Conditional Generative Modeling for De Novo Protein Design with Hierarchical Functions

Tim Kucera [1,*], Matteo Togninalli [2,*,†] and Laetitia Meng-Papaxanthos [3,*,†]

[1]Department of Biosystems Science and Engineering, ETH Zürich, Basel 4058, Switzerland, [2]Visium, Lausanne, 1015, Switzerland and [3]Google Research, Brain Team, Zurich, 8002, Switzerland.

* To whom correspondence should be addressed.

† These authors jointly supervised the work.

Abstract

Motivation: Protein Design has become increasingly important for medical and biotechnological applications. Because of the complex mechanisms underlying protein formation, the creation of a novel protein requires tedious and time-consuming computational or experimental protocols. At the same time, Machine Learning has enabled to solve complex problems by leveraging the large amounts of available data, more recently with great improvements on the domain of generative modeling. Yet, generative models have mainly been applied to specific sub-problems of protein design.

Results: Here we approach the problem of general purpose Protein Design conditioned on functional labels of the hierarchical Gene Ontology. Since a canonical way to evaluate generative models in this domain is missing, we devise an evaluation scheme of several biologically and statistically inspired metrics. We then develop the conditional generative adversarial network ProteoGAN and show that it outperforms several classic and more recent deep learning baselines for protein sequence generation. We further give insights into the model by analysing hyperparameters and ablation baselines. Lastly, we hypothesize that a functionally conditional model could create proteins with novel functions by combining labels and provide first steps into this direction of research.

Availability: Code and data is available at https://github.com/timkucera/proteogan

Contact: tim.kucera@bsse.ethz.ch, mt@visum.ch, lpapaxanthos@google.com

1 Introduction

Designing new proteins with a target biological function is a frequent task in biotechnology and has broad applications in synthetic biology and pharmaceutical research, for example in drug discovery (Huang *et al.*, 2016). The task is challenging because the sequence-structure-function relationship of proteins is extremely complex and not yet fully understood (Dill and MacCallum, 2012). Protein design is therefore mostly done by trial-and-error methods, such as directed evolution (Arnold, 1998), which rely on a few random mutations of known proteins and selective pressure to explore a space of related proteins. This process can be time-consuming and cost-intensive, and most often only explores a small portion of the sequence space. At the same time, data characterizing proteins and their functions is readily available and constitutes a promising opportunity for machine learning applications in protein sequence design.

Multiple generative models have recently been proposed to design proteins for different tasks, such as developing new therapies (Mueller *et al.*, 2018; Davidsen *et al.*, 2019), enzymes (Repecka *et al.*, 2019), nanobody sequences (Riesselman *et al.*, 2019) or proteins that lead to antibiotic resistance (Chhibbar and Joshi, 2019). These models are typically focused on a sub-task of protein design and thus are limited to a given application, often even to a specific protein family. This requires retraining for a new task, but more importantly also does not utilize the entire information contained in the available data. In other domains, such as the closely related natural language generation, one can observe a trend towards general-purpose models that are then used in various contexts (Brown *et al.*, 2020). We posit that, also in protein design, a one-fits-all model may learn common underlying principles across different protein classes improving the quality of generated sequences. Going further, it may even be able to create not only novel sequences, but novel functions by combining different aspects of functionality it has learned in different protein families.

We therefore develop ProteoGAN, a general purpose generative model for conditional protein design based on the Molecular Function Gene Ontology (GO), a hierarchy of labels describing aspects of protein function. These functions vary from binding specific agents, to transporter or sensor activity, catalysis of biochemical reactions and many more. Here, additionally, the information encoded in the hierarchical organisation may help model performance. We base our model on the popular Generative Adversarial Network (GAN) framework, and examine different mechanisms for conditioning on the multilabel hierarchical information.

However, developing such a generative model can be challenging, not least because problem-specific evaluations are lacking. An evaluation metric needs to assess whether a generated sample is valid (i.e. realistic and functional), a hard problem in itself, and further needs to be fast to compute on a large number of samples. The evaluation of generative models is still ongoing research, particularly in the domain of protein design (Papineni *et al.*, 2002; Salimans *et al.*, 2016; Heusel *et al.*, 2017; Shmelkov *et al.*, 2018;

Kynkäänniemi *et al.*, 2019; DeVries *et al.*, 2019). While gold-standard validation of a generated sequence implies the synthesis of the proteins in the lab, the lack of *in-silico* assessments makes it difficult to efficiently compare methods for protein sequence design.

We therefore compose an array of evaluation metrics for generative protein design based on the Maximum Mean Discrepancy (MMD) statistic to measure distributional similarity and conditional consistency of generated sequences with real proteins. We further propose measures to account for sequence diversity.

