Bayesian Optimisation for Protein Sequence Design: Gaussian Processes with Zero-Shot Protein Language Model Prior Mean

Carolin Benjamins
Leibniz University Hannover

Shikha Surana
InstaDeep Ltd.

Oliver Bent
InstaDeep Ltd.

Marius Lindauer
Leibniz University Hannover
L3S Research Center

Paul Duckworth[*]
InstaDeep Ltd.

Abstract

Bayesian optimisation (BO) is a popular sequential decision making approach for maximising black-box functions in low-data regimes. It can be used to find highly-fit protein sequence candidates since gradient information is not available *in vitro*. Recent *in silico* protein design methods have leveraged large pre-trained protein language models (PLMs) as fitness predictors. However PLMs have a number of shortcomings for sequential design tasks: i) their current capability to model uncertainty, ii) no closed-form Bayesian updates in light of new experimental data, and iii) the challenge of fine-tuning on small down-stream task datasets. We take a step back to traditional BO by using Gaussian process (GP) surrogate models with sequence kernels, which are able to properly model uncertainty and update their belief over multi-round design tasks. In this work we empirically demonstrate that BO with GP surrogates is consistent with large pre-trained PLMs on the multi-round sequence design benchmark ProteinGym. Furthermore, we demonstrate improved performance by augmenting the GP with the strong zero-shot PLM predictions as a GP prior mean function, and show that by using a learned linear combination of zero-shot PLM and constant prior mean the GP surrogate can regulate the effects of the PLM guided prior.

1 Introduction

Evolution encodes functional information within amino acid sequences. Understanding the effects of sequence mutations on function is a fundamental problem for designing new proteins. Traditionally, directed evolution [Arnold, 1998] methods have been employed to sequentially optimise proteins to obtain better characteristics, where each round consists of random mutagenesis, gene recombinations, screening and selection. This process is slow, labour-intensive, and requires costly *in-vitro* infrastructure. Consequently, it is of interest to explore efficient *in-silico* sequence design methods.

Labeled protein sequence datasets that link amino acid sequences to quantitative measurements of relevant biological properties are increasingly available [Rao et al., 2019, Dallago et al., 2021, Trabucco et al., 2022, Notin et al., 2023a, Groth et al., 2023]. However, due to experimental limitations during dataset creation, it may only be practical to obtain measurements for dozens or a few hundred proteins at a time [Biswas et al., 2021]. To this end, the focus is on design methodologies that efficiently use small labelled datasets to accelerate real-world experimentation.

[*]corresponding email: p.duckworth@instadeep.com

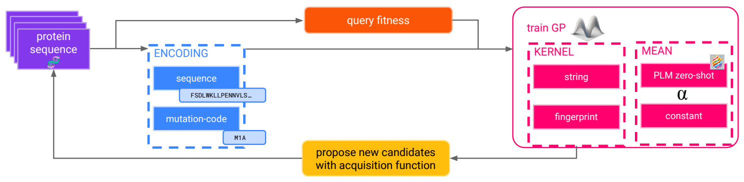


Figure 1: Traditional Bayesian optimization design loop using Gaussian processes for protein sequence design with string or fingerprint kernels and a constant prior mean, zero-shot predictions of a protein language model as prior mean or a learned interpolation thereof.

A concurrent trend in protein sequence design considers large pre-trained protein language models (PLMs), which take string sequences as input, and fine-tune task-specific downstream predictive models; a review of the field is provided by Ruffolo and Madani [2024]. However, whilst being strong zero-shot and few-shot supervised fitness predictors, PLMs have a number of shortcomings for design routines, such as: they are computationally expensive to train and fine-tune; they have no closed-form or efficient Bayesian update rule to incorporate new experimental data; they lack formal methods to model predictive uncertainty and do not generalize well outside of training data distributions [Gruver et al., 2021, Greenman et al., 2023, Tagasovska et al., 2024]; and fine-tuning large models on small downstream task data is often challenging due to hyperparameters and overfitting.

In this work, we take a step back to simplicity and consider Gaussian process (GP) surrogates with string and fingerprint kernels suitable for large protein sequences [Griffiths et al., 2023]. Whilst GP surrogates have been used to optimise proteins and antibody designs previously [Belanger et al., 2019, Khan et al., 2023] via Bayesian optimization (BO) [Mockus, 1989, Garnett, 2023], those works specifically focus on small regions of the sequence (at most sequence length 11), and not the high dimensional sequence design setting required for real world protein design, which can be hundreds of amino acids. For an overview of high-dimensional BO literature, see Kim et al. [2021], Santoni et al. [2024], González-Duque et al. [2024].

Gaussian processes (GPs) explicitly incorporate prior knowledge via the choice of kernel function and prior mean. In this work, we propose to combine the GP with the knowledge encoded in a large pre-trained evolutionary scale PLM, e.g. ESM-2 [Lin et al., 2022]. Given that PLMs are strong zero-shot fitness predictors [Meier et al., 2021, Riesselman et al., 2018], we augment the GP surrogate with these predictions via the prior mean function. Additionally, we use a weighted linear combination between the PLM prior mean and a constant mean, where the weight is optimised during training. This allows the GP surrogate to regulate the PLM zero-shot predictions. Our method combines the strengths of two paradigms: (i) uncertainty guided acquisitions via the GP and (ii) strong zero-shot priors encoded via the PLM. We evaluate our method on eight multi-round ProteinGym single-mutant landscapes from Notin et al. [2023a] and show that GPs with appropriate kernels perform consistent or better than fine-tuning ESM-2 in each round. We also demonstrate that adding the zero-shot predictions from ESM-2 as a prior mean in a weighted linear combination with a traditional constant prior mean yields improved performance compared to a single prior mean function.

