# Plug & play directed evolution of proteins with gradient-based discrete MCMC

IOP Publishing Mach. Learn.: Sci. Technol. 4 (2023) 025014 https://doi.org/10.1088/2632-2153/accacd

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
16 December 2022

REVISED
16 March 2023

ACCEPTED FOR PUBLICATION
29 March 2023

PUBLISHED
25 April 2023



PAPER

# Plug & play directed evolution of proteins with gradient-based discrete MCMC

**Patrick Emami**[1,*], **Aidan Perreault**[2], **Jeffrey Law**[1], **David Biagioni**[3] and **Peter St. John**[1]

[1] National Renewable Energy Lab, Golden, CO, United States of America
[2] Stanford University, Stanford, CA, United States of America
[3] Maplewell Energy, Broomfield, CO, United States of America
[*] Author to whom any correspondence should be addressed.

**E-mail:** pemami@nrel.gov

**Keywords:** discrete MCMC, directed evolution, unsupervised learning, protein language models, protein engineering, biological sequence design

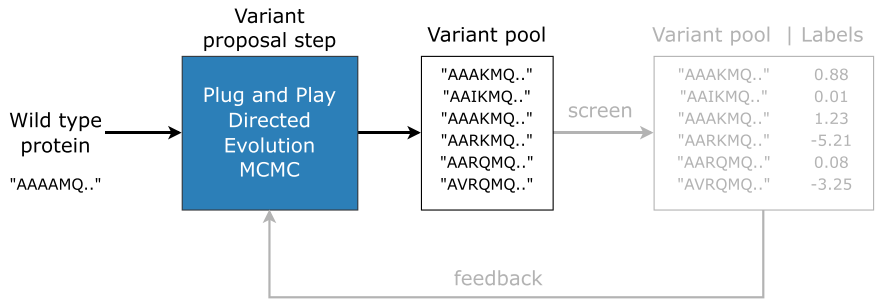
## Abstract

A long-standing goal of machine-learning-based protein engineering is to accelerate the discovery of novel mutations that improve the function of a known protein. We introduce a sampling framework for evolving proteins *in silico* that supports mixing and matching a variety of unsupervised models, such as protein language models, and supervised models that predict protein function from sequence. By composing these models, we aim to improve our ability to evaluate unseen mutations and constrain search to regions of sequence space likely to contain functional proteins. Our framework achieves this without any model fine-tuning or re-training by constructing a product of experts distribution directly in discrete protein space. Instead of resorting to brute force search or random sampling, which is typical of classic directed evolution, we introduce a fast Markov chain Monte Carlo sampler that uses gradients to propose promising mutations. We conduct *in silico* directed evolution experiments on wide fitness landscapes and across a range of different pre-trained unsupervised models, including a 650 M parameter protein language model. Our results demonstrate an ability to efficiently discover variants with high evolutionary likelihood as well as estimated activity multiple mutations away from a wild type protein, suggesting our sampler provides a practical and effective new paradigm for machine-learning-based protein engineering.

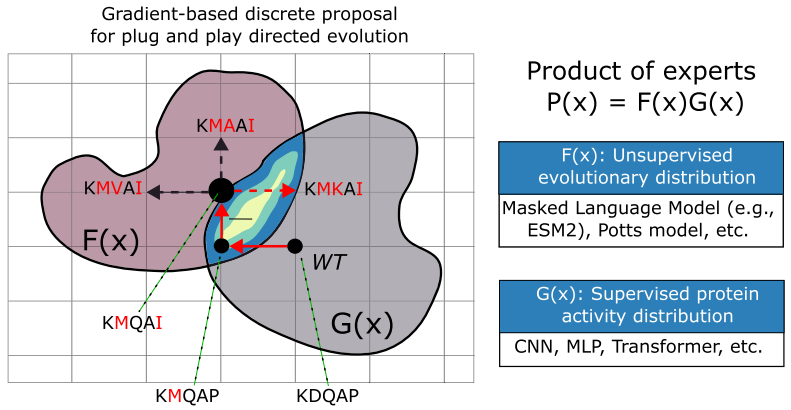
## 1. Introduction

Engineering proteins to improve their productivity or catalyze new reactions requires scientists to navigate the complex landscape mapping a protein's amino acid sequence to its structure and function (Li *et al* 2020). *Directed evolution* is a classic approach inspired by natural evolution where random mutations to a protein's sequence are screened in a wet lab until higher-performing variants are found, at which point the process repeats starting from these variants (Kuchner and Arnold 1997). However, this becomes impractical when proteins cannot be assessed in a high-throughput fashion. The simplest approach, brute force search, is limited in practice to variants with one or two mutations. For a protein with 400 amino acids, there are $\sim 10^{19}$ ways to make five single substitutions assuming the standard vocabulary of 20 amino acids. It is also remarkably difficult to find proteins with improved function. Most of protein space is non-functional and beneficial mutations are rare (Arnold 1998).

As supervised machine learning (ML) methods for protein function prediction from primary sequence improve (Dallago *et al* 2021, Hsu *et al* 2022), *machine-learning-based directed evolution* has emerged, which offers a way to judiciously propose candidates for screening that ultimately reduces time spent in the wet lab (Yang *et al* 2019, Biswas *et al* 2021, Wu *et al* 2021). Our work is concerned with improving the variant proposal step by increasing the chance of discovering a mutation that improves a target property of a known 'wild type' (WT) protein (figure 1(a)). We argue that the promising performance of recent unsupervised




(a)




(b)

**Figure 1.** (a) Illustration of the machine-learning-based directed evolution pipeline. This paper proposes a plug and play sampler for proposing variants of improved fitness near a wild-type (WT) protein for each design round. (b) We flexibly compose unsupervised and supervised protein sequence models *as a product of experts* distribution and derive a fast gradient-based discrete MCMC sampler to efficiently sample from this distribution. A product of experts places its probability mass at the *intersection* of the expert distributions. The key idea of gradient-based discrete MCMC is to bias the mutation proposal distribution towards promising mutations within a local neighborhood around the current variant. Then, to avoid enumerating all mutations in this neighborhood, the proposal is approximated with a first-order Taylor series. An example MCMC trajectory of single point mutations starting from the WT is shown.

sequence models at mutation effect prediction (Meier *et al* 2021, Hsu *et al* 2022, Weinstein *et al* 2022) has motivated a reconsideration of simple black-box optimization algorithms for searching sequence space. Black-box algorithms are appealing due to their simplicity, flexibility, compatibility with discrete spaces, and use of interpretable mutation operators. They offer a way to mix and match unsupervised and supervised models for proposing variants without requiring any model fine-tuning or re-training, i.e. in a 'plug and play' manner. Combining both types of models is generally advantageous. While unsupervised models learn information that can steer search away from adversarial inputs which fool supervised models due to overestimation errors (Szegedy *et al* 2014), supervised models learn specific information about beneficial mutations gleaned from assay-labeled data. However, black-box algorithms tend to be extremely inefficient at searching for variants with improved fitness. A practical plug and play framework for searching protein sequence space has remained elusive due to the difficulty of fast search in discrete, high-dimensional spaces.

This paper fills that gap by introducing plug and play directed evolution (PPDE), a practical plug and play sampling framework for the efficient discovery of functional variants directly in discrete protein space. PPDE flexibly combines unsupervised and supervised models with a *product of experts* distribution (Hinton 2002) $p(x) = \prod_i p_i(x)$. Combining both types of protein models in this manner encourages variants to have both high sequence likelihood and high predicted function (e.g. activity). To sample efficiently from the high-dimensional, discrete, and unnormalized distribution $p(x)$, we derive a fast Markov chain Monte Carlo (MCMC) sampler that uses *gradients* of $p(x)$ to propose mutations. See figure 1(b) for a visual depiction the sampler. PPDE nearly maintains all of the flexibility of black-box algorithms—it only has to additionally assume that $p(x)$ is differentiable at each discrete point of protein space. We theoretically characterize the efficiency of this sampler and empirically show that our framework works with a variety of pre-trained models without using continuous relaxations and without model retraining or fine-tuning. Although our formulation can be applied to a broad class of biological sequence design tasks, we focus on proteins due to the current widespread availability of pre-trained models.

A summary of our contributions:

- We introduce a plug and play sampling framework that enables mixing and matching various unsupervised and supervised protein models without model re-training or fine-tuning.
- We demonstrate a novel application of gradient-based discrete MCMC for conducting search with a product of experts distribution.
- We conduct rigorous *in silico* directed evolution experiments on three target proteins with partially characterized wide ($\geqslant$2 mutation) fitness landscapes. Our results provide strong evidence that our sampler offers a practical and effective approach for ML-based directed evolution beyond brute force and random search.

## 2. Related work

**Black-box approaches to directed evolution:** A popular paradigm for ML-based directed evolution has been to learn a surrogate for the sequence-to-function landscape from a small labeled dataset and then to use a black-box optimizer or sampler such as an evolutionary algorithm or simulated annealing to search for mutants (Hansen and Ostermeier 1996, Angermüller *et al* 2020, Sinai *et al* 2020, Biswas *et al* 2021). Black-box approaches are plug and play in that they can easily combine unsupervised and supervised protein models without any model fine-tuning or re-training. Navigating the high-dimensional discrete space is approached by either using a random walk, which is highly inefficient, or by using gradient ascent via a continuous relaxation of the space, which can bias the search process towards poor local optima. These are the main baselines for our method and will be described in more detail in section 5. Reinforcement learning (RL) has also been used as a black-box optimization framework for training autoregressive models to generate high-fitness variants (Angermueller *et al* 2019). However, this approach has less flexibility than plug and play methods, in that it does not support easy mixing of unsupervised pre-trained models with the learned RL policy.

**Protein design by generative modeling:** An alternative to directed evolution is to cast protein design as inference, i.e. fitting a conditional generative model to assay-labeled proteins showing high activity (Brookes and Listgarten 2018, Gupta and Zou 2018, Brookes *et al* 2019, Fannjiang and Listgarten 2020, Jain *et al* 2022, Zhang *et al* 2022a). Since high performing variants are rare, this task amounts to density estimation of a rare event which is a difficult statistical inference problem in its own right. Another popular alternative is latent space optimization, in which gradient ascent is performed on a surrogate function trained directly in the latent space of a generative model. These generative models use regularization to shape their latent space favorably so that optimization remains constrained to the manifold of viable proteins (Killoran *et al* 2017, Gomez-Bombarelli *et al* 2018, Kumar and Levine 2020, Linder *et al* 2020, Chan *et al* 2021, Trabucco *et al* 2021, Castro *et al* 2022). The deep manifold sampler (Gligorijevic *et al* 2021) more explicitly constrains optimization by alternating between steps of gradient ascent and re-projection of a noised version of the sequence onto the latent space of a custom denoising autoencoder (DAE). Our plug and play approach works with a variety of unsupervised models and can combine multiple such models if desired.

At the expense of trust in candidate proteins, unconditional generative models can also be used to hallucinate proteins from distant, unexplored regions of protein space. For example, training a generative model on sequences from a target protein family has been used to generate functional variants (Costello and Martin 2019, Hawkins-Hooker *et al* 2021, Shin *et al* 2021). Unconditional sampling from protein language models trained on unaligned (Madani *et al* 2020, Rives *et al* 2021, Ferruz *et al* 2022, Hesslow *et al* 2022, Lin *et al* 2022, Nijkamp *et al* 2022, Yang *et al* 2022) and aligned (Rao *et al* 2021, Notin *et al* 2022) sequences has recently been explored. While this is a promising new direction for protein engineering, our focus is on directed evolution from known proteins.

