-1.02 (0.07)	**-0.86 (0.0)**	**-0.86 (0.0)**	**-0.86 (0.0)**	**-0.86 (0.0)**	-0.91 (0.04)
	Transformer	0.05	-0.43 (0.03)	-0.8 (0.1)	-0.57 (0.05)	-0.44 (0.02)	**-0.37 (0.05)**	-0.44 (0.02)	-0.38 (0.04)
		0.1	-0.36 (0.03)	-0.65 (0.1)	-0.55 (0.04)	-0.39 (0.01)	**-0.35 (0.04)**	-0.41 (0.02)	-0.41 (0.02)
		1	**-0.52 (0.05)**	-0.82 (0.1)	-0.58 (0.05)	-0.58 (0.04)	-0.62 (0.09)	-0.58 (0.1)	-0.65 (0.08)
SELFIES	Transformer (pdop)	0.05	**0.43 (0.0)**	0.36 (0.02)	0.42 (0.0)	0.42 (0.0)	0.42 (0.0)	0.15 (0.03)	**0.43 (0.0)**
		0.1	**0.43 (0.0)**	0.34 (0.02)	0.42 (0.01)	0.42 (0.0)	0.41 (0.01)	0.09 (0.03)	**0.43 (0.01)**
		1	**0.41 (0.01)**	0.36 (0.02)	0.39 (0.01)	0.4 (0.0)	0.38 (0.01)	0.31 (0.03)	0.4 (0.0)
	Transformer (rano)	0.05	0.33 (0.01)	0.07 (0.01)	**0.36 (0.01)**	0.33 (0.01)	0.32 (0.01)	0.2 (0.02)	0.31 (0.01)
		0.1	0.33 (0.01)	0.08 (0.02)	**0.36 (0.02)**	0.32 (0.01)	0.32 (0.01)	0.1 (0.01)	0.33 (0.01)
		1	0.31 (0.01)	0.16 (0.02)	**0.39 (0.02)**	0.31 (0.01)	0.33 (0.02)	0.05 (0.02)	0.33 (0.01)
	Transformer (zale)	0.05	0.43 (0.01)	0.27 (0.03)	0.42 (0.01)	0.43 (0.01)	0.42 (0.0)	0.15 (0.03)	**0.44 (0.01)**
		0.1	**0.44 (0.01)**	0.19 (0.03)	0.42 (0.01)	0.42 (0.01)	0.42 (0.01)	0.16 (0.01)	**0.44 (0.01)**
		1	**0.42 (0.01)**	0.38 (0.02)	0.39 (0.01)	**0.42 (0.01)**	0.37 (0.01)	0.31 (0.03)	**0.42 (0.01)**
SELFIES 25	Transformer (pdop)	1	0.48 (0.01)	0.43 (0.01)	0.46 (0.01)	0.47 (0.01)	**0.51 (0.02)**	0.49 (0.02)	0.47 (0.01)
	Transformer (rano)	1	0.37 (0.01)	0.3 (0.01)	0.37 (0.01)	**0.38 (0.01)**	0.36 (0.01)	0.31 (0.02)	0.36 (0.01)
	Transformer (zale)	1	0.41 (0.02)	0.4 (0.01)	0.41 (0.01)	0.47 (0.02)	0.41 (0.01)	**0.49 (0.01)**	0.41 (0.02)
SMILES	GRU	0.05	**3.29 (0.1)**	1.71 (0.24)	3.13 (0.07)	3.26 (0.11)	3.18 (0.06)	2.47 (0.22)	3.26 (0.08)
		0.1	**3.55 (0.14)**	1.74 (0.2)	3.2 (0.1)	3.31 (0.16)	3.15 (0.11)	2.57 (0.31)	3.33 (0.12)
		1	3.85 (0.17)	2.1 (0.32)	2.24 (0.28)	3.66 (0.16)	**3.89 (0.28)**	2.48 (0.29)	**3.89 (0.2)**
	LSTM	0.05	3.29 (0.07)	2.32 (0.19)	3.12 (0.08)	3.3 (0.1)	3.28 (0.1)	2.73 (0.33)	**3.37 (0.09)**
		0.1	**3.66 (0.2)**	1.85 (0.34)	2.65 (0.19)	3.52 (0.22)	3.57 (0.18)	1.78 (0.43)	3.54 (0.16)
		1	**3.6 (0.14)**	3.09 (0.16)	2.6 (0.3)	3.18 (0.17)	3.28 (0.17)	2.71 (0.34)	3.48 (0.11)
	Transformer	0.05	**3.21 (0.08)**	1.79 (0.13)	3.1 (0.07)	3.14 (0.04)	3.14 (0.04)	2.88 (0.24)	3.19 (0.08)
		0.1	3.23 (0.04)	2.24 (0.1)	**3.28 (0.08)**	3.11 (0.05)	3.09 (0.06)	2.15 (0.18)	3.16 (0.06)
		1	**3.2 (0.07)**	2.0 (0.14)	2.8 (0.13)	3.13 (0.07)	3.11 (0.1)	2.25 (0.18)	**3.2 (0.06)**
	Average rank		**1.97**	6.47	4.1	3.13	3.63	5.77	2.93
	# within 1 std of best		**18**	0	7	9	11	4	16