2 Methodology

Bayesian optimization (BO) [Mockus, 1989] is a sequential decision making approach that maximises black-box functions in a sample-efficient way. For black-box functions, we can only observe the output for a given candidate, but not gradients. Generally in Bayesian optimization (BO), the probabilistic surrogate model (here, a GP with fingerprint or string kernel) is an iteratively refined approximation of the black-box problem that guides the optimisation process. A crucial property of GPs is that they provide an approximate uncertainty of the underlying function at unobserved candidates. In each iteration, the acquisition function, which is a utility function that trades off exploration and exploitation, proposes the next candidates to evaluate based on the surrogate model predictions and uncertainty. In the case of proposing one candidate per round, it is the candidate with the highest acquisition function value. In the batch setting, several candidates are proposed leveraging via some batch criteria [Desautels et al., 2014, González et al., 2016, Wu and Frazier, 2016, Neiswanger et al., 2022]. The surrogate model is then updated with the new observation(s), and the current best candidate is updated. These steps are usually repeated for a given overall optimisation budget.

Whilst PLMs are the leading approach for protein sequence fitness prediction [Meier et al., 2021, Truong Jr and Bepler, 2023, Notin et al., 2023b], they have limitations for multi-round design settings. For that reason, instead of PLMs we consider GP models to guide the design. They are the default choice in BO [Mockus, 1989, Garnett, 2023] and provide principled uncertainty estimates.

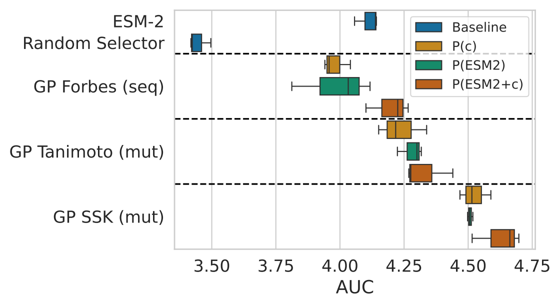
In this section, we first define the problem setting, then describe protein encodings and how to translate the encodings for the choice of kernel functions. We then introduce injecting a prior mean function guided by PLM zero-shot predictions, and our linear combination of prior means.

Problem setting Formally, we aim to find the candidate $x^* \in \mathcal{X}$ in the design space $\mathcal{X}$ with the maximum fitness of a given black-box function $f : \mathcal{X} \to \mathbb{R}$: $x^* \in \arg\max_{x \in \mathcal{X}} f(x)$. In our case, it is finding the protein sequence candidate with the highest fitness with the fewest number of trials. Protein sequence optimisation in high-dimensions is challenging due to the large combinatorial design space, with 20^l possible sequences of length l and 20 different amino-acids (AAs).

Sequence encoding As proteins are often represented as a string of hundreds of AAs, they need encoding to obtain a numerical representation. We utilise single-mutant protein landscapes from ProteinGym [Notin et al., 2023a] and investigate two different sequence representations: (i) the *sequence* string of AA tokens (between 86 and 724 in length), and (ii) the *mutation-code* to wild-type sequence. The mutation-code represents a single mutation relative to a wild-type sequence as a tuple of the AA to be replaced, the position of the mutation, and the new AA, e.g. `M12A`, resulting in a compact representation of symbols instead of a long sequence of mostly repeated AAs. To the best of our knowledge, we do not know of any work using mutation-codes as encoding for protein sequence design. In order to use fingerprint kernels (see below), we convert the sequence representation into binary vectors, where each AA in the sequence is replaced by its index in the AA alphabet. The integer index is then represented as binary, e.g. for a sequence of length 86 AAs we obtain a binary vector of length $86 \cdot 5$ (number of digits to represent the 20 AA library in binary form).

GP kernels A Gaussian process (GP) is a stochastic process that specifies the full distribution over the black-box function $f(x) \sim \mathcal{N}(m(x), k(x, x'))$, with $m(x)$ as its mean function and $k(x, x')$ as its covariance function or *kernel*. The choice of the kernel determines the majority of the generalization properties, enabling posterior model uncertainty and robust out-of-distribution behaviour. The kernel should be specified such that similar candidates yield similar predictions [Rasmussen and Williams, 2006]. There are specific kernels for fingerprints, strings, and graphs [Griffiths et al., 2023]. String kernels are directly applicable to both encoding types: sequence and mutation-code. Molecular fingerprints usually are an enumeration of subgraphs that are hashed into a binary vector. As fingerprint kernels we use Dice, Forbes [Forbes, 1925], Innerproduct, Intersection, Minmax, Otsuka, Russell-Rao, Sorgenfrei and Tanimoto. As string kernel we use the subsequence string kernel (SSK) [Lodhi et al., 2002, Beck and Cohn, 2017, Moss et al., 2020]. Although the SSK on classic protein sequences has the drawbacks of being computationally inefficient and not considering the positions of subsequences [Stanton et al., 2022], this is alleviated by using the mutation-code representation, which is a drastically shorter representation and contains the position of the mutation. All kernels are available in GAUCHE [Griffiths et al., 2023]. For more details see Appendix A.2.