1.1 Related generative models for protein design

1.1.1 Guided and conditional protein generative models.
Machine learning models and more recently deep generative models (Eddy, 2004; Goodfellow *et al.*, 2014; Kingma and Welling, 2014; Rezende *et al.*, 2014; Vaswani *et al.*, 2017; Li *et al.*, 2017a) have been used to design *in silico* biological sequences, such as RNA, DNA or protein sequences (Durbin *et al.*, 1998; Davidsen *et al.*, 2019; Brookes *et al.*, 2019), often with the aim to create sequences with desired properties. There are two main strategies to achieve this, one is guided and the other conditional. Guided approaches use a predictor (also called oracle) in order to guide the design towards target properties, through iterative training-generation-prediction steps (Brookes *et al.*, 2019; Gane *et al.*, 2019; Angermueller *et al.*, 2019; Gupta and Zou, 2019; Killoran *et al.*, 2017; Repecka *et al.*, 2019). In a scenario with multiple functional labels however, the lack of highly accurate and fast multilabel predictors for protein function impairs guided-generation techniques in functional protein generation (Zhou *et al.*, 2019). On the contrary, conditional approaches integrate the functional information in the generation mechanism itself, eliminating the need for a predictor. For example, Madani *et al.* (2020) developed ProGen, a conditional transformer that enables a controlled generation of a large range of functional proteins, but the need for a sequence context can be experimentally constraining and is not compatible with *de novo* design. Ingraham *et al.* (2019) present a graph-based conditional generative model that relies on structural information, which is only sparsely available. Das *et al.* (2018) and Greener *et al.* (2018) train conditional VAEs in order to generate specific proteins, such as metalloproteins. Karimi *et al.* (2019) used a guided conditional Wasserstein-GAN to generate proteins with novel folds. All these models have in common that they either focus on a sub-task of protein design only, or they rely on context information such as 3D structure or template sequence fragments. We here propose a general purpose model for protein design that only requires to specify the desired functional properties for generation.

1.1.2 Evaluation of generative models
To this date, there is no definitive consensus on the best evaluation measures for the evaluation of *quality*, *diversity* and *conditional consistency* of the output of a (conditional) generative model (Papineni *et al.*, 2002; Salimans *et al.*, 2016; Heusel *et al.*, 2017; Shmelkov *et al.*, 2018; Kynkäänniemi *et al.*, 2019; DeVries *et al.*, 2019). Most measures that stand out in computer vision such as the Inception Score (IS) (Salimans *et al.*, 2016), the Frechet Inception Distance (FID) (Heusel *et al.*, 2017), or GAN-train and GAN-test (Shmelkov *et al.*, 2018) depend on an external, domain-specific predictor. For functional protein design such predictors are neither good nor fast enough to entirely rely on their predictions when evaluating and training neural networks. The Critical Assessment for Functional Annotation (CAFA) challenge reports the currently best model (NetGO) with an F_{max} score of 0.63, which has a prediction speed of roughly 1.000 sequence per hour (F_{max} is the maximal F1-score over confidence thresholds, see their paper or our supplemental section A) (Radivojac *et al.*, 2013; Zhou *et al.*, 2019; You *et al.*, 2019). On the contrary, the domain-agnostic duality gap can be computed during training and at test time, and has been shown to correlate well with FID (Grnarova *et al.*, 2019). Despite the increasing interest of the research community for protein generation, no clear metrics have emerged as reliable tools to compare sequences.

2 System and methods

2.1 Evaluation framework for conditional protein sequence design

Generative models are difficult to evaluate because there is no ground truth one could compare each generated sample with. Instead, the goal of generative modeling is to create data that is similar in its properties but not identical to some target data. Evaluation is further complicated when the data can not be straightforwardly validated, such as in protein design where a generated sample would need to be physically synthesized to prove its functionality. It therefore only remains to assess whether the generated samples are similar to the target data by statistical means or by predicting given aspects of biological plausibility as an estimate of real world performance.

We therefore compose an array of metrics to evaluate conditional generative models for protein design which are based on two-sample goodness-of-fit statistics that can be computed for structured data such as protein sequences and the resulting high-dimensional feature vectors. They have the advantage to be fast to compute and to be differentiable, and can therefore be used during training, for hyperparameter selection, early stopping or as a loss. We corroborate these metrics by comparing the statistics computed with biologically relevant embeddings.

The following sections detail specific aspects of the evaluation and the respective metric we devised.

2.1.1 Evaluating *distribution similarity* with MMD
As generative models aim to model the distribution of target data, it is a natural choice to evaluate them with a statistical two-sample test that compares generated and training data distributions. This approach is difficult to apply to protein sequence data directly but can be applied to extracted feature vectors. We propose to use Maximum Mean Discrepancy (MMD) (Gretton *et al.*, 2012), a test statistic that compares mean embeddings in a Reproducing Kernel Hilbert Space (RKHS). In the past, MMD has been used to infer biological pathways or sequence homology from biological sequences or for distinguishing sets of structured biological sequences (Leslie *et al.*, 2001; Vegas *et al.*, 2016; Borgwardt *et al.*, 2006).