Table 9: Effect of increasing λ parameter for LES, for the expressions dataset. Increasing the value of the parameter λ increases the percentage of valid solution in all cases.

Architecture	β	**LES** ($\lambda = 0.05$)	**LES** ($\lambda = 0.5$)
GRU	**0.05**	0.92	0.96
	0.1	0.7	0.82
	1	0.72	0.87
LSTM	**0.05**	0.93	0.98
	0.1	0.93	0.98
	1	0.76	0.97
Transformer	**0.05**	0.87	0.94
	0.1	0.88	0.96
	1	0.83	0.85

C.1 Comparing Likelihood and LES for density estimation

Here's a smoothed version:

To clarify the differences between Likelihood and LES, we analyze Gaussian vectors ($\boldsymbol{z}$) of varying dimensions ($d = 25$, $d = 56$, $d = 75$, and $d = 256$) after applying the softmax transformation to obtain probability vectors ($\boldsymbol{x}$). For each data point, we compute LES, the likelihood score, and the **true** density of $\boldsymbol{x}$ (calculated analytically using the change-of-variables formula) under the softmax transformation (i.e., $\boldsymbol{x} = \text{softmax}(\boldsymbol{z})$).

To visualize these differences, we uniformly sample data points between -20 and 20 along the first dimension of $\boldsymbol{z}$, while sampling the remaining dimensions from a Gaussian distribution with a standard deviation of 0.1.

The results, shown in fig. 3, clearly demonstrate that LES provides a more accurate estimate of the **true** density of $\boldsymbol{x}$. In contrast, the likelihood score fails to capture the true density's correct structure.

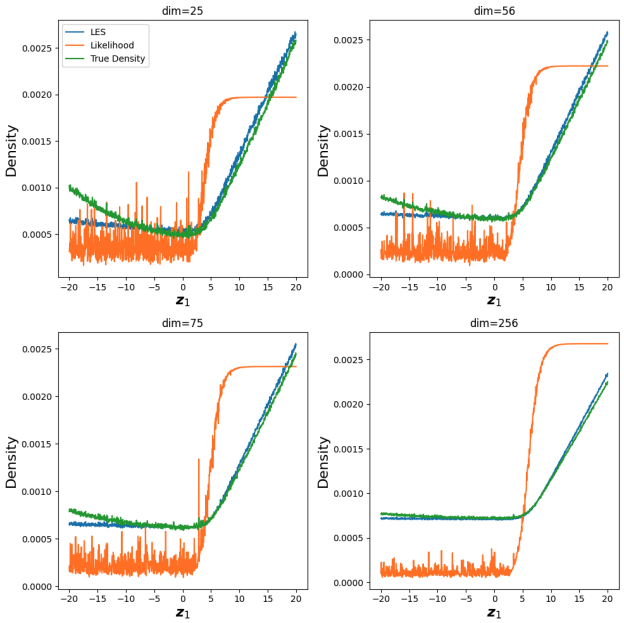

Figure 3: Normalized density values (divided by the sum) for LES, Likelihood along with the true density of $\boldsymbol{x} = \text{softmax}(\boldsymbol{z})$. Values of the density are visualized across the first dimension of $\boldsymbol{z}$, sampled along an evenly spaced grid between -20 and 20. Across the four dimentions LES provides a better estimate of the true likelihood of $\boldsymbol{x}$.

Table 10: Proportion of valid solutions found during LSO (higher is better) across datasets and decoder architectures. We bold the best method (higher is better) and underline the second best. LES improves the validity of the solutions compared with LSO (GA) (which is LES with $\lambda = 0$) across all datasets.