GP prior mean The mean function of a GP is the expected value of the function we aim to optimise $m(x) = \mathbb{E}[f|x]$. The choice of prior mean receives little attention as the kernel or the covariance function mainly determines the sampling behaviour [Garnett, 2023]. In addition, the prior mean influences the posterior only via the posterior mean. This means that the posterior mean is mainly determined by the data in areas where the observed data highly correlates with the GP function values. On the other hand, in regions where the correlation is close to zero and the distance of x from the observed data increases, the posterior mean resembles the prior mean [De Ath et al., 2020, Garnett, 2023]. Usually the prior mean is a constant c and initialised to be zero everywhere: $m(x) = c$. We standardise the observations following general practice, thus the mean function becomes the arithmetic mean of the observed function values [De Ath et al., 2020]. The constant is optimised during training alongside any kernel parameters. Instead of a constant mean, a general model encoding prior knowledge can be leveraged, e.g. when optimising physics problems [Ziatdinov et al., 2022]. Note that the prior mean does not influence the kernel and thus the uncertainty, but rather it biases the posterior predictions.



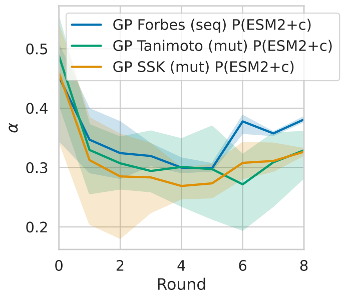


Figure 2: **Left:** AUC over all landscapes. In general, Tanimoto and SSK kernel on mutant encoding show better AUC than ESM-2. Using ESM-2 as prior mean (`P(ESM2)`) mostly improves on constant mean (`P(c)`). The weighted linear combination of PLM prior mean and constant mean (`P(ESM2+c)`) with the ability to recover from either generally shows best performance. **Right:** Weight of linear combination α between zero-shot PLM prior and constan mean prior on BLAT-ECOLX-Jacquier-2013. α is learned and changed throughout the design.

PLM-augmented prior mean We propose to leverage zero-shot predictions from a large pre-trained PLM. The cost of which is only incurred at inference time, and no fine-tuning of the PLM is required. The zero-shot predictions of PLMs trained with a masked-language modelling objective, e.g. ESM-2 [Lin et al., 2022], are obtained with the masked-marginals heuristic and publicly provided in ProteinGym [Notin et al., 2023a]. One would expect the quality of the GP guided optimisation is linked to the quality of the PLM zero-shot predictions. As the PLM is not fine-tuned during the design we instead propose to interpolate between zero-shot PLM predictions f_{PLM} and a learnable constant mean. The interpolation parameter α is also learned during optimisation. We formulate our interpolated prior mean as follows:

$$m(x) = \alpha \cdot f_{\text{PLM}}(x) + (1 - \alpha) \cdot c\,. \tag{1}$$

We compare three variants: i) $\alpha = 0$ utilises the standard GP optimisable constant prior mean (with no PLM predictions), ii) $\alpha = 1$ relies entirely on the PLM zero-shot predictions as prior mean, and iii) $\alpha =$ optimised based on the observed data. α is optimised during the GP training in the same way as the constant c of the constant prior mean. In this way, one expects that strong zero-shot knowledge in the PLM can be leveraged whilst also being able to recover from poor predictions.

3 Experiments

Experimental setup We evaluate our BO setup on single-mutant protein sequences available in the ProteinGym benchmark [Notin et al., 2023a]. We follow the evaluation protocol introduced by Notin et al. [2023b] and Hawkins-Hooker et al. [2024]. The optimisation task is as follows: For a given protein "wild-type" sequence, i.e., a given fitness landscape, find the highest-fit mutated sequence within the fewest number of trials. This benchmark focuses on single mutations from the wild-type; that is, our dataset consists of sequences where only one amino acid has been edited compared to the wild-type. As per the design setting introduced in [Notin et al., 2023b], we have 10 optimisation rounds where in each round we acquire 100 new candidates and fitness values to include into the surrogate model training dataset. This setting resembles *in vitro* experiments where batches of newly proposed sequences are sent to a lab for evaluation in each round and the returned fitness values are incorporated into the model. As an acquisition function we use LogEI [Mockus et al., 1978, Ament et al., 2023], which select the candidates from a candidate pool consisting of all single-mutant sequences recorded in the wild-type dataset. We present results using the following kernels: Forbes for sequence encoding, and SSK as a string kernel and Tanimoto as a fingerprint kernel for mutant encoding. SSK is significantly more compute-intensive. Thus, we also compare it to Tanimoto (which runs entirely on CPU). For the performance of additional kernels with constant prior mean see Appendix Table 2. We repeat the experiment for 3 random seeds, and compare the top-30% recall over the wild-type landscape as per Notin et al. [2023b], and report area-under-curve (AUC) (where higher is better). More details on the hyperparameters, hardware, and metrics are provided in Appendices A.3 to A.5.