**Plug and play text generation:** Controlled generation of text has similarly been approached by combining pre-trained unsupervised models (e.g. large language models) and supervised models (e.g. sentiment classifiers) as a discrete product of experts (Holtzman *et al* 2018, Dathathri *et al* 2020, Qin *et al* 2022). Unlike controlled text generation, where the goal is to guide a sampling process to transform a sentence by altering entire words and phrases, *directed evolution seeks to accumulate a few precise changes to an initial sequence by modifying a small number of amino acids* (i.e. individual characters). Moreover, plug and play approaches for controlled text generation intentionally avoid solving a combinatorial search problem by instead restricting the class of suitable unsupervised experts to left-to-right autoregressive language models. This enables the use of sequential sampling algorithms such as left-to-right beam search decoding. Our use of gradient-based discrete MCMC for PPDE allows us to efficiently search the combinatorial space of mutations near the WT protein and to use a more general class of unsupervised experts that includes *orderless* protein sequence models which capture epistatic relationships between any pair of amino acids. The Metropolis-adjusted Langevin algorithm (MALA) used for *plug and play image generation* (Nguyen *et al* 2017) is a simple

gradient-based MCMC sampler for continuous spaces, and is closely related to locally-balanced discrete MCMC (Sun *et al* 2022a, Zhang *et al* 2022b). Therefore, we use MALA with a hand-crafted continuous relaxation as a baseline for plug and play sampling from discrete product of experts in our experiments.

## 3. Background

Before presenting our sampler, we define our search problem, briefly review Metropolis–Hastings (MH) MCMC, and introduce gradient-based discrete MCMC, a class of MH samplers able to efficiently explore high-dimensional discrete distributions.

**Problem definition:** The variant proposal step of directed evolution involves searching for mutations that improve one or more target properties of a given WT protein. This search problem is formally defined over discrete protein sequences $x := \{x_0, \ldots, x_{L-1}\}$, $x \in X$, of length $L$ with each $x_i$ taking on a value in a vocab of size $V$ (typically $V = 20$ for the 20 standard amino acids). We assume that each $x_i$ is one-hot encoded. The search is initialized at the WT protein $x^{\text{WT}}$ and terminates when a predefined condition is met (e.g. a maximum allowable number of search steps is reached).

**MH:** The MH algorithm (Metropolis *et al* 1953, Hastings 1970) defines the following algorithm for drawing samples from a distribution over protein variants $p(x)$. Let $f(x)$ be the unnormalized log probability of $x$ such that $\log p(x) = f(x) - \log Z$ where $Z = \sum_{x \in X} \exp(f(x))$ is a normalizing constant. Given the current state $x$, draw a candidate next state $x'$ from proposal distribution $q(x' \mid x)$. Accept the proposed state with probability

$$\min\left\{1, \exp(f(x') - f(x))\frac{q(x|x')}{q(x'|x)}\right\},$$

otherwise reject the transition and stay at $x$. This probabilistic acceptance/rejection criterion is desirable since it provides MH samplers with theoretical convergence guarantees and does not contain any hyperparameters, simplifying the implementation of MH in practice.

**Gradient-based discrete MCMC:** Uninformed MH proposals such as the uniform distribution are often inefficient for sampling from high-dimensional discrete distributions since candidate states $x'$ are proposed 'blindly'. In general, the efficiency of MH is highly dependent on the choice of proposal distribution. MH with *locally-balanced informed proposals* (Zanella 2020) use distributions of the form

$$q(x'|x) \propto \exp(f(x') - f(x))^{\frac{1}{2}} \mathbf{1}(x' \in \mathcal{N}(x)), \tag{1}$$

where $\mathbf{1}(x' \in \mathcal{N}(x))$ is an indicator function. The key idea is to bias the proposal distribution towards local state transitions within a neighborhood $\mathcal{N}(x)$ that incur an increase in likelihood. We take the square root of the exponential in this proposal as the choice of local balancing function $w(t) = tw(1/t), \forall t > 0$. This function 'balances' the acceptance and rejection probabilities in the local neighborhood $\mathcal{N}(x)$ to achieve a high acceptance rate. The square root $w(t) = \sqrt{t}$ was empirically validated as a good default option in Zanella (2020). Since enumerating all local moves in $\mathcal{N}(x)$ in a discrete high-dimensional spaces is infeasible, an efficient alternative is available for functions $f(x)$ whose gradient can be evaluated at the discrete state $x$ (Grathwohl *et al* 2021). A *gradient-based* locally-balanced informed proposal is the first-order Taylor-series approximation of equation (1) around $x$:

$$\tilde{q}(x'|x) \propto \exp\left(\frac{1}{2}\nabla_x f(x)^T (x' - x)\right)\mathbf{1}(x' \in \mathcal{N}(x)). \tag{2}$$

When $\mathcal{N}(x)$ is the 1-Hamming ball, this proposal amounts to a tempered softmax over single changes to one dimension of $x$. One forward pass and one backwards pass is required to compute the forward $\tilde{q}(x'|x)$ and reverse $\tilde{q}(x|x')$ approximate proposals.

A concern with this sampler is that it is susceptible to getting trapped in poor local optima since it cannot perform large jumps across the search space at each step of MCMC (Sun *et al* 2022b). One way to address this is to increase the Hamming window size, $U$, to $U > 1$. While this enables the sampler to propose changes to multiple dimensions of $x$ simultaneously, it also incurs a large increase in computation since $\mathcal{N}(x)$ now contains $O((LV)^U)$ states to evaluate. Alternatively, a *path proposal* sequentially samples $R \sim \text{Unif}(1, U)$ single changes to apply to the current state $x$ (Sun *et al* 2022b). For example, if $x$ is the sequence 'AAA', then a sampled path of length 3 could look like 'AAA' $\rightarrow$ 'AAB' $\rightarrow$ 'AAC' $\rightarrow$ 'ABC'.

The approximate path proposal distribution starting at $x^0 := x$ is

$$\tilde{q}_R(x'|x) = \prod_{r=1}^{R} \tilde{q}(x^r|x^{r-1}) \propto \prod_{r=1}^{R} \exp\left(\frac{1}{2}\nabla_{x^0} f(x^0)^T (x^r - x^{r-1})\right)\mathbf{1}(x^r \in \mathcal{N}(x^{r-1})). \tag{3}$$

The $r$th path state $x^r$ is sampled from $\tilde{q}\left(x^r|x^{r-1}\right)$. The sampler decides whether to accept or reject the terminal path state $x' := x^R$ (e.g. 'ABC'), i.e. the accumulation of all $R$ changes applied to $x$. The reverse proposal $\tilde{q}_R(x|x')$ is computed using the gradient at the terminal state $\nabla_{x'} f(x')$ and the reversed sequence of states $(x^R, x^{R-1}, \ldots, x^0)$. The number of forward and backwards passes required to compute the forward and reverse path proposals is still two. In the next section, we will introduce and characterize a gradient-based path proposal for product of experts distributions.

## 4. PPDE

Our approach for proposing mutations that improve a target property of the WT protein is to sample from a distribution constructed by combining unsupervised and supervised sequence models without using any continuous relaxations, model re-training, or fine-tuning. This distribution places its probability mass on proteins that have both high *evolutionary density* $f(x)$ (i.e. high likelihood of being a naturally occurring protein) and high predicted function $g(x)$ (e.g. activity or stability) by taking the product of multiple pre-trained 'expert' distributions:

$$\log p(x) = \sum_i f_i(x) + \lambda \sum_j g_j(x) - \log Z. \tag{4}$$

Each $f_i(x)$ is an unsupervised model, each $g_j(x)$ is a supervised model, and

$$Z = \sum_{x \in X} \exp\left(\sum_i f_i(x) + \lambda \sum_j g_j(x)\right)$$

is the unknown normalizing constant. Typically, $f_i(x)$ has been trained to do density estimation on unlabeled yet aligned sequences (e.g. a collection of evolutionarily related sequences provided by a multiple sequence alignment (MSA)) or unaligned sequences. While in this work we assume the $g_j(x)$ are an ensemble of nonlinear models trained to regress activity from a labeled dataset of mutants, this formulation can easily be extended to the multi-objective case where, for e.g. $g_1$ predicts activity and $g_2$ predicts stability.

The unsupervised experts $p_f(x) \propto \prod_i \exp\left(f_i(x)\right)$ act as a soft constraint that keeps the sampler near regions of high evolutionary density and away from, e.g. adversarial local optima of the supervised models. Examples of unsupervised models that provide an evolutionary density score include the EVmutation Potts model (Hopf *et al* 2017), the ESM protein masked language models (Rives *et al* 2021, Lin *et al* 2022), DAEs, energy based models (EBMs), autoregressive protein language models (e.g. ProGen2 (Nijkamp *et al* 2022)), and normalizing flows (non-exhaustive list). The supervised expert $p_g(x)$ acts as a soft constraint that guides sampling towards proteins that have high activity, where $p_g(x) \propto \prod_j \exp\left(\lambda\, g_j(x)\right)$ assigns high probability to sequences with high activity. The hyperparameter $\lambda \geqslant 0$ allows us to balance the contribution of the unsupervised and supervised experts; for example, we can emphasize the 'realism' of the protein (the evolutionary density) by setting $\lambda$ to 0. However, recent evidence indicates that evolutionary density scores also positively correlate with protein fitness in a 'zero-shot' manner (i.e. without using any labels) (Meier *et al* 2021, Hsu *et al* 2022, Weinstein *et al* 2022).

### 4.1. Product of experts gradient-based discrete MCMC

Sampling from the product of experts (equation (4)) is difficult since the normalization constant $Z$ is assumed unknown and is intractable to compute in practice. Although we have a good initialization for an MCMC sampler—$x^{\mathrm{WT}}$, the WT protein of verified viability—traditional MCMC based on random walk exploration is too inefficient to discover variants with high predicted activity in reasonable time.

Our solution is to use fast gradient-based discrete MCMC. We need only additionally assume that each expert is a continuous function that is differentiable at each discrete $x \in X$ (e.g. as is the case when the experts are neural networks). In detail, assume we have $M$ continuously differentiable unsupervised experts and $N$ continuously differentiable supervised experts. During each step of MCMC, we use the gradients of the $M + N$ experts to approximate the change in product of experts likelihood due to making $R$ point mutations to the current protein variant. This allows us to bias the proposal distribution towards the most promising $R$ mutations. The gradient-based Taylor approximation of our MCMC proposal for $\log p(x)$ with path length $R \sim \mathrm{Unif}(1, U)$ is $\tilde{q}_R(x'|x) = \prod_{r=1}^{R} \tilde{q}\left(x^r|x^{r-1}\right)$, where

$$\tilde{q}\left(x^r|x^{r-1}\right) \propto \exp\left(\frac{1}{2}\sum_{i=1}^{M} \nabla_{x^0} f_i(x^0)^T (x^r - x^{r-1}) + \frac{\lambda}{2}\sum_{j=1}^{N} \nabla_{x^0} g_j(x^0)^T (x^r - x^{r-1})\right) \mathbf{1}(x^r \in \mathcal{N}(x^{r-1})). \tag{5}$$

We use this path proposal to sample $R$ single amino acid substitutions, which we apply to the current variant $x^0 := x$ at each step of MCMC. The terminal state of the path $x^R := x'$ is the variant that results from the accumulation of the $R$ substitutions. To avoid computing extra forward and backwards passes through the product of experts for the intermediate path proposals $\tilde{q}(x^r|x^{r-1})$, following Sun *et al* (2022b) we re-use the gradient taken with respect to the path origin $x^0$ instead of recomputing gradients at intermediate states. The same is done for the reverse path proposals $\tilde{q}(x^{r-1}|x^r)$ with respect to the terminal state $x^R$. In the next section, we characterize the efficiency of our sampler to understand how composing Taylor approximations for multiple experts affects the theoretical convergence rate.