Let $R = \{r_i\}_{i=1}^n$ and $G = \{g_j\}_{j=1}^m$ be samples from the distributions of real and generated proteins sequences, respectively P_r and P_g. Then:

$$\mathrm{MMD}^2(R, G) = \left\| \frac{1}{n} \sum_{i=1}^{n} \phi(r_i) - \frac{1}{m} \sum_{j=1}^{m} \phi(g_j) \right\|_2^2 \tag{1}$$

We decided to use the (normalized) Spectrum kernel (Leslie *et al.*, 2001) with its feature map ϕ since it is fast to compute and sufficiently complex to distinguish protein properties of interest, which we validate in meta evaluations of the metrics in the Discussion section. The feature mapping counts the occurrences of kmers in a sequence ($k = 3$, resulting in 8.000 features). We verify that our measure is robust to the choice of kernel by using an alternative Gaussian kernel.

To confirm our evaluations with the Spectrum kernel feature map we further compute MMD using the biological embeddings ProFET (Ofer and Linial, 2015), a collection of handcrafted features, and UniRep (Alley *et al.*, 2019), a learned protein embedding based on an LSTM recurrent network. We remove kmer related features in ProFET to avoid confounding with our Spectrum kernel based metrics, resulting in ca. 500 features which were then scaled to the same range. UniRep has a dimensionality of 1.900, and we use the mean hidden state over the sequence as the representation.

We further confirm the results computed by the MMD statistic with a second statistic, the Kolmogorov-Smirnov (KS) test, which is more expensive to compute.

2.1.2 Evaluating *conditional consistency* with MRR

For conditional generation specifically, we need to assess the model's capability to generate sequences consistent with some target labels. We extend the MMD metric by computing MMD between subsets of sequences for each label and ranking the RKHS distance between generated samples and their target label among distances to off-target labels. It measures how many sets of real sequences with off-target labels are closer in distribution to the generated sequences than real sequences with the target label.

Let R be a set of real sequences R_i with annotated labels $i \in \{1, ..., d\}$, where d is the total number of labels. Let $G = \{G_i\}_{i=1}^{d}$ be an equally structured set of generated sequences. We want to maximise the following mean reciprocal rank (MRR):

$$\text{MRR}(R, G) = \frac{1}{d} \sum_{i=1}^{d} \frac{1}{\text{rank}_R(\text{MMD}(R_i, G_i))} \tag{2}$$

where $\text{rank}_R(\text{MMD}(R_i, G_i))$ is the rank of $\text{MMD}(R_i, G_i)$ among elements of the sorted list $[\text{MMD}(R_1, G_i), \text{MMD}(R_2, G_i), \ldots, \text{MMD}(R_d, G_i)]$. $\text{MRR}(R, G)$ is maximal and of value 1 when the generated distributions of proteins for a given label are the closest to the distribution of real proteins with the same label.

Since the GO protein function labels we are using are organized in a hierarchy we also include a variant of MRR that gives more insight on conditional performance for closely-related functions in the label hierarchy, by not penalizing ranking errors arising from parent and children labels (MRR_B).

2.1.3 Evaluating the *diversity* of generated sequences

A common failure mode of generative models and specifically in GANs is mode collapse, where a model produces a single mode of the data (Salimans *et al.*, 2016). In practice, we would like to ensure diversity of generated samples in order to represent a significant part of the space of possible sequences while ensuring that the sequences remain realistic. In order to capture this trade-off we consider three measures. First, we monitor the duality gap (Grnarova *et al.*, 2019) of our GAN model (Figure S7). A small duality gap indicates good convergence and common failure modes, such as mode collapse, can be detected. Second, we propose two heuristic *diversity estimates* of the distributions of generated and real sequences. These are the average entropy over feature dimensions ($n = 1.000$ bins) as well as the average pairwise RKHS distance between sequences. Ideally, we would expect these two statistics in the generated data to exhibit small differences (noted ΔEntropy or ΔDist.) relative to the real distribution (i.e. as measured in the test set). Finally, to ensure that we are not overfitting to the training data, we also report nearest-neighbour squared euclidean distances for our model between the feature maps of the generated sequences and training sequences, and control that they are not closer than to the testset (Figure S8).