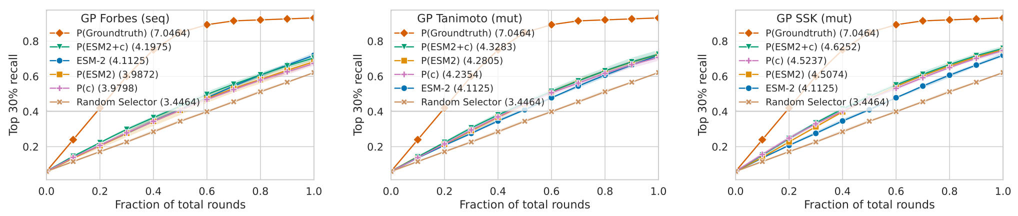


Figure 3: Design Curves (with legend ranked by AUC). Left: GP Forbes on sequence, middle: GP Tanimoto on mutation-code, right: GP SSK on mutation-code. `P(c)`: constant prior mean, `P(ESM2)`: zero-shot ESM2 prior mean, `P(ESM2+c)`: linear interpolation between zero-shot ESM2 prior mean and constant prior mean, `P(Groundtruth)`: Prior mean using ground-truth values.

Baselines We compare against the evolutionary scale model ESM-2 [Lin et al., 2022] as a PLM. ESM-2 is an 8 million parameter pre-trained masked language model. We pass the average sequence embeddings through a linear regression layer to predict fitness, and fine-tune all parameters. We use a greedy acquisition function since ESM-2 provides no measure of uncertainty. Besides the PLM baseline, we also compare to random search. To obtain an upper bound on how well the GP can perform, we also utilise the ground-truth values as a prior mean (referred to as `P(Groundtruth)`).

Protein Design Results In Figure 2, we compare the AUC of all variants averaged over all landscapes. The general trend is that Tanimoto (fingerprint kernel) and SSK (string kernel) with constant prior mean (denoted as `P(c)` in the figure) show higher AUC than ESM-2 – this is the standard setup for BO, and only uses data observed during the design process, whereas ESM-2 has been pre-trained on a large corpus on protein sequences.

Additionally, augmenting the GP with the zero-shot PLM prior mean function (denoted as `P(ESM-2)`) proves to be very useful on some landscapes, whilst hindering the GP on others, see Appendix Figure 4 for per landscape results. We posit that this is directly related to the zero-shot performance of the PLMs predictions on a landscape, i.e., before it is fine-tuned. To combat this, we use a weighted linear combination of the zero-shot PLM prior mean and constant mean (`P(ESM-2+c)`), where the weight is optimised during design, which performs better than only constant prior or zero-shot PLM prior as it can utilise the knowledge encoded in the PLM whilst being able to recover from poor PLM zero-shot predictions. The weight α is not constant throughout the design and varies per landscape (see Figure 2 right or Appendix Figure 5 for all landscapes), indicating that it is beneficial to allow flexibility between the two prior means.

Finally, using the ground-truth data as a prior mean (`P(Groundtruth)`) unsurprisingly yields the strongest performance and shows the potential of using a stronger zero-shot predictive model to inform the prior mean in order to close this, see Figure 3 for design curves (legend ranked by AUC).

4 Limitations and Future Work

One limitation of our method is that it only utilises zero-shot predictions of the PLM. Therefore, we consider fine-tuning the PLM with each batch of acquired data, as per the ESM-2 baseline. In addition, we can also leverage knowledge encoded in a PLM to inform a learned kernel function, as done by Khan et al. [2022]. Another limitation is that the binary representation of the sequence relies on the index of the AA alphabet. This means that the fingerprint kernels are not invariant to the order of the alphabet because of the implied similarity of neighbouring AAs in the alphabet introduced by the order. However, they are able to handle very long sequences. For future work, we plan to extend our methodology to multi-mutant landscapes in ProteinGym. Additionlly, we intend to investigate the exploration-exploitation trade-off specifically maximising information over batches, as explored by Belanger et al. [2019], either by automatically adjusting the trade-off over time [Benjamins et al., 2023], or by employing diversity-generating acquisition functions [Neiswanger et al., 2022].

5 Conclusion

This work identifies a promising research avenue for traditional BO with GPs to tackle multi-round protein sequence design. With appropriate kernels, we demonstrate how longer protein sequences than previously possible can now be optimised. We show that BO with classic GPs outperforms the popular ESM-2 PLM that requires large pre-training datasets and substantial fine-tuning budgets, whilst BO with GPs only requires a fraction of the compute budget. In addition, incorporating the zero-shot predictions of the PLM as a prior mean in a learned linear combination with a constant mean generally improves performance further. We believe probabilistic surrogates hold value, especially in large design spaces where generalisation and uncertainty-driven acquisitions are important.

References

S. Ament, S. Daulton, D. Eriksson, M. Balandat, and E. Bakshy. Unexpected improvements to expected improvement for bayesian optimization. In *NeurIPS 2023*, 2023. 3, A.3

F. Arnold. Design by directed evolution. *Accounts of chemical research*, 31(3):125–131, 1998. 1

M. Balandat, B. Karrer, D. Jiang, S. Daulton, B. Letham, A. Wilson, and E. Bakshy. Botorch: A framework for efficient monte-carlo Bayesian optimization. In Larochelle et al. [2020]. A.3

D. Beck and T. Cohn. Learning kernels over strings using gaussian processes. In *IJCNLP 2017*, pages 67–73, 2017. 2, A.2

D. Belanger, S. Vora, Z. Mariet, R. Deshpande, D. Dohan, C. Angermueller, K. Murphy, O. Chapelle, and L. Colwell. Biological sequence design using batched bayesian optimization. In *NeurIPS 2019 Workshop on Machine Learning and the Physical Sciences*, 2019. 1, 4