**Algorithm 1. Plug and Play Directed Evolution (PPDE).**

**input** one-hot encoded wild-type protein $x^{WT}$, unsupervised experts $f_i$, supervised experts $g_j$, scale $\lambda$, max path length $U$
**output** evolved protein $x^*$
**while** *still searching* **do**
**define** $x := x^0$, $x' := x^R$, $\pi(x) := \sum_i f_i(x) + \lambda \sum_j g_j(x)$
// `compute the forward path proposal distribution`
sample path length $R \sim \text{Unif}(1, U)$
**for** $r \in \{1, \ldots, R\}$ **do**
$\tilde{h}(x^{r-1}) = \sum_i \nabla_x f_i(x)^T (x^r - x^{r-1}) + \lambda \sum_j \nabla_x g_j(x)^T (x^r - x^{r-1})$
$\tilde{q}(x^r|x^{r-1}) = \text{categorical}\left(\text{softmax}\left(\frac{\tilde{h}(x^{r-1})}{2}\right)\right)$
// `sample a single amino acid substitution and apply it to` $x^{r-1}$
$x^r \sim \tilde{q}(x^r|x^{r-1})$
**end for**
**for** $r \in \{R, \ldots, 1\}$ **do**
$\tilde{h}(x^r) = \nabla_{x'} \sum_i f_i(x')^T (x^{r-1} - x^r) + \lambda \nabla_{x'} \sum_j g_j(x')^T (x^{r-1} - x^r)$ $\quad \tilde{q}(x^{r-1}|x^r) = \text{categorical}\left(\text{softmax}\left(\frac{\tilde{h}(x^r)}{2}\right)\right)$
**end for**
// `accept` $x'$ `with probability`
$$\min\left\{1, \exp(\pi(x') - \pi(x)) \frac{\prod_{r=R}^{1} \tilde{q}(x^{r-1}|x^r)}{\prod_{r=1}^{R} \tilde{q}(x^r|x^{r-1})}\right\}$$
**end while**

Algorithm 1 shows pseudo-code for our fast MCMC sampler for plug and play directed evolution of proteins. The sampler follows the basic structure of MH MCMC. At each sampler step, we first compute the forward path proposal distribution $\tilde{q}_R(x'|x)$ which we use to sample the proposed protein $x'$. Then, we compute the reverse path proposal distribution $\tilde{q}_R(x|x')$. We use these distributions to compute an acceptance criterion for determining whether to accept or reject $x'$, after which the process repeats until termination (e.g. a predetermined number of MCMC steps is reached).

### 4.2. Sampler analysis

Since we are using a gradient-based approximation of the product of experts proposal distribution, a natural question is whether this approximation reduces the sample efficiency of our sampler. It turns out that the choice of unsupervised and supervised experts plays a key role in determining the theoretical sampler efficiency. In particular, the following corollary to theorem 3 from Sun *et al* (2022b) relates the smoothness of each expert's gradient to our sampler's ability to efficiently explore protein space.

**Corollary 1.** *Assume $\lambda = 1$, each expert $h_i$ is differentiable, $\nabla_x h_i(x)$ is $K_i$-Lipschitz, the max path length is $U$, and a 1-Hamming ball neighborhood $\mathcal{N}(x)$. Let $Q_R(x, x')$ and $\tilde{Q}_R(x, x')$ be the Markov transition kernels induced by our sampler with the product of experts proposal $q_R(x'|x)$ and with its approximation $\tilde{q}_R(x'|x)$, respectively. These transition kernels are related by*

$$\tilde{Q}_R(x, x') \geqslant \left(\prod_{i=1}^{M} e^{-K_i \frac{U(U+1)}{2}}\right) Q_R(x, x'). \tag{6}$$

See appendix B for the proof.

**Remark.** This result bounds the efficiency of the sampler by a *product of exponential functions* of each expert's gradient's Lipschitz constant $K_i$. This means that if just one expert has a gradient with a large Lipschitz constant (e.g. the unsupervised experts are highly nonlinear protein language models), it is possible that this expert greatly reduces the overall efficiency of the sampler. For supervised experts (possibly an ensemble), which are

typically shallow neural networks whose gradients have small Lipschitz constants, we can expect that the first-order approximation will not greatly reduce the sampler's efficiency. However, equation (6) is a fairly loose bound. We will empirically compare how the sampler fares with different types of experts in practice (see section 5.2.3).

## 5. Experiments

In this section, we present the results of multiple synthetic experiments. First, we validate that our proposed sampler is able to discover a diverse set of good optima for product of experts distributions in high-dimensional discrete spaces using a toy MNIST-based task. Then, with a realistic *in silico* directed evolution experimental setup, we characterize advantages of combining unsupervised and supervised models and compare our sampler with appropriate baseline plug and play algorithms. The baselines are:

- **Simulated annealing:** This is a simple random-mutation-based MCMC-style algorithm from Biswas *et al* (2021) for sampling from the global optimum of a discrete Boltzmann distribution $p = 1/Z \exp(-y/T)$ with temperature $T$ and $y = -\log p(x)$. We apply this to sample from our product of experts (equation (4)). At each sampling step, a new sequence is proposed by randomly sampling $m \sim \text{Poisson}(\mu - 1) + 1$ uniformly sampled amino acid substitutions with rate $\mu \sim \text{Uniform}(1, 2.5)$. Proposed variants are accepted with probability $\min(1, \exp((\hat{y} - y)/T))$ with $T$ annealed over time to encourage the sampler to become more exploitative. Variants with higher $\log p(x)$ are always accepted whereas sequences that decrease $\log p(x)$ are accepted with a probability proportional to the difference scaled by the temperature.
- **Random sampling:** We provide a simple baseline that does *not* use search but rather uniformly samples variants around the WT protein, with aims of quantifying the advantage of using search to accumulate multiple promising mutations. Given a budget of $N$ variants, we use the same mutation proposal algorithm as `simulated annealing` to sample and apply a single mutation (which could consist of multiple amino acid substitutions) to the WT. After sampling $N$ such variants, we sort them by $\log p(x)$ and return the top $K$.
- **MALA-*approx*:** This baseline employs a continuous relaxation of discrete protein space to use stochastic gradient-based continuous optimization inspired by Nguyen *et al* (2017). During each step, `MALA-approx` samples

$$\tilde{x}' \sim \mathcal{N}\left(\tilde{x} + \frac{\epsilon}{2} \nabla_{\tilde{x}} \log p(\texttt{round}(\tilde{x})), \epsilon^2\right), \tag{7}$$

  where $\epsilon$ is a step size and $\nabla_{\tilde{x}} \log p(\texttt{round}(\tilde{x}))$ is a straight-through estimate of the gradient (Bengio *et al* 2013) of the product of experts with respect to the current relaxed variant. The continuous relaxation is based on a temperature-controlled relaxation of categorical variables to continuous vectors on the simplex (Jang *et al* 2016, Maddison *et al* 2016) with temperature $\tau$. That is, we interpret each sequence position of protein $x$ to be a rounded sample from a relaxed categorical distribution. We run this sampler in 'logit space' of these distributions since the support of logit space extends across the entire real line, whereas enforcing valid probability distributions at each sequence position would require solving a difficult constrained optimization problem. Given the one-hot WT protein $x^{\text{WT}}$, the sampler is initialized at logit values given by a convex combination of a uniform distribution over all amino acids and the WT: $\log((1-\tau) \cdot 1/|V| + \tau \cdot x^{\text{WT}})$. To `round` to discrete values for evaluating $\log p(\texttt{round}(\tilde{x}))$, we sample from the relaxed categorical distributions with logits $\tilde{x}$ and then take the argmax at each sequence position.
- **CMA-ES:** The `CMA-ES` algorithm (Hansen and Ostermeier 1996) is a powerful evolutionary strategies optimizer that accelerates search by maintaining an estimate of higher order relational statistics between each dimension of the search space. This algorithm is designed for continuous spaces with tens or hundreds of dimensions to eventually concentrate around the global optimum. It iteratively updates a Gaussian distribution (both mean and full covariance matrix) by sampling and evaluating a population of candidates. Like `MALA-approx`, it requires using a continuous relaxation of protein space. Simply, we 'flatten' a one-hot protein $x$ into a vector of dimension $LV$ which we then re-interpret as a continuous vector in $\mathbb{R}^{LV}$. To undo this relaxation, we reshape the vector into a length $L$ sequence and interpret the $V$ entries at each sequence position as scores, i.e. we take the argmax.

We use multiple metrics to compare samplers that measure not only activity but also whether designed proteins are evolutionarily plausible. Our results provide strong evidence that combining unsupervised evolutionary models with supervised models produces variants more likely to appear in nature and with higher fitness, especially when using unsupervised models fit on aligned sequences. Moreover, we find that composing multiple unsupervised models trained on aligned *and* unaligned sequences has the best

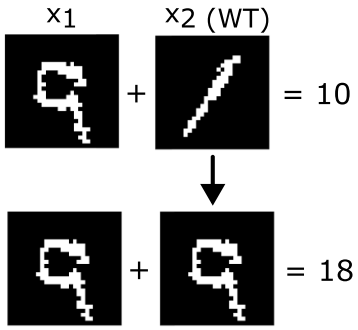



**Figure 2.** MNIST-Sum: the objective is to evolve the wild type image (right) to maximize the sum of the two digits.

performance. We also verify that, by and large, our gradient-based discrete MCMC sampler for product of experts achieves better sampling performance than the random-walk-based and evolutionary-strategies-based baselines.

### 5.1. MNIST-Sum

Before we present results on ML-based directed evolution, we evaluate the samplers on a toy search problem where optimized designs are easy to interpret.

**Setup:** We construct a binary MNIST task that emulates a typical ML-based directed evolution problem. We are given two binary MNIST images $x_1$ and $x_2$ where $x_2$ is the WT image. Each image is treated as a length $L = 784$ one-hot sequence where $V = \{0, 1\}$. The goal is to evolve $x_2$, keeping $x_1$ fixed, to maximize the sum of the two digits (figure 2). Here, the equivalent of an amino acid substitution is flipping a binary pixel. This task is difficult for methods that optimize in discrete space, since flipping most pixels causes the image to 'fall off' the underlying low dimensional manifold of MNIST digits. To obtain supervised experts $p_g(x)$ for a product of experts $\log p(x)$ (equation (4)), we use a labeled dataset of 50 K pairs of binary MNIST digits with sum $\leqslant 10$ (the 'assay-labeled proteins') to train an ensemble of three shallow siamese ConvNets that regresses the sum $x_1 + x_2$. We then use an unlabeled datasest of 50 K binary MNIST images (the 'unlabeled protein database', such as UniRef) to train a deep EBM and to train a DAE for unsupervised experts $p_f(x)$ (see appendix for model details).

PPDE is compared against `simulated annealing`, `MALA-approx`, and `CMA-ES`. All methods use a single EBM unsupervised expert and an ensemble of three -convolutional neural networks (CNNs) as supervised experts. To examine the importance of the unsupervised expert, we run PPDE with only supervised experts (`PPDE (None)` or `PPDE (supervised only)`). We then highlight PPDE's plug and play ability by running it with an EBM expert (`PPDE (EBM)`) and then swapping it out for a DAE expert (`PPDE (DAE)`) *without performing any further fine-tuning or re-training*. Key hyperparameters for each sampler are tuned on the digit pair $x_1 = 1, x_2 = 5$ (which has maximum sum 10 since $x_2$ is evolved while $x_1$ is held fixed) for $10\,K$ steps. We test each sampler on the out-of-distribution image pair $x_1 = 9, x_2 = 1$ with maximum possible sum 18. In this case, 'solving' the task is equivalent to evolving the WT '1' digit into a '9'. Each method evolves a population of size 128 and each sample in the population is optimized for a budget of $20\,K$ steps. To estimate 'ground truth' sums for optimized image pairs, we train an ensemble of three siamese ConvNets to near perfect accuracy on pairs of digits whose sums are $\leqslant 18$.

**Results:** The qualitative and quantitative results in figure 3 show that composing the supervised experts with an unsupervised expert (the EBM or the DAE) is *necessary* to solve the task; otherwise, the PPDE sampler is unable to avoid adversarial inputs and each run of the sampler converges to white noise images. We also observe that PPDE samples a highly diverse population of digits with noticeable semantic differences. We see that `PPDE (DAE)` outperforms `PPDE (EBM)`, which we suspect could be caused by the DAE's gradients being more informative due to its denoising training objective. `Simulated annealing` and `MALA-approx` fail in this high-dimensional space due to the difficulty of using random walks to find search directions that increase the product of experts probability. `CMA-ES` is outperformed by PPDE despite its use of covariance information to accelerate search in high-dimensional spaces. PPDE's evolved images are considerably more realistic than those produced by CMA-ES. We also observe that on this task, `CMA-ES` produces images that show low *semantic* diversity (the images differ at the pixel-level but the by and large look highly similar). We speculate that PPDE gains an advantage over CMA-ES by using gradients to select which pixels to mutate instead of approximate high-order statistics.