	Architecture	β	LES	LSO (L-BFGS)	UC	LSO (GA)	Prior	TuRBO	Likelihood
Expressions	GRU	0.05	0.92	0.59	**1.0**	0.91	0.91	0.94	0.89
		0.1	0.7	0.57	**1.0**	0.64	0.66	0.9	0.67
		1	0.72	0.45	**0.99**	0.69	0.69	0.83	0.69
	LSTM	0.05	0.93	0.62	**1.0**	0.88	0.89	0.94	0.92
		0.1	0.93	0.67	**1.0**	0.9	0.89	0.94	0.92
		1	0.76	0.58	**0.99**	0.65	0.66	0.89	0.73
	Transformer	0.05	0.87	0.37	**1.0**	0.83	0.85	0.9	0.84
		0.1	0.88	0.28	**1.0**	0.8	0.81	0.89	0.85
		1	0.83	0.36	**1.0**	0.74	0.76	0.87	0.79
SELFIES	Transformer (pdop)	0.05	0.8	0.05	**0.81**	0.76	0.73	0.14	0.77
		0.1	0.68	0.04	**0.71**	0.62	0.56	0.14	0.64
		1	**0.59**	0.08	0.55	0.49	0.44	0.08	0.53
	Transformer (rano)	0.05	0.66	0.02	**0.73**	0.61	0.57	0.09	0.62
		0.1	0.57	0.01	**0.67**	0.52	0.44	0.04	0.53
		1	0.43	0.02	**0.49**	0.36	0.28	0.01	0.39
	Transformer (zale)	0.05	0.71	0.08	**0.74**	0.66	0.62	0.12	0.69
		0.1	0.66	0.01	**0.67**	0.59	0.51	0.08	0.63
		1	**0.54**	0.18	0.47	0.43	0.35	0.17	0.46
SELFIES 25	Transformer (pdop)	1	0.47	0.23	0.41	0.41	0.42	0.24	**0.48**
	Transformer (rano)	1	0.37	0.14	0.28	0.27	0.28	0.11	**0.42**
	Transformer (zale)	1	0.53	0.48	0.5	0.52	0.51	**0.54**	0.49
SMILES	GRU	0.05	**0.61**	0.14	0.48	0.47	0.44	0.12	0.59
		0.1	**0.37**	0.07	0.25	0.23	0.22	0.06	0.35
		1	**0.09**	0.02	0.05	0.05	0.04	0.02	0.08
	LSTM	0.05	**0.6**	0.11	0.44	0.42	0.39	0.11	0.57
		0.1	**0.08**	0.01	0.06	0.05	0.04	0.01	0.07
		1	**0.16**	0.09	0.07	0.08	0.07	0.09	0.12
	Transformer	0.05	0.7	0.42	0.7	0.68	0.68	0.31	**0.72**
		0.1	0.65	**0.67**	0.63	0.6	0.55	0.41	0.64
		1	**0.48**	0.35	0.46	0.34	0.26	0.3	0.45
	Average		0.61	0.26	**0.63**	0.54	0.52	0.37	0.58

D BACKGROUND ON RELATED WORK

Bayesian uncertainty Notin et al. (2021) Under a Bayesian viewpoint, the trained neural network parameters ($\boldsymbol{\theta}$) follow a variational distribution, which we can sample from using MC-Dropout Gal & Ghahramani (2016). Based on this distribution, the uncertainty is defined as

$$\mathcal{M}(\boldsymbol{z}) = \mathcal{H}_p(p(\boldsymbol{X}|\boldsymbol{Z}=\boldsymbol{z})) - \mathbb{E}_{\boldsymbol{\theta}}\mathcal{H}_{p_{\boldsymbol{\theta}}}(p_{\boldsymbol{\theta}}(\boldsymbol{X}|\boldsymbol{Z}=\boldsymbol{z})), \tag{37}$$

where $\mathcal{H}$ is the entropy and $p(\boldsymbol{X}|\boldsymbol{Z}=\boldsymbol{z})$ is the posterior predictive distribution. The uncertainty is estimated using MC-Dropout with important sampling (using the posteior predictive as the importance distribution) designed to approximate the expectations over the random variable $\boldsymbol{X}$, as it is typically a very large space (i.e., the sample of molecules that can implemented using a SMILE string with 120 characters)