C. Benjamins, E. Raponi, A. Jankovic, C. Doerr, and M. Lindauer. Self-adjusting weighted expected improvement for bayesian optimization. In A. Faust, C. White, F. Hutter, R. Garnett, and J. Gardner, editors, *Proceedings of the Second International Conference on Automated Machine Learning*. Proceedings of Machine Learning Research, 2023. 4

S. Biswas, G. Khimulya, E. Alley, K. Esvelt, and G. Church. Low-N protein engineering with data-efficient deep learning. *Nature Methods*, 18(4):389–396, 2021. 1

N. Cancedda, E. Gaussier, C. Goutte, and J.M. Renders. Word sequence kernels. *Journal of Machine Learning Research*, 3:1059–1082, 2003. A.2

C. Dallago, J. Mou, K. Johnston, B. Wittmann, N. Bhattacharya, S. Goldman, A. Madani, and K. Yang. FLIP: Benchmark tasks in fitness landscape inference for proteins. *bioRxiv*, 2021. 1

G. De Ath, J. Fieldsend, and R. Everson. What do you mean? the role of the mean function in Bayesian optimisation. In J. Ceberio, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'20)*, page 1623–1631, 2020. 2

T. Desautels, A. Krause, and J. Burdick. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *JMLR*, 15(1):3873–3923, 2014. 2

S. Forbes. Method of determining and measuring the associative relations of species. *Science*, 61 (1585):518–524, 1925. 2, A.2

R. Garnett. *Bayesian Optimization*. Cambridge University Press, 2023. Available for free at `https://bayesoptbook.com/`. 1, 2

J. González, Z. Dai, P. Hennig, and N. Lawrence. Batch bayesian optimization via local penalization. In *AISTATS 2016*, 2016. 2

M. González-Duque, R. Michael, S. Bartels, Y. Zainchkovskyy, S. Hauberg, and W. Boomsma. A survey and benchmark of high-dimensional bayesian optimization of discrete sequences. *arXiv preprint arXiv:2406.04739*, 2024. 1

John C Gower. A general coefficient of similarity and some of its properties. *Biometrics*, pages 857–871, 1971. A.2

K. Greenman, A. Amini, and K. Yang. Benchmarking uncertainty quantification for protein engineering. *bioRxiv*, 2023. doi: 10.1101/2023.04.17.536962. 1

R. Griffiths, L. Klarner, H. Moss, A. Ravuri, S. Truong, Y. Du, S. Stanton, G. Tom, B. Rankovic, A. Rokkum Jamasb, A. Deshwal, J. Schwartz, A. Tripp, G. Kell, S. Frieder, A. Bourached, A. Chan, J. Moss, C. Guo, J. Peter Dürholt, S. Chaurasia, J. Won Park, F. Strieth-Kalthoff, A. Lee, B. Cheng, A. Aspuru-Guzik, P. Schwaller, and J. Tang. GAUCHE: A library for gaussian processes in chemistry. In *NeurIPS 2023*, 2023. 1, 2, A.2, A.3

P. Groth, R. Michael, P. Tian, J. Salomon, and W. Boomsma. FLOP: Tasks for fitness landscapes of protein families using sequence- and structure-based representations, 2023. 1

Nate Gruver, Samuel Stanton, Polina Kirichenko, Marc Finzi, Phillip Maffettone, Vivek Myers, Emily Delaney, Peyton Greenside, and Andrew Gordon Wilson. Effective Surrogate Models for Protein Design with Bayesian Optimization. In *The 2021 ICML Workshop on Computational Biology*, 2021. 1

A. Hawkins-Hooker, J. Kmec, O. Bent, and P. Duckworth. Likelihood-based fine-tuning of protein language models for few-shot fitness prediction and design. *bioRxiv*, pages 2024–05, 2024. 3

A. Khan, A. Cowen-Rivers, A. Grosnit, D. Deik, P. Robert, V. Greiff, E. Smorodina, P. Rawat, R. Akbar, K. Dreczkowski, R. Tutunov, D. Bou-Ammar, J. Wang, A.s Storkey, and H. Bou-Ammar. Toward real-world automated antibody design with combinatorial Bayesian optimization. *Cell Reports Methods*, 3:100374, 2023. 1

M. Khan, A. Cowen-Rivers, D. Deik, A. Grosnit, K. Dreczkowski, P. Robert, V. Greiff, R. Tutunov, D. Bou-Ammar, Jun Wang, and Haitham Bou-Ammar. Antbo: Towards real-world automated antibody design with combinatorial bayesian optimisation. *CoRR*, abs/2201.12570, 2022. 4

S. Kim, P. Lu, C. Loh, J. Smith, J Snoek, and M. Soljačić. Deep learning for bayesian optimization of scientific problems with high-dimensional structure. *arXiv preprint arXiv:2104.11667*, 2021. 1

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR*, 2015. A.6

H. Larochelle, M. Ranzato, R. Hadsell, M.-F. Balcan, and H. Lin, editors. *Proceedings of the 34th International Conference on Advances in Neural Information Processing Systems (NeurIPS'20)*, 2020. 5

Z. Lin, H. Akin, R. Rao, B. Hie, Z. Zhu, W. Lu, N. Smetanin, A. dos Santos Costa, M. Fazel-Zarandi, T. Sercu, S. Candido, et al. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *bioRxiv*, 2022. 1, 2, 3

H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2(Feb):419–444, 2002. 2, A.2