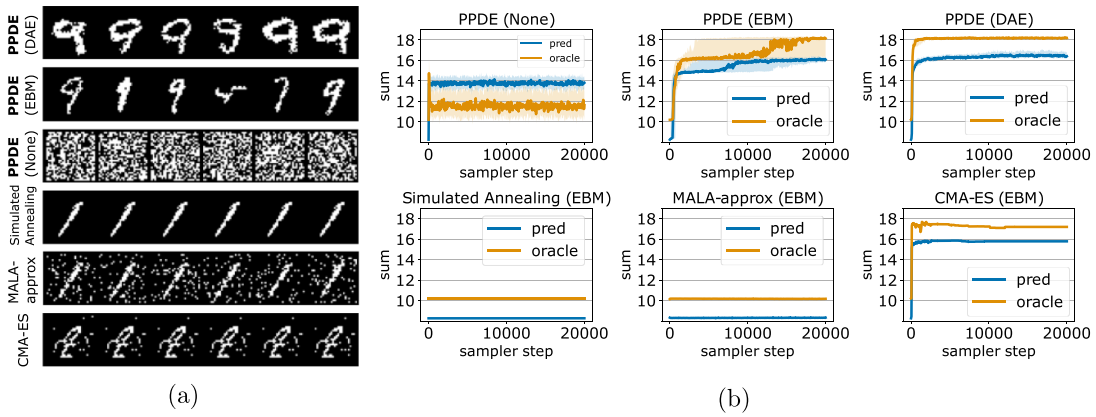



**Figure 3.** MNIST-Sum results: (a) random samples from the optimized digit populations. The samples from `PPDE (DAE)` have the highest diversity and also closely resemble MNIST images of a '9'. `PPDE (None)` does not use an unsupervised expert which causes sampler trajectories to become trapped in adversarial regions of the space where samples resemble white noise. The random mutation proposals employed by `simulated annealing` and `MALA-approx` cause these samplers to fail due to the high-dimensionality of the space. `CMA-ES`'s samples have low diversity yet converge onto a population of images that somewhat resemble a '9'. (b) The supervised experts' predicted sum (orange) vs. the oracle's predicted sum (blue) for the sampler trajectories. Each line plot is the 70th percentile sum with the shaded region showing the 50th and 90th percentiles. `PPDE (DAE)` reliably maximizes the sum of the digits ('18') as measured by the oracle model.

### 5.2. *In silico* directed evolution

**Datasets:** We use three benchmark proteins with partially characterized wide fitness landscapes and MSAs from Hsu *et al* (2022) for *in silico* directed evolution experiments: the poly(A)-binding protein (PABP) dataset of variants measuring binding activity (95 residues, each variant has $\leqslant 2$ mutations), the ubiquitination factor E4B (UBE4B) protein dataset measuring ligase activity (103 residues, each variant has $\leqslant 6$ mutations), and green fluorescent protein (GFP) dataset measuring fluorescence (238 residues, each variant has $\leqslant 15$ mutations). To emulate a realistic protein engineering setup with reasonable amounts of data for training our supervised experts $g(x)$, we use the '2-vs-rest' mutation train/test split as suggested in Dallago *et al* (2021)—that is, after splitting each dataset 80/20, we keep only sequences with two or fewer mutations for training and subsample 10% of these. For PABP, both the train and test sets only have variants with at most two mutations. This amounts to 3 K sequences for PABP, 2 K for UBE4B, and 1 K for GFP.

**Metrics:** Quantitative evaluation of optimized variants involves measuring the activity (log fitness), the likelihood to appear in nature (evolutionary density), population diversity, and ability to explore the fitness landscape around the WT (average number of mutations per variant). To compute log fitness, we take inspiration from model-based optimization benchmarks (Trabucco *et al* 2022) and simulate an expensive wet lab verification with a powerful 'oracle' model that we train with 80% of all variants (with all numbers of mutations) for each protein. Log fitness is scored relative to WT with the Augmented EVmutation Potts model from Hsu *et al* (2022) as the oracle. To compute an evolutionary density score relative to WT, we use a different transformer, the MSA Transformer (Rao *et al* 2021), conditioned on 500 randomly subsampled sequences from each MSA. For both log fitness and evolutionary density, positive values mean the variant outscored the WT while negative values are worse than WT. We run each sampler 128 times for 10 K steps each, initialized at the WT, and select the sample with the maximum $\log p(x)$ per run for the final population. As plug and play baselines we again use `simulated annealing`, `CMA-ES`, and `MALA-approx`, as well as `random sampling`.

**Experts:** We use the EVmutation Potts and ESM2 family of protein language models as pre-trained unsupervised experts. All baselines use a single Potts expert and are directly compared with `PPDE (Potts)`. Unless stated otherwise, we use the 150 M parameter version of ESM2. We also run our sampler with *both* Potts and ESM2 (i.e. we take the sum of the Potts and ESM2 evolutionary density scores) (`PPDE (Potts+ESM2)`). For the supervised expert we use an ensemble of three shallow CNNs introduced as a baseline in Dallago *et al* (2021) which regress activity directly from one-hot encoded proteins. See the appendix for architecture details and details on how we implement the evolutionary density scoring for the Potts and protein language model scoring functions.

**Choosing $\lambda$:** Multiplying the supervised experts with the hyperparameter $\lambda \geqslant 0$ trades off how much the unsupervised and supervised experts influence which variants are sampled from the product of experts distribution. A larger value for $\lambda$ inflates the supervised expert scores which causes the sampler to have more confidence in variants with high predicted activity. We use a simple heuristic to select $\lambda$ for the three

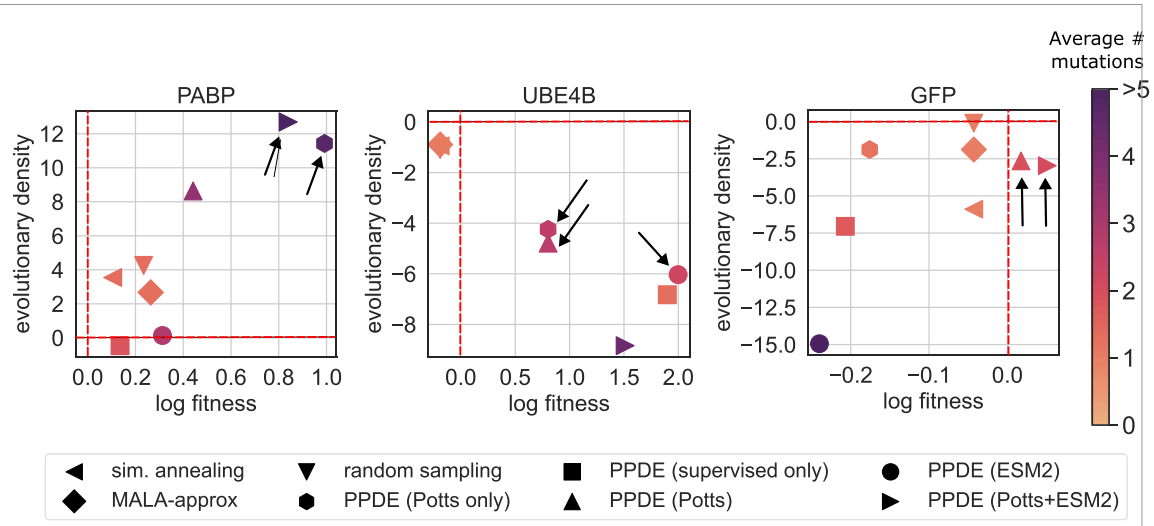



**Figure 4.** *In silico* directed evolution 80th percentile results. Positive log fitness and evolutionary (evo.) density scores are better than the WT while negative values are worse than WT. The red dashed lines intersect at WT performance $(0, 0)$. Each sampler is colored by the average number of mutations per variant (higher is better). On PABP (left), `PPDE (Potts+ESM2)` and `PPDE (Potts only)` (arrows) achieve the highest evo. density and log fitness—both better than WT—and with relatively high numbers of mutations. On UBE4B (center), `PPDE (Potts)` and `PPDE (Potts only)` (arrows) appear to best trade off evo. density, log fitness, and mutation count. `PPDE (ESM2)` achieves the best log fitness but with slightly worse evo. density scores. On GFP (right), only `PPDE (Potts)` and `PPDE (Potts+ESM2)` (arrows) discover samples with log fitness better than WT. `CMA-ES` is omitted for viewing clarity due to outlier results on UBE4B (log fitness: 2.54, evo. density: −94.76) and GFP (log fitness: −1.13, evo. density: −22.85). Best viewed in color.

**Table 1.** Population diversity (% of variants that are unique).

| | Random search | Simulated annealing | MALA-*approx* | CMA-ES | PPDE (Potts only) | PPDE (Super. only) | PPDE (Potts) | PPDE (ESM2) | PPDE (Potts+ESM2) |
|---|---|---|---|---|---|---|---|---|---|
| PABP | 32.8 | 28.9 | 28.9 | 0.8 | **85.2** | 60.2 | 65.6 | 63.1 | **85.2** |
| UBE4B | 7.0 | 4.7 | 6.2 | 3.1 | 12.5 | 18.8 | 18.8 | **36.5** | 31.3 |
| GFP | 9.4 | 3.9 | 9.4 | 92.2 | 8.6 | 59.4 | 22.7 | **92.9** | 21.9 |

benchmark proteins and for each of the Potts and ESM2 unsupervised experts. The heuristic roughly assigns equal emphasis to the unsupervised and supervised expert scores. In detail, we estimate the {`min`, `max`, `mean`} statistics of expert scores computed from a small representative sample of labeled protein variants and then picks $\lambda$ such that the statistics for the supervised expert scores are close to those of the unsupervised expert scores. We sample 100 protein variants from the training set that have lower activity than WT and 100 proteins with higher activity higher than WT and then evaluate the {`min`, `max`, `mean`} of the unsupervised expert scores and $\lambda \times$ {`min`, `max`, `mean`} of the supervised expert scores for $\lambda \in \{0.5, 1, 3, 5, 15\}$. When using the Potts unsupervised expert we use $\lambda = 5, 0.5, 15$ and when using the ESM2 expert we use $\lambda = 5, 3, 1$ for the PABP, UBE4B, and GFP proteins, respectively. Less manual approaches that tune $\lambda$ over a grid of values or *learn* $\lambda$ over a held-out validation set of labeled variants could be used instead if desired (Holtzman *et al* 2018).

*5.2.1. Main results*

Figure 4 shows the 80th percentile metrics for each evolved population colored by the population's average number of mutations. The 50th and 100th percentile results for all samplers are provided in a table in the appendix (table 2). Diversity scores are in table 1. We make the following observations.

**Diversity:** Across all three proteins, PPDE-based samplers achieve the best diversity (highest percentage of unique variants), implying superior exploration of good optima around the WT protein.