2.1.4 A note on out-of-distribution evaluation

A particularly interesting aspect of generative protein modeling is the creation of novel sequences. While this is already useful for in-distribution samples which expand the repertoire of existing proteins with new variants, an exciting outlook is the generation of out-of-distribution (OOD) data which would correspond to a novel kind of protein. The evaluation of OOD generation is however notoriously difficult (Ren *et al.*, 2019; Nalisnick *et al.*, 2019). We go first steps into this direction by holding out five manually selected label combinations from the training data and generating sequences conditioned on these label combinations after training. We then report Top-X accuracy ($X \in 1, 10$) where a generated sequence is counted accurate if a true sequence from the held out sample is among its X nearest neighbors in embedding space. The OOD sets contain approximately 1000 sequences each and the number of real sequences that are not in the OOD sets is a multiple of the number of sequences in the OOD set, with a multiplication factor varying from 2 to 30. The held out label names and GO identifiers can be found in supplemental Table S2.

While this metric should give a sense of OOD generation capability in a comparison of different models, we note that biological plausibility of such truly novel OOD samples remains to be shown.

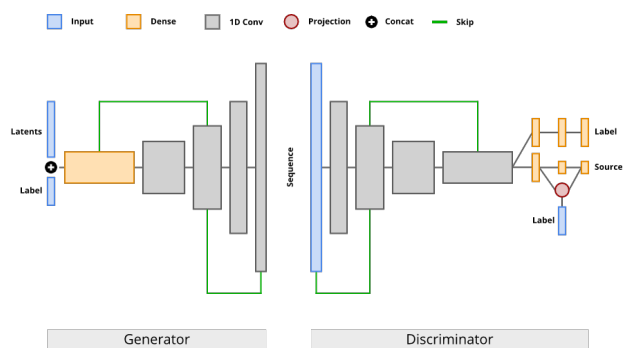


Fig. 1: A schematic overview of the general model structure that we screened during hyperparameter search. Architectural features such as layer number and skip connections can vary. We note that the architecture of our final model differs from the one depicted here.

2.2 A conditional generative model for hierarchical multi-label protein design

After having set the framework to evaluate and compare models for generative protein design, we now develop a conditional generative model to generate proteins with desired functions. While most existing *de novo* protein sequence generation models focus on a specific function, we here aim at modeling different biological functions at the same time. Hence, we introduce ProteoGAN, a conditional GAN (cGAN) able to generate protein sequences given a large set of functions in the Gene Ontology (GO). The GO is a curated set of labels describing protein function and is organized in a directed acyclic graph. We are therefore dealing with a hierarchical multilabel problem.

We explore several conditioning mechanisms, label embeddings and model architectures to find well-suited configurations specifically for hierarchical multilabel protein design. The final model is found by an extensive hyperparameter search guided by our metrics MMD and MRR. We then analyse the results of the optimization by functional Analysis of Variance (fANOVA (Hutter *et al.*, 2014)) to give insights about model parameters. The following sections detail conditioning mechanisms and variants thereof which we propose, the general model architecture and the hyperparameter optimization scheme.

2.2.1 Model architecture

We focus on GANs due to their promising results on protein sequence design tasks (Repecka *et al.*, 2019). Our base model is a Wasserstein-GAN with Gradient Penalty (Arjovsky *et al.*, 2017; Gulrajani *et al.*, 2017). It contains convolutional layers and skip connections in both generator and discriminator, its funnel-like structure is similar to DCGAN (Radford *et al.*, 2015). In the generator, the label is concatenated to the latent noise vector input of the network. In the discriminator we explore various mechanisms for conditioning. Exact model configurations are determined through a hyperparameter search detailed in section F.

2.2.2 Conditioning

We assess the performance of three types of conditioning mechanisms during our hyperparameter search: projection(s), auxiliary classifiers, or a combination of both. To the extent of our knowledge, there is no generative model that uses either multiple projections or a combination of projections and auxiliary classifiers in the literature.

In the conditional GAN with *projection* discriminator, as introduced in Miyato and Koyama (2018), the discriminator D is decomposed into a sum of two terms, one being the inner product between a label embedding and an intermediate transformation of the sequence input, and the second term being solely dependent on the sequence input. Let $(\boldsymbol{x}, \boldsymbol{y}) \sim p$ be a sample from the dataset, where $\boldsymbol{x}$ is a one-hot encoded sequence, and $\boldsymbol{y}$ an encoding of choice of the categorical label. The projection discriminator can be written as $D(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{A}(\boldsymbol{v}(\boldsymbol{y})^T \boldsymbol{\phi}_\theta(\boldsymbol{x}) + \psi_\gamma(\boldsymbol{\phi}_\theta(\boldsymbol{x})))$, with $\boldsymbol{y} \rightarrow \boldsymbol{v}(\boldsymbol{y})$ a linear projection of the label encoding, $\boldsymbol{\phi}_\theta$ an embedding function applied to the sequence $\boldsymbol{