J. Meier, R. Rao, R. Verkuil, J. Liu, T. Sercu, and A. Rives. Language models enable zero-shot prediction of the effects of mutations on protein function. In *NeurIPS 2021*, volume 34, pages 29287–29303, 2021. 1, 2

J. Mockus. *Bayesian Approach to Global Optimization. Theory and Applications*. Kluwer Academic Publishers, 1989. 1, 2

J. Mockus, V. Tiesis, and A. Zilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2(117-129), 1978. 3, A.3

H. Moss, D. Beck, J. Gonzàles, D. Leslie, and P. Rayson. BOSS: Bayesian optimization over string spaces. In Larochelle et al. [2020]. 2

H. Moss, D. Leslie, and P. Rayson. MUMBO: Multi-task max-value Bayesian optimization. In F. Hutter, K. Kersting, J. Lijffijt, and I. Valera, editors, *Machine Learning and Knowledge Discovery in Databases (ECML/PKDD'20)*, volume 12459, pages 447–462, 2021. A.2

W. Neiswanger, L. Yu, S. Zhao, C. Meng, and S. Ermon. Generalizing bayesian optimization with decision-theoretic entropies. In *NeurIPS 2022*, 2022. 2, 4

P. Notin, A. Kollasch, D. Ritter, L. van Niekerk, S. Paul, H. Spinner, N. Rollins, A. Shaw, R. Orenbuch, R. Weitzman, J. Frazer, M. Dias, D. Franceschi, Y. Gal, and D. Marks. ProteinGym: Large-scale benchmarks for protein fitness prediction and design. In *Advances in Neural Information Processing Systems*, volume 36, pages 64331–64379, 2023a. 1, 2, 3, A.1, 1

P. Notin, R. Weitzman, D. Marks, and Y. Gal. Proteinnpt: Improving protein property prediction and design with non-parametric transformers. In *NeurIPS 2023*, volume 36, pages 33529–33563, 2023b. 2, 3, A.1, A.6

R. Rao, N. Bhattacharya, N. Thomas, Y. Duan, P. Chen, J. Canny, P. Abbeel, and Y. Song. Evaluating Protein Transfer Learning with TAPE. In *NeurIPS 2019*, volume 32, 2019. 1

C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006. 2

Adam J. Riesselman, John B. Ingraham, and Debora S. Marks. Deep generative models of genetic variation capture the effects of mutations. *Nature Methods*, 15(10):816–822, 2018. URL https://www.nature.com/articles/s41592-018-0138-4. 1

J. Ruffolo and A. Madani. Designing proteins with language models. *nature biotechnology*, 42(2): 200–202, 2024. 1

M. Santoni, E. Raponi, R. Leone, and C. Doerr. Comparison of high-dimensional bayesian optimization algorithms on bbob. *ACM Transactions on Evolutionary Learning*, 4(3):1–33, 2024. 1

S. Stanton, W. Maddox, N. Gruver, P. Maffettone, E. Delaney, P. Greenside, and A.G. Wilson. Accelerating Bayesian Optimization for Biological Sequence Design with Denoising Autoencoders. In *ICML*, pages 20459–20478, 2022. 2

N. Tagasovska, J. Park, M. Kirchmeyer, N. Frey, A. Watkins, A. Ismail, A. Jamasb, E. Lee, T. Bryson, S. Ra, et al. Antibody domainbed: Out-of-distribution generalization in therapeutic protein design. In *ICLR*, 2024. 1

B. Trabucco, X. Geng, A. Kumar, and S. Levine. Design-bench: Benchmarks for data-driven offline model-based optimization. In *ICML*, pages 21658–21676, 2022. 1

T. Fei Truong Jr and T. Bepler. PoET: A generative model of protein families as sequences-of-sequences. In *NeurIPS*, 2023. 2

J. Wu and P. Frazier. The parallel knowledge gradient method for batch Bayesian optimization. In D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, 2016. 2

M. Ziatdinov, A. Ghosh, and S. Kalinin. Physics makes the difference: Bayesian optimization and active learning via augmented gaussian process. *Mach. Learn. Sci. Technol.*, 3(1):15003, 2022. 2

A Appendix

A.1 Protein landscapes

We use the set of eight single-mutant landscapes selected for ablations and hyperparameter selection in Notin et al. [2023b] included in ProteinGym [Notin et al., 2023a], see Table 1 for an overview of each landscape.

Table 1: Single-mutant Protein Landscapes from ProteinGym [Notin et al., 2023a]

Landscape	Sequence Length	Number of Sequences
TAT-HV1BR-Fernandes-2016	86	1577
REV-HV1H2-Fernandes-2016	116	2147
RL40A-YEAST-Roscoe-2013	128	1195
CALM1-HUMAN-Weile-2017	149	1813
DYR-ECOLI-Thompson-2019	159	2363
BLAT-ECOLX-Jacquier-2013	286	989
P53-HUMAN-Giacomelli-2018-WT-Nutlin	393	7467
DLG4-RAT-McLaughlin-2012	724	1576

A.2 String and Fingerprint Kernels

The kernel $k(x, x')$ measures the similarity between inputs and acts as an inductive bias over the underlying optimisation landscape. Kernel functions have hyperparameters, such as the lengthscale, which can be optimised during model training. String kernels operate on strings or a sequence of symbols and mostly compare the similarity of their sub-strings. In our case the string is either a sequence of an alphabet of 20 amino acids (length l up to hundreds) or the mutation-code (length $l = 3$) of an alphabet of 20 amino acids plus the number of possible mutation locations. More formally, our alphabet $\mathcal{A}$ is the set of k symbols $\mathcal{A} = \{s_0, ..., s_k\}$ ($|\mathcal{A}| = k$). Then our vector space is $\mathcal{A}^l$ with l being the sequence length. Each element $x \in \mathcal{A}^l$ is a vector $x = (x_0, x_1, ..., x_l)$ with $x_i \in \mathcal{A} \, \forall i = 1, 2, ..., l$.