**Evolutionary density, log fitness, and mutation counts:** Out of the samplers using the Potts unsupervised expert, `PPDE (Potts)` discovers variants with higher log fitness and average number of mutations than `random search`, `simulated annealing`, and `MALA-approx`. In terms of evolutionary density, `PPDE (Potts)` achieves the highest scores on PABP, and on the more challenging UBE4B and GFP proteins, we suspect that the slightly lower 80th percentile evolutionary density scores compared to the baselines (except for `CMA-ES`, which scored poorly) are in part due to PPDE discovering variants with higher average mutations ($2-4+$ mutations vs. $\sim$1 mutation for the baselines) and higher log fitness. We found

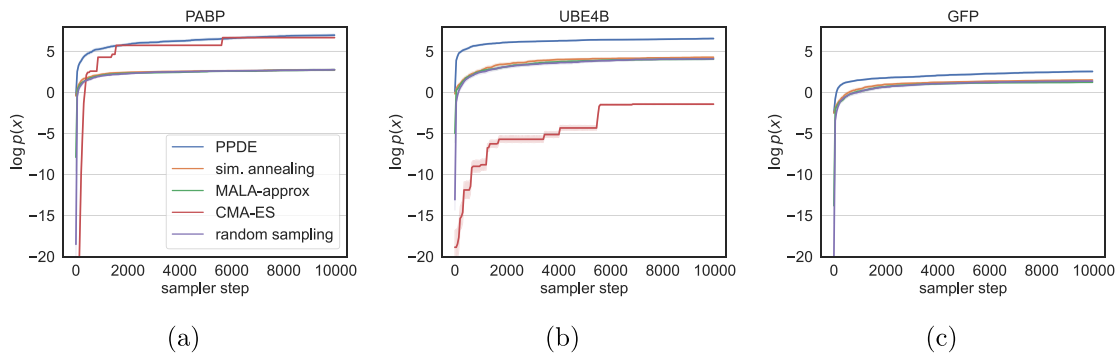



**Figure 5.** PPDE is most efficient at finding good optima of the high-dimensional product of experts. Cumulative maximum product of experts log probability averaged across the population (higher is better). All samplers use the same product of experts target distribution with Potts unsupervised expert (other unsupervised experts not shown for clarity of presentation). `CMA-ES` is not visible in (c) because the optimization starts and remains around $\log p(x) \approx -50$.

that the `CMA-ES` sampler had inconsistent performance across the three proteins. Among samplers using the Potts unsupervised expert, it achieves the highest fitness scores on two proteins (PABP and UBE4B), but does so with diversity scores of 0.8% and 3.1% compared to 85.2% and 12.5% for `PPDE (Potts)` (table 1). CMA-ES finds variants with high numbers of mutations ($\sim 10-17$) but low evolutionary density scores (50th percentile scores of 3.47 on PABP, $-94.76$ on UBE4B and $-62.43$ on GFP compared to 6.92, $-4.79$, and $-5.98$ for PPDE; see table 2). It finds just one protein variant for PABP that is 17 mutations from WT with an evolutionary density score of 3.47, and it seems to have significant difficulty with the larger proteins UBE4B and GFP; e.g. on GFP the average 50th percentile log fitness is $-2.50$ compared to $-0.04$ for PPDE. Figure 5 shows sampler trajectories, with which we see that across all three proteins, PPDE is the superior approach for efficiently sampling from the high-dimensional discrete product of experts distributions. We see here again that `CMA-ES` performs worse as the protein sequence length and fitness landscape complexity increases. It appears that a major limitation of CMA-ES is its continuous relaxation, which projects each sequence in the population to a variant far from the WT. For GFP, the population average product of experts probability starts and stays around $\log p(x) \approx -50$. We conclude that `CMA-ES` can be recommended for use only when a single protein variant is desired and the length of the protein is relatively small (e.g. $\leqslant 95$ residues).

**Swapping unsupervised experts:** *Without using an unsupervised expert*, PPDE discovers variants with worse evolutionary density scores than even `random sampling`, confirming our observations from the previous MNIST-Sum experiment that supervised experts alone are susceptible to adversarial inputs. We also examined PPDE without any supervised expert (`PPDE (Potts only)`) by setting $\lambda = 0$. The observation that `PPDE (Potts only)` achieves a higher log fitness on PABP and only slightly worse log fitness on UBE4B than `PPDE (Potts)` is not surprising, as recent evidence suggests unsupervised evolutionary density scores are (at least moderately) predictive of mutation effects (Meier *et al* 2021, Hsu *et al* 2022, Weinstein *et al* 2022). However, the sampler performance degrades significantly by all metrics on GFP. When using the ESM2 unsupervised expert, PPDE discovers variants with lower evolutionary density scores than when using the Potts. This is not altogether unexpected since the MSA Transformer, which is used to score the variants, is conditioned on the same MSA as used to fit the Potts model. Although, this result also suggests that using unsupervised experts fit to *aligned sequences* is more promising for ML-based directed evolution. We observe a promising compositional effect from combining unsupervised experts trained on unaligned sequences (ESM2) and aligned sequences (Potts); e.g. across all three proteins the `PPDE (Potts+ESM2)` sampler achieves higher 100th percentile fitness and density scores than when using the Potts or ESM2 experts alone, and increases the diversity and average number of mutations as well.

### *5.2.2. Qualitative results*

We provide a qualitative analysis of the mutations discovered by the `supervised only`, `simulated annealing`, `PPDE (Potts)` and `PPDE (Potts+ESM2)` samplers in figure 6. Many of the variants found by PPDE tend to have the same 1–3 'highly plausible' mutations in addition to a few 'rare' mutations that show up in variants at a much lower rate.

### *5.2.3. Comparing protein language model experts*

Figure 7 highlights the ability of PPDE to successfully sample from the 35 M, 150 M, and 650 M parameter ESM2 language models without any extra fine-tuning or re-training. This flexibility makes it easy to leverage continued improvement in protein language models, or use language models fine-tuned for specific protein

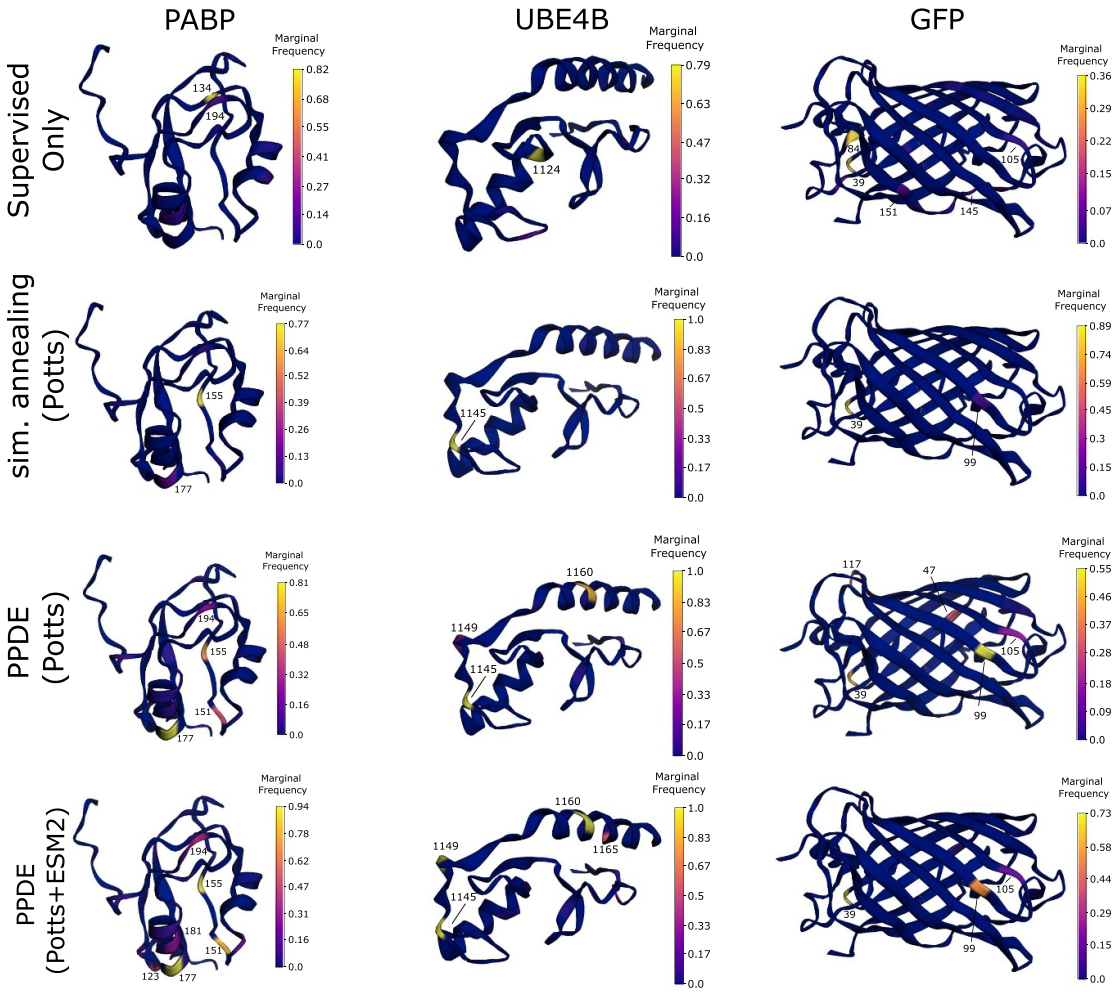



**Figure 6.** Visualization of mutation site frequency. Best viewed in color and zoomed in. The marginal frequency of a mutation site is the fraction of protein variants in the population with a mutation at a particular sequence position. We visualize site frequencies as a heatmap on the 3D structure of the WT protein. For example, nearly 100% of UBE4B variants discovered by `PPDE (Potts+ESM2)` have a mutation at position 1145. We annotate only a few high frequency sequence positions per protein for clarity; many sequence positions with a low frequency are not annotated.

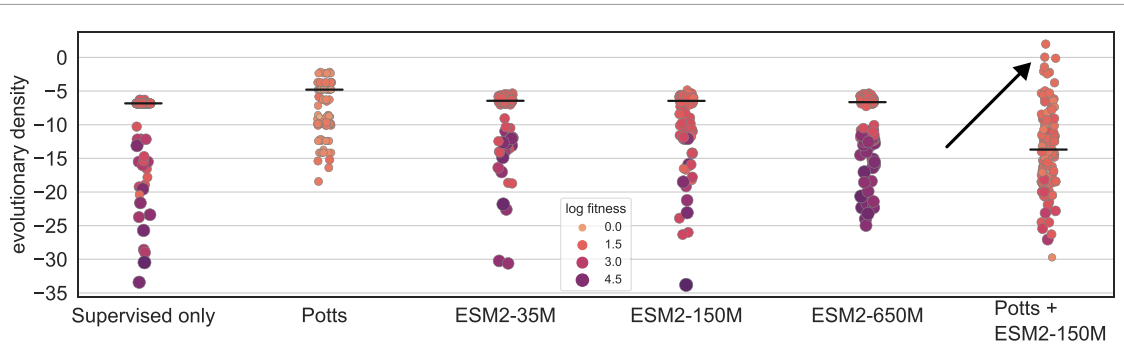



**Figure 7.** Swapping in different protein language model experts. Full populations (128 variants) colored by log fitness with the median evolutionary density annotated. We swap out different unsupervised experts for the *in silico* directed evolution of UBE4B using the same supervised expert. This includes a 650 M parameter protein language model (`ESM2-650 M`). Combining unsupervised experts trained on aligned and unaligned sequences (`Potts+ESM2-150 M`) discovers variants with highest density and higher predicted fitness than WT (arrow).

families. As the number of ESM2 parameters increase, we observe a trend where variants with higher predicted activity and marginally higher evolutionary density are discovered. A similar observation in Nijkamp *et al* (2022) also suggests that model size positively correlates with zero-shot fitness prediction on wide fitness landscapes. When directly comparing `supervised only` and `PPDE (Potts)`, we can see a large improvement in evolutionary density when the Potts expert is used. ESM2, which is trained on

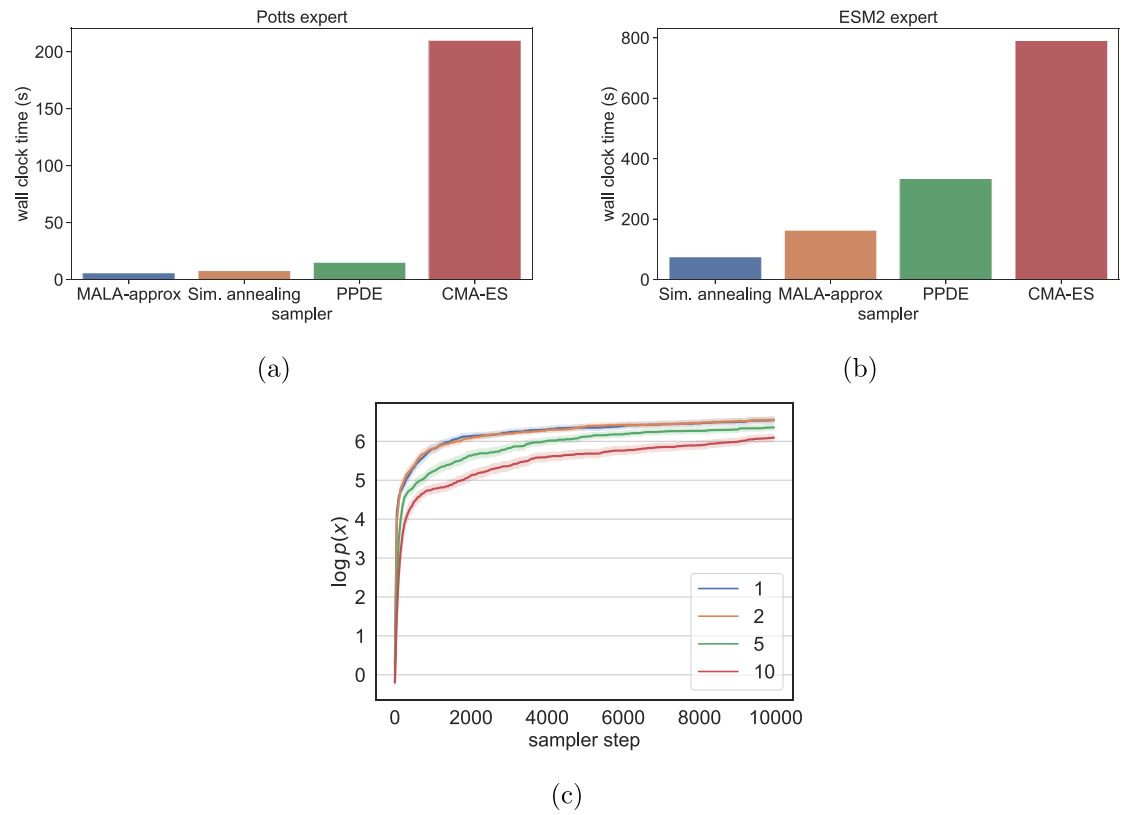



**Figure 8.** Wall clock run time comparisons using the Potts (a) and ESM2 (b) unsupervised experts. Run times are averaged over seven trials using the PABP protein, 1 K sampler steps, and a population of size 16. (c) Comparing PPDE on UBE4B with various max path lengths $U = 2X - 1$ where $X \in \{1, 2, 5, 10\}$.

unaligned proteins, achieves lower evolutionary density scores than the Potts, suggesting it acts as a much softer search constraint. Combining the Potts and ESM2 experts results in improving the top percentile of discovered variants compared to those found by using the Potts or ESM2 experts alone.

*5.2.4. Path length sensitivity analysis*

PPDE's proposal distribution samples a *path* of $R \sim \text{Unif}(1, U)$ single substitutions at each sampler step to rapidly navigate through protein space. We conducted a sensitivity analysis on the maximum path length $U$, where $U = 2X - 1$ and $X \in \{1, 2, 5, 10\}$ (figure 8(c)) on the UBE4B protein. In this experiment, we found that path lengths longer than 3 did not significantly improve sampler efficiency. This is possibly because we already have a good initialization for the sampler (the WT) and short path lengths are sufficient for discovering high quality variants. It is also possible that, in our setting, the accumulation of errors from the first order Taylor approximation along a long path leads to worse sampler efficiency despite better exploration. Moreover, using a longer path length leads to discovering variants with a higher number of mutations, whose activity could be inaccurately characterized by the supervised experts.

*5.2.5. Run time comparison*

We compare wall clock run times for PPDE and the baseline samplers under the Potts (figure 8(a)) and ESM2 (figure 8(b)) experts using the PABP protein, 1 K sampler steps, and a population size of 16 variants. Samplers are run on a single NVIDIA Tesla cloud GPU; see appendix for complete hardware details. `CMA-ES` is significantly slower than the other samplers. When using the Potts expert, `PPDE`'s run time is relatively comparable to `simulated annealing` and `MALA-approx`. The cost of computing the gradient of ESM2 is much higher than for the Potts model, so in this setting `simulated annealing` (which does not use gradients) has lower run time than `MALA-approx` and `PPDE`.

## 6. Limitations

In this section, we discuss limitations of the proposed sampling method related to computational costs and handling insertion/deletion mutations.

Each step of the PPDE sampler requires two backwards passes—one for computing the forward proposal and one for the reverse proposal. When PPDE is used with a large protein language model whose backwards pass has a large memory footprint, this becomes a bottleneck if sufficient computing resources are unavailable. Fortunately, MCMC samplers are embarrassingly parallel. For example, with a budget of $K$ available GPUs, we can evolve a single protein per GPU in parallel.

The wall-clock time incurred by running PPDE depends on the ability to rapidly evaluate and differentiate through the product of experts distribution. For certain classes of unsupervised experts, this is difficult. For example, evaluating the log probability of a protein with a variational autoencoder (e.g. DeepSequence (Riesselman *et al* 2018)) requires computing a large-sample Monte Carlo estimate of the evidence lower bound, which is highly computationally expensive.

PPDE also does not currently support inserting and deleting amino acids (indels). We necessarily assume the product of experts is locally smooth in a neighborhood around the current protein to compute our proposal distribution for MCMC. This assumption would need to still hold despite proposing length changes to the sequence. While we do not believe lack of support for indels is a fundamental limitation of the sampler, supporting indel mutations appears to be nontrivial.

## 7. Conclusions

In this study, we have shown how to flexibly combine unsupervised models of evolutionary density, including protein language models, and supervised models of protein function and how to efficiently sample from the resulting distribution to discover proteins that maximize a desired function while avoiding poor local optima. This strategy leverages the vast amounts of unlabeled data that are available for unsupervised (or self-supervised) pre-training to improve designed sequences, even when relatively few labelled data are available for training the fitness function. Our empirical results suggest our framework offers a practical and effective new paradigm for machine-learning-based directed evolution. Although our framework can potentially be applied to a broader range of biological sequence design tasks, we focused solely on proteins due to the variety of available pre-trained models.

Based on our findings, we recommend constructing the product of experts distribution with unsupervised experts that have been pre-trained on *aligned* protein sequences (e.g. MSAs) when available. We found that these models more strongly constrain search to the evolutionarily plausible mutations than unsupervised models pre-trained on *unaligned* protein sequences. However, *combining* unsupervised models of both types produced designs superior to using one or the other alone.

One important capability that we did not demonstrate is the incorporation of additional hard constraints, such as a maximum allowable number of mutations per variant or the preservation of particular regions of the WT sequence. In our framework, we can easily accomplish this by forcing the mutation proposal probabilities at the sequence positions in question to zero, which can be done by setting their logits to negative infinity in the softmax in Algorithm 1. Future work may also extend this framework to larger problems in biological design. For instance, the simultaneous engineering of several sequences in multimeric enzyme complexes, or incorporating substrate structure in evaluating the likelihood of enzyme-substrate complexes.

## Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/pemami4911/ppde.

## Acknowledgments

This work was authored by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. Funding provided by the U.S. Department of Energy Bioenergy Technologies Office and the Laboratory Directed Research and Development (LDRD) Program at NREL. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

## Appendix A. Deriving the gradient-based approximate proposal

We provide a short derivation that illustrates how to obtain a gradient-based approximate locally-balanced informed proposal (equation (2)) from the locally-balanced informed proposal of equation (1). The goal is to avoid evaluating $f(x')$ for all $x' \in \mathcal{N}(x)$ when computing the proposal distribution $q(x'|x) \propto \exp(f(x') - f(x))^{\frac{1}{2}} \mathbf{1}(x' \in \mathcal{N}(x))$. The basic idea from Grathwohl *et al* (2021) is to assume that $f$ is a continuously differentiable function, which allows us to approximate $f(x') - f(x)$ with a first-order Taylor series.

In detail, the first-order Taylor series of the function $f(x)$ at the point $a$ is

$$f(x) \approx f(a) + \nabla_a f(a)^\intercal (x - a). \tag{8}$$

By subtracting $f(a)$ from both sides, we get

$$f(x) - f(a) \approx \nabla_a f(a)^\intercal (x - a). \tag{9}$$

To arrive at the gradient-based proposal, suppose $a$ is the current state of our MCMC sampler and let $x$ be any point in the neighborhood of $a$, $x \in \mathcal{N}(a)$. When we plug in our first-order approximation,

$$\tilde{q}(x|a) \propto \exp(\nabla_a f(a)^\intercal (x - a))^{\frac{1}{2}} \mathbf{1}(x \in \mathcal{N}(a)). \tag{10}$$

## Appendix B. Proof of corollary 1

The basic idea of the proof is to use the triangle inequality to obtain a bound on the approximation error of a *sum* of experts which assumes that each expert has sufficiently smooth gradients.

**Definition.** A function $f : \mathbb{R}^N \to \mathbb{R}$ has *K-Lipschitz continuous gradient* when

$$\|\nabla_{x'} f(x') - \nabla_x f(x)\| \leqslant L \|x' - x\| \tag{11}$$

for all $x, x' \in \mathbb{R}^N$.

For convenience, we reproduce a pertinent result here from Nesterov (1998) (Lemma 1.2.3).

**Lemma 1.2.3.** *Nesterov (1998) If $f : \mathbb{R}^N \to \mathbb{R}$ has an L-Lipschitz gradient, then for any $x, x' \in \mathbb{R}^N$ we have:*

$$|f(x') - f(x) - \langle \nabla_x f(x), x' - x \rangle| \leqslant \frac{L}{2} \|x' - x\|^2. \tag{12}$$

**Lemma B.1.** *For functions f and g with $K_f$-Lipschitz and $K_g$-Lipschitz gradients respectively, the sum-composition $h = f + g$ has $(K_f + K_g)$-Lipschitz gradient.*

**Proof.** By the triangle inequality:

$$\begin{aligned} \|\nabla_{x'} h(x') - \nabla_x h(x)\| &= \|\nabla_{x'} (f(x') + g(x')) - \nabla_x (f(x) + g(x))\| \\ &= \|\nabla_{x'} f(x') + \nabla_{x'} g(x') - \nabla_x f(x) - \nabla_x g(x)\| \\ &\leqslant \|\nabla_{x'} f(x') - \nabla_x f(x)\| + \|\nabla_{x'} g(x') - \nabla_x g(x)\| \\ &\leqslant (K_f + K_g) \|x' - x\|. \end{aligned} \tag{13}$$

□

**Lemma B.2.** *Suppose $f_i$, $i = 1, \ldots, M$ are functions with $K_i$-Lipschitz gradient. Then for any $x, x' \in \mathbb{R}^N$ we have*

$$\left| \sum_{i=1}^{M} \left( f_i(x') - f_i(x) \right) - \left\langle \sum_{i=1}^{M} \nabla_x f_i(x), x' - x \right\rangle \right| \leqslant \frac{\sum_{i=1}^{M} K_i}{2} \|x' - x\|^2. \tag{14}$$

**Proof.** Let $g = \sum_{i=1}^{M} f_i$ where each $f_i$, $i = 1, \ldots, M$ has $K_i$-Lipschitz gradient. By lemma B.1 we can see that $g$ has a $\sum_{i=1}^{M} K_i$-Lipschitz gradient. Then lemma B.2 follows immediately by applying lemma 1.2.3 from Nesterov (1998) to $g$. □

The proof of corollary 1 proceeds by bounding the approximation error between two consecutive states $x^{r-1}$ and $x' \in \mathcal{N}(x^{r-1})$ in a path of length $R \sim \text{Unif}(1, U)$. For simplicity we assume $\mathcal{N}(x)$ is the 1-Hamming ball, i.e. $\|x^r - x^{r-1}\|^2 = 1$.