The subsequence string kernel (SSK) [Lodhi et al., 2002, Cancedda et al., 2003, Beck and Cohn, 2017, Moss et al., 2021] we use as provided by GAUCHE [Griffiths et al., 2023] compares sub-strings of length n (set to $n = 5$). The sub-sequences are used as features and can be non-contiguous. An SSK (n-th) order between to strings x and x' is defined as:

$$k_{\text{SSK}}(x, x') = \sum_{u \in \mathcal{A}^n} c_u(x) \cdot c_u(x')$$

with

$$c_u(s) = \lambda_m^{|u|} \sum_{1 < i_1 < \cdots < i_{|u|} < |s|} \lambda_g^{i_{|u|} - i_1} \mathbb{I}_u((s_{i_1}, \ldots, s_{i_{|u|}})) \, ,$$

where $\mathcal{A}^n$ is the set of all possible ordered collections containing up to n characters from the alphabet $\mathcal{A}$, $\nVdash_x(x')$ the indicator function if strings x and x' are equal. The contribution of a subsequence u to a string s is measured by $c_u(s)$. The kernel hyperparameters are the match decay $\lambda_m \in [0, 1]$ and the gap decay $\lambda_g \in [0, 1]$. They control the weighting of long and/or highly non-contiguous sub-strings. The kernel is normalized $\tilde{k}_{\text{SSK}}(x, x') = k_{\text{SSK}}(x, x') / \sqrt{k_{\text{SSK}}(x, x) k_{\text{SSK}}(x', x')}$ to be able to meaningfully compare strings of varied lengths.

Fingerprint kernels do not operate on the alphabet of symbols but on a binary vector, thus $x, x' \in \{0, 1\}^d$ (d length of vector). For example, the Forbes kernel [Forbes, 1925] is defined as

$$k_{\text{Forbes}}(x, x') = \sigma_f^2 \cdot \frac{d \cdot \langle x, x' \rangle}{\|x\| + \|x'\|}$$

with $\|\cdot\|$ the Euclidean norm.

The Tanimoto kernel [Gower, 1971] is defined as

$$k_{\text{Tanimoto}}(x, x') = \sigma_f^2 \cdot \frac{\langle x, x' \rangle}{\|x\|^2 + \|x'\|^2 - \langle x, x' \rangle} \, .$$

A.3 Hyperparameters

For our BO with GPs we use Botorch [Balandat et al., 2020] with, LogEI [Mockus et al., 1978, Ament et al., 2023] and kernels from GAUCHE [Griffiths et al., 2023]. We standardize the response values and use standard hyperparameters. The prior mean is standardised as well.

A.4 Hardware

For our experiments we used NVIDIA-A100-SXM4-80GB GPU for the PLM baselines and BO with the SSK kernel. For the remaining experiments we used AMD EPYC 7452 CPUs.

A.5 Metrics

Top 30% Recall We define the top 30% recall as follows: The number of acquired candidates who have a fitness value higher equal the threshold divided by the number of candidates in the pool with a fitness value higher than the threshold. The threshold marks the lowest fitness value of the top 30% of all candidates in the pool. It represents how many relevant items have been retrieved so far. For the design curves aggregating results over all tasks we first average over tasks and plot the 95%-CI over the seeds. To account for the different number of rounds we normalize the rounds and interpolate the values accordingly. The per-task design curves are based on the raw data.

AUC We calculate the area-under-curve (AUC) as the integral of the top 30% recall over rounds. For the aggregation over tasks we average the top 30% recall over tasks first and then calculate AUC per seed.

A.6 Baseline model

Hyperparameters for the fine-tuning PLM method are consistent with the practice used to select hyperparameters for the baselines from ProteinNPT [Notin et al., 2023b].

ESM-2 was fine-tuned using the Adam optimiser [Kingma and Ba, 2015] using gradient accumulation with an effective batch size of 32. Learning rates were selected in each case after a sweep over the values $1e-4, 3e-5, 1e-5$ on the eight single mutant landscapes. Linear regression heads were added to embeddings extracted from ESM-2. We averaged embeddings across the sequence dimension before feeding them to the regression head.

A.7 Additional Results

AUC per Task

The AUC per task (Figure 4) differs per landscape. Some landscapes, like P53-HUMAN-Giacomelli-2018-WT-Nutlin and REV-HV1H2-Fernandes-2016, are more difficult to optimise with the chosen methods. Mostly, the GP surrogates perform better than ESM-2. Table 2 shows the final mean AUC averaged over all landscapes and seeds.

Linear Combination Weight per Task

Figure 5 shows the weight α of the interpolation between zero-cost PLM prior mean and standard constant mean. α is learned and thus changed during the design process. Similar trends are observed for different choices of kernels and encodings. Depending on the landscape, α either increases towards relying more on the PLM prior mean (e.g., RL40A-YEAST-Roscoe-2013) or decreases, relying more on the constant prior mean (e.g., BLAT-ECOLX-Jacquier-2013). Interestingly, for DYR-ECOLI-Thompson, α is mostly zero meaning only the constant prior mean is considered.