For $g = \sum_{i=1}^{M} f_i$ which has $K = \sum_{i=1}^{M} K_i$-Lipschitz gradient, lemma 2 gives us that

$$-\frac{K}{2} \leqslant g(x') - g(x^{r-1}) - \langle \nabla g(x^{r-1}), x' - x^{r-1} \rangle \leqslant \frac{K}{2}.$$

Then an upper bound for $g(x') - g(x^{r-1})$ is

$$\begin{aligned} g(x') - g(x^{r-1}) &\leqslant \langle \nabla g(x^{r-1}), x' - x^{r-1} \rangle + \frac{K}{2} \\ &= \langle \nabla g(x^0), x' - x^{r-1} \rangle + \langle \nabla g(x^{r-1}) - \nabla g(x^0), x' - x^{r-1} \rangle + \frac{K}{2} \\ &\leqslant \langle \nabla g(x^0), x' - x^{r-1} \rangle + Kr + \frac{K}{2} \\ &= \langle \nabla g(x^0), x' - x^{r-1} \rangle + K\left(r - \frac{1}{2}\right). \end{aligned}$$

Following similar steps, we also have

$$g(x') - g(x^{r-1}) \geqslant \langle \nabla g(x^0), x' - x^{r-1} \rangle + K\left(r + \frac{1}{2}\right).$$

The remainder of the proof for corollary 1 exactly follows equations 64–72 in the proof of theorem 3 in Sun *et al* (2022b) with $g(x)$ which has $K$-Lipschitz gradient.

## Appendix C. Experiment details

### C.1. Neural architectures

**MNIST DAE:** The encoder and decoder neural networks are constructed out of residual blocks, where a single block has two $3 \times 3$ 2D Conv layers, each followed by a BatchNorm layer and Swish activation. The first Conv layer in the block uses a stride of 2 and a $1 \times 1$ shortcut Conv layer is added to its output. The encoder consists of one $3 \times 3$ 2D Conv layer, followed by two blocks, then another $3 \times 3$ 2D Conv layer, followed by a fully connected layer that projects the flattened output features into a 16-dimensional latent space. The encoder and decoder are symmetric, and the decoder is implemented with transposed Conv layers and the per-pixel output is interpreted as the logits of a Bernoulli distribution. All Conv layers use 64 channels. To train the DAE, we corrupt the input $28 \times 28$ binary MNIST image by randomly flipping $p$% of the pixels, where $p \sim$Unif(0,15), and minimize the reconstruction error. We use the AdamW optimizer with a learning rate of $1 \times 10^{-4}$ to train the model for 10 K steps using a mini-batch size of 100.

**MNIST EBM:** This model follows the residual EBM architecture from Grathwohl *et al* (2021) closely. The model consists of one $3 \times 3$ 2D Conv layer, followed by two residual blocks (the same blocks used by the DAE), and then an additional six residual blocks with all Conv layers using a stride of 1. The output features are flattened with global average pooling and then mapped to a scalar with a fully connected layer.
See Grathwohl *et al* (2021) for details about the contrastive divergence training algorithm and training hyperparameters.

**Protein ConvNet:** We use the ConvNet baseline from the FLIP benchmark (Dallago *et al* 2021) to regress activity. This model takes in a mini-batch of one-hot encoded protein sequences of shape $B \times L \times V$, applies a 1D Conv with kernel size 5 and ReLU activation to project $V$ to dimension $L$, followed by a fully connected layer to expand the feature dimension to $2L$. Then, max pooling is used to reduce the length of the sequence to 1, and a linear layer projects the feature dimension from $2L$ to 1. The model is trained with AdamW optimizer with a learning rate of $1 \times 10^{-3}$ using a mini-batch size of 256.

### C.2. Unsupervised expert score functions

We provide details for computing $\sum_i f_i(x)$ with each type of unsupervised expert.

**MNIST DAE:** For a given image $x$, we define $f(x)$ to be the sum of the binary cross entropy between the input pixel and the predicted pixel logits over all pixel locations.

**MNIST EBM:** We use the scalar output as $f(x)$, since it can be interpreted as an unnormalized log probability for the input. This only requires a single forward pass to compute.

**Protein EVmutation Potts:** We use the difference in the Potts Hamiltonian between the protein variant $x$ and the WT such that $f(x) = H(x) - H^{\mathrm{WT}}(x)$. The Hamiltonian for a one-hot protein $x$ is defined as

$$H(x) = \sum_{i=1}^{L} h_i^T x_i + \sum_{i,j=1}^{L} x_i^T J_{ij} x_j$$

where $J \in \mathbb{R}^{L\times L\times 20\times 20}$ and $h \in \mathbb{R}^{L\times 20}$ are the Potts parameters.

**Protein ESM2:** We use the difference in the sum of the per-amino-acid log probabilities between the protein variant $x$ and the WT for $f(x)$. For computational efficiency, we compute the score with a single forward pass and therefore do not use any masking of input sequence positions. In detail, for a one-hot protein $x$ let $\phi(x)$ be the $L \times V$ matrix of logits computed by a single unmasked forward pass of ESM2. Then the score is

$$f(x) = \sum_{i=1}^{L}\sum_{j=1}^{V} x_{ij}\texttt{log\_softmax}(\phi(x)_i)_j - \sum_{i=1}^{L}\sum_{j=1}^{V} x_{ij}^{WT}\texttt{log\_softmax}(\phi(x^{WT})_i)_j.$$

### C.3. Hardware details

We ran our experiments on the Summit supercomputer on a single NVIDIA V100 GPU. When running with ESM2, we speed up the sampler by parallelizing the population of 128 proteins across multiple GPUs. For example, we ran each of the 128 MCMC trajectories on a separate GPU in parallel when we used the 650 M parameter ESM2 model. Wall clock run time measurements were taken after-the-fact using a single NVIDIA Tesla T4 GPU available via the Google Colab free tier.

### C.4. Aligning the amino acid vocabulary across models

Not all pre-trained protein sequence models order the amino acids in their vocabulary in the same way. Also, some models, such as masked language models, have extra tokens such as <mask> beyond the standard 20 amino acids. To enable combining a diverse array of models, we first decide on a canonical ordering for the standard amino acids. For any pre-trained model that uses a different ordering than the canonical one, we pre-compute a permutation matrix that, when applied to the columns of the one-hot encoded protein $x$, will put the amino acids in the order which the model expects. We also pad the permuted $x$ with extra columns of zeros if the pre-trained model has a vocabulary larger than 20.

**Table 2.** 50th (100th) percentile scores. Population size is 128. Across all three proteins, `PPDE (Potts)` and `PPDE (Potts+ESM2)` in particular discover variants with higher predicted fitness and average # of mutations than `Random search`, `Simulated annealing`, and `MALA-approx`. Out of the samplers using the Potts expert, `PPDE (Potts)` achieves the highest evolutionary density scores on PABP. We suspect that the slightly lower 50th percentile evolutionary density scores compared to the baselines (except `CMA-ES`) on the more challenging UBE4B and GFP proteins are in part due to PPDE discovering variants with more average mutations ($2-4+$ mutations vs. $\sim 1$ mutation per variant) and higher fitness. The 100th percentile performance of `PPDE (Potts+ESM2)` shows impressively high log fitness *and* evolutionary density, however. `CMA-ES` sampler had inconsistent performance across the three proteins—it finds variants with high numbers of mutations ($10-17$) but low evolutionary density scores (3.47 on PABP, $-94.76$ on UBE4B and $-62.43$ on GFP compared to 6.92, $-4.79$, and $-5.98$ for `PPDE (Potts)`). It finds just one protein variant for PABP that is 17 mutations from WT with an evolutionary density score of 3.47, and it seems to have significant difficulty with the larger proteins UBE4B and GFP; e.g. on GFP the average 50th percentile log fitness is $-2.50$ compared to $-0.04$ for PPDE. We conclude that `CMA-ES` can be recommended for use only when a single protein variant is desired and the length of the protein is relatively small (e.g. $\leqslant 95$ residues).

| | Log fitness ↑ (Augmented EVmutation) | | | Evolutionary density ↑ (MSA Transformer) | | | Exploration (mean±std # muts) | | |
|---|---|---|---|---|---|---|---|---|---|
| Potts expert | PABP | UBE4B | GFP | PABP | UBE4B | GFP | PABP | UBE4B | GFP |
| PPDE | 0.27(0.86) | 0.39(1.18) | −0.04(0.24) | 6.92(13.43) | −4.79(−2.14) | −5.98(−0.76) | $3.5 \pm 0.9$ | $2.7 \pm 0.6$ | $2.0 \pm 0.3$ |
| Random search | 0.09(0.82) | −0.19(0.34) | −0.04(0.04) | 4.26(6.87) | −1.09(2.46) | −0.11(−0.11) | $1.3 \pm 0.5$ | $1.1 \pm 0.3$ | $1.0 \pm 0.2$ |
| Sim. annealing | 0.09(0.44) | −0.19(0.28) | −0.04(0.10) | 3.55(8.63) | −0.94(2.70) | −5.89(−0.99) | $1.3 \pm 0.5$ | $1.0 \pm 0.2$ | $1.0 \pm 0.1$ |
| MALA−*approx* | 0.09(0.56) | −0.19(0.58) | −0.04(0.10) | 2.25(5.14) | −0.89(1.54) | −6.74(−1.88) | $1.3 \pm 0.5$ | $1.03 \pm 0.2$ | $1.03 \pm 0.2$ |
| CMA−ES | 1.37(1.37) | 2.54(2.54) | −2.50(−0.15) | 3.47(3.47) | −94.76(0.0) | −62.43(0.0) | $17.0 \pm 0$ | $15.5 \pm 6.2$ | $10.2 \pm 9.4$ |
| Unsupervised expert | | | | | | | | | |
| Potts only | 0.70(1.47) | 0.12(0.99) | −0.18(0.21) | 9.17(18.54) | −4.24(−2.68) | −1.88(−1.59) | $4.7 \pm 1.2$ | $2.6 \pm 0.5$ | $1.2 \pm 0.4$ |
| Supervised only | 0.14(0.44) | 1.66(5.26) | −0.23(0.14) | −2.56(0.48) | −6.83(−6.29) | −9.28(−2.13) | $1.9 \pm 0.8$ | $1.3 \pm 0.7$ | $1.7 \pm 0.8$ |
| ESM2 | 0.14(0.63) | 1.66(5.56) | −5.55(0.16) | −2.38(5.56) | −6.58(−3.83) | −126.82(5.90) | $2.9 \pm 1.3$ | $2.2 \pm 2.2$ | $14.9 \pm 12.6$ |
| Potts+ESM2 | 0.44(1.48) | 1.30(3.33) | −0.04(0.33) | 9.12(19.34) | −13.6(1.98) | −7.17(8.11) | $5.3 \pm 1.8$ | $4.3 \pm 0.7$ | $2.1 \pm 0.3$ |

## ORCID iDs

Patrick Emami https://orcid.org/0000-0001-7846-5578
Peter St. John https://orcid.org/0000-0002-7928-3722

## References

Angermueller C, Dohan D, Belanger D, Deshpande R, Murphy K and Colwell L 2019 Model-based reinforcement learning for biological sequence design *Int. Conf. on Learning Representations*

Angermüller C, Belanger D, Gane A, Mariet Z, Dohan D, Murphy K, Colwell L and Sculley D 2020 Population-based black-box optimization for biological sequence design *Proc. 37th Int. Conf. on Machine Learning (Virtual Event, 13–18 July 2020) (ICML 2020)* vol 119 (PMLR) pp 324–34

Arnold F H 1998 Design by directed evolution *Acc. Chem. Res.* **31** 125–31

Bengio Y, Léonard N and Courville A 2013 Estimating or propagating gradients through stochastic neurons for conditional computation (arXiv:1308.3432)

Biswas S, Khimulya G, Alley E C, Esvelt K M and Church G M 2021 Low-n protein engineering with data-efficient deep learning *Nat. Methods* **18** 389–96

Brookes D H and Listgarten J 2018 Design by adaptive sampling (arXiv:1810.03714)

Brookes D H, Park H and Listgarten J 2019 Conditioning by adaptive sampling for robust design *Proc. 36th Int. Conf. on Machine Learning (ICML 2019) (Long Beach, CA, USA, 9–15 June 2019)* vol 97, ed K Chaudhuri and R Salakhutdinov (PMLR) pp 773–82 (available at: http://proceedings.mlr.press/v97/brookes19a.html)

Castro E, Godavarthi A, Rubinfien J, Givechian K B, Bhaskar D and Krishnaswamy S 2022 ReLSO: a transformer-based model for latent space optimization and generation of proteins (available at: https://arxiv.org/abs/2201.09948)

Chan A, Madani A, Krause B and Naik N 2021 Deep extrapolation for attribute-enhanced generation *Adv. Neural Inf. Process. Syst.* **34** 14084–96

Costello Z and Garcia Martin H 2019 How to hallucinate functional proteins (arXiv:1903.00458)

Dallago C, Mou J, Johnston K E, Wittmann B J, Bhattacharya N, Goldman S, Madani A and Yang K K 2021 Flip: benchmark tasks in fitness landscape inference for proteins *bioRxiv Preprint* (available at: https://doi.org/10.1101/2021.11.09.467890)

Dathathri S, Madotto A, Lan J, Hung J, Frank E, Molino P, Yosinski J and Liu R 2020 Plug and play language models: a simple approach to controlled text generation *8th Int. Conf. on Learning Representations (ICLR 2020) (Addis Ababa, Ethiopia, 26–30 April 2020)* (OpenReview.net) (available at: https://openreview.net/forum?id=H1edEyBKDS)

Fannjiang C and Listgarten J 2020 Autofocused oracles for model-based design *Advances in Neural Information Processing Systems 33: Annual Conf. on Neural Information Processing Systems 2020 (Virtual, 6–12 December 2020) (NeurIPS 2020)* ed H Larochelle, M'A Ranzato, R Hadsell, M-F Balcan and H-T Lin (available at: https://proceedings.neurips.cc/paper/2020/hash/972cda1e62b72640cb7ac702714a115f-Abstract.html)

Ferruz N, Schmidt S and Höcker B 2022 A deep unsupervised language model for protein design (available at: http://biorxiv.org/lookup/doi/10.1101/2022.03.09.483666)

Gligorijevic V, Berenberg D, Stephen R, Watkins A, Kelow S, Cho K and Bonneau R 2021 Function-guided protein design by deep manifold sampling (available at: http://biorxiv.org/lookup/doi/10.1101/2021.12.22.473759)

Gomez-Bombarelli R, Wei J N, Duvenaud D, Miguel Hernández-Lobato J, Sánchez-Lengeling B, Sheberla D, Aguilera-Iparraguirre J, Hirzel T D, Adams R P and Aspuru-Guzik A 2018 Automatic chemical design using a data-driven continuous representation of molecules *ACS Cent. Sci.* **4** 268–76

Grathwohl W, Swersky K, Hashemi M, Duvenaud D and Maddison C J 2021 Oops I took a gradient: scalable sampling for discrete distributions *Proc. 38th Int. Conf. on Machine Learning (ICML 2021) (Virtual Event, 18–24 July 2021)* vol 139, ed M Meila and T Zhang (PMLR) pp 3831–41 (available at: http://proceedings.mlr.press/v139/grathwohl21a.html)

Gupta A and Zou J 2018 Feedback GAN (FBGAN) for DNA: a novel feedback-loop architecture for optimizing protein functions (arXiv:1804.01694)

Hansen N and Ostermeier A 1996 Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation *Proc. IEEE International Conf. on Evolutionary Computation* (IEEE) pp 312–7

Hastings W K 1970 Monte Carlo sampling methods using Markov chains and their applications *Biometrika* **57** 97–109

Hawkins-Hooker A, Depardieu F, Baur S, Couairon G, Chen A, Bikard D and Orengo C A 2021 Generating functional protein variants with variational autoencoders *PLoS Comput. Biol.* **17** e1008736

Hesslow D, Zanichelli No, Notin P, Poli I and Marks D RITA: a study on scaling up generative protein sequence models 2022 (arXiv:2205.05789)

Hinton G E 2002 Training products of experts by minimizing contrastive divergence *Neural Comput.* **14** 1771–800

Holtzman A, Buys J, Forbes M, Bosselut A, Golub D and Choi Y 2018 Learning to write with cooperative discriminators (arXiv:1805.06087)

Hopf T A, Ingraham J B, Poelwijk F J, Charlotta P I Sarfe, Springer M, Sander C and Marks D S 2017 Mutation effects predicted from sequence co-variation *Nat. Biotechnol.* **35** 128–35

Hsu C, Nisonoff H, Fannjiang C and Listgarten J 2022 Learning protein fitness models from evolutionary and assay-labeled data *Nat. Biotechnol.* **40** 1114–22

Jain M *et al* 2022 Biological sequence design with gflownets *Int. Conf. on Machine Learning* (PMLR) pp 9786–801

Jang E, Shixiang G and Poole B 2016 Categorical reparameterization with gumbel-softmax (arXiv:1611.01144)

Killoran N, Lee L J, Delong A, Duvenaud D and Frey B J 2017 Generating and designing DNA with deep generative models (arXiv:1712.06148)

Kuchner O and Frances H A 1997 Directed evolution of enzyme catalysts *Trends Biotechnol.* **15** 523–30

Kumar A and Levine S 2020 Model inversion networks for model-based optimization *Advances in Neural Information Processing Systems 33: Annual Conf. on Neural Information Processing Systems 2020 (NeurIPS 2020) (Virtual, 6–12 December 2020)* ed H Larochelle, M'A Ranzato, R Hadsell, M-F Balcan and H-T Lin (available at: https://proceedings.neurips.cc/paper/2020/hash/373e4c5d8edfa8b74fd4b6791d0cf6dc-Abstract.html)

Li C, Zhang R, Wang J, Marie Wilson L M and Yan Y 2020 Protein engineering for improving and diversifying natural product biosynthesis *Trends Biotechnol.* **38** 729–44

Lin Z *et al* 2022 Language models of protein sequences at the scale of evolution enable accurate structure prediction *bioRxiv Preprint* (available at: https://doi.org/10.1101/2022.07.20.500902)
Linder J, Bogard N, Rosenberg A B and Seelig G 2020 A generative neural network for maximizing fitness and diversity of synthetic DNA and protein sequences *Cell Syst.* **11** 49–62.e16
Madani A, McCann B, Naik N, Shirish Keskar N, Anand N, Eguchi R R, Huang P-S and Socher R 2020 ProGen: language modeling for protein generation (arXiv:2004.03497)
Maddison C J, Mnih A and Whye Teh Y 2016 The concrete distribution: a continuous relaxation of discrete random variables (arXiv:1611.00712)
Meier J, Rao R, Verkuil R, Liu J, Sercu T and Rives A 2021 Language models enable zero-shot prediction of the effects of mutations on protein function (available at: http://biorxiv.org/lookup/doi/10.1101/2021.07.09.450648)
Metropolis N, Rosenbluth A W, Rosenbluth M N, Teller A H and Teller E 1953 Equation of state calculations by fast computing machines *J. Chem. Phys.* **21** 1087–92
Nesterov Y 1998 *Introductory Lectures on Convex Programming Volume I: Basic course* pp 1–212
Nguyen A, Clune J, Bengio Y, Dosovitskiy A and Yosinski J 2017 Plug & play generative networks: conditional iterative generation of images in latent space *2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2017) (Honolulu, HI, USA, 21–26 July 2017)* (IEEE Computer Society) pp 3510–20 (available at: https://doi.org/10.1109/CVPR.2017.374)
Nijkamp E, Ruffolo J, Weinstein E N, Naik N and Madani A 2022 ProGen2: exploring the boundaries of protein language models (available at: https://arxiv.org/abs/2206.13517)
Notin P, Dias M, Frazer J, Marchena-Hurtado J, Gomez A, Marks D S and Gal Y 2022 Tranception: protein fitness prediction with autoregressive transformers and inference-time retrieval *Int. Conf. on Machine Learning* pp 16990–7017 (available at: https://proceedings.mlr.press/v162/notin22a.html)
Qin L, Welleck S, Khashabi D and Choi Y 2022 COLD decoding: energy-based constrained text generation with langevin dynamics (arXiv:2202.11705)
Rao R M, Liu J, Verkuil R, Meier J, Canny J, Abbeel P, Sercu T and Rives A 2021 MSA transformer *Int. Conf. on Machine Learning* (PMLR) pp 8844–56
Riesselman A J, Ingraham J B and Marks D S 2018 Deep generative models of genetic variation capture the effects of mutations *Nat. Methods* **15** 816–22
Rives A *et al* 2021 Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences *Proc. Natl Acad. Sci.* **118** e2016239118
Shin J-E, Riesselman A J, Kollasch A W, McMahon C, Simon E, Sander C, Manglik A, Kruse A C and Marks D S 2021 Protein design and variant prediction using autoregressive generative models *Nat. Commun.* **12** 2403
Sinai S, Wang R, Whatley A, Slocum S, Locane E and Kelsic E D 2020 AdaLead: a simple and robust adaptive greedy search algorithm for sequence design (available at: https://arxiv.org/abs/2010.02141)
Sun H, Dai H, Dai B, Zhou H and Schuurmans D 2022a Discrete Langevin sampler via Wasserstein gradient flow (available at: https://arxiv.org/abs/2206.14897)
Sun H, Dai H, Xia W and Ramamurthy A 2022b Path auxiliary proposal for mcmc in discrete space *Int. Conf. on Learning Representations*
Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I J and Fergus R 2014 Intriguing properties of neural networks *2nd Int. Conf. on Learning Representations (ICLR 2014) (Banff, AB, Canada, 14–16 April 2014)* ed Y Bengio and Y LeCun (Conference Track Proceedings) (available at: http://arxiv.org/abs/1312.6199)
Trabucco B, Geng X, Kumar A and Levine S 2022 Design-bench: benchmarks for data-driven offline model-based optimization (available at: https://arxiv.org/abs/2202.08450)
Trabucco B, Kumar A, Geng X and Levine S 2021 Conservative objective models for effective offline model-based optimization *Proc. 38th Int. Conf. on Machine Learning (ICML 2021) (Virtual Event, 18–24 July 2021)* vol 139, ed M Meila and T Zhang (PMLR) pp 10358–68 (available at: http://proceedings.mlr.press/v139/trabucco21a.html)
Weinstein E N, Amin A N, Frazer J and Marks D S 2022 Non-identifiability and the blessings of misspecification in models of molecular fitness *Advances in Neural Information Processing Systems* vol 35 eds A H Oh, A Agarwal, D Belgrave and K Cho pp 5484–97 (available at: https://proceedings.neurips.cc/paperõfiles/paper/2022/hash/247e592848391fe01f153f179c595090-Abstract-Conference.html)
Wu Z, Johnston K E, Arnold F H and Yang K K 2021 Protein sequence design with deep generative models *Curr. Opin. Chem. Biol.* **65** 18–27
Yang K K, Lu A X and Fusi N 2022 Convolutions are competitive with transformers for protein sequence pretraining (available at: http://biorxiv.org/lookup/doi/10.1101/2022.05.19.492714)
Yang K K, Zachary W and Arnold F H 2019 Machine-learning-guided directed evolution for protein engineering *Nat. Methods* **16** 687–94
Zanella G 2020 Informed proposals for local mcmc in discrete spaces *J. Am. Stat. Assoc.* **115** 852–65
Zhang D, Jie F, Bengio Y and Courville A 2022a Unifying likelihood-free inference with black-box optimization and beyond *Int. Conf. on Learning Representations*
Zhang R, Liu X and Liu Q 2022b A Langevin-like sampler for discrete distributions *Int. Conf. on Machine Learning* (PMLR) pp 26375–96