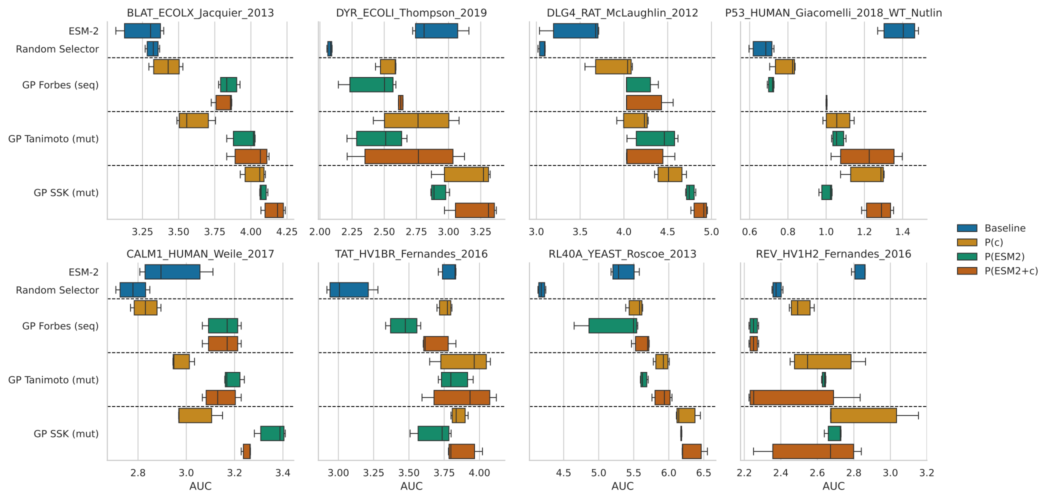


Figure 4: AUC per Task

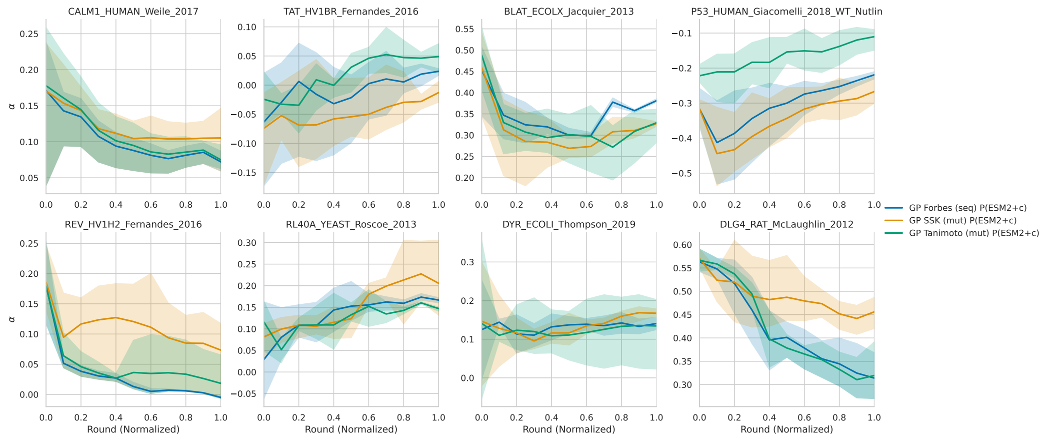


Figure 5: Weight of linear combination α between zero-shot PLM prior and constant prior mean. α is learned and changed throughout the design and its trajectory depends on the landscape.

Table 2: Final mean AUC averaged over all 8 landscapes and 3 seeds of all surrogate models. `seq`: sequence encoding, `mut`: mutation-code, `P(c)`: constant prior mean, `P(ESM2)`: zero-shot ESM2 prior mean, `P(ESM2+c)`: linear interpolation between zero-shot ESM2 prior mean and constant prior mean.

Model	AUC
ESM-2	4.1125
Random Selector	3.4464
GP Forbes (seq) P(Groundtruth)	7.0464
GP Forbes (seq) P(ESM2+c)	4.1975
GP Forbes (seq) P(ESM2)	3.9872
GP Forbes (seq) P(c)	3.9798
GP InnerProduct (seq) P(c)	3.9267
GP Otsuka (seq) P(c)	3.8541
GP MinMax (seq) P(c)	3.6946
GP Tanimoto (seq) P(c)	3.6946
GP Dice (seq) P(c)	3.6945
GP RussellRao (seq) P(c)	3.4050
GP SSK (mut) P(Groundtruth)	7.0464
GP Tanimoto (mut) P(Groundtruth)	7.0464
GP SSK (mut) P(ESM2+c)	4.6252
GP SSK (mut) P(c)	4.5237
GP SSK (mut) P(ESM2)	4.5074
GP Tanimoto (mut) P(ESM2+c)	4.3283
GP Tanimoto (mut) P(ESM2)	4.2805
GP MinMax (mut) P(c)	4.2354
GP Tanimoto (mut) P(c)	4.2354
GP Dice (mut) P(c)	3.9977
GP Forbes (mut) P(c)	3.9762
GP RussellRao (mut) P(c)	3.8959
GP InnerProduct (mut) P(c)	3.8884
GP Otsuka (mut) P(c)	3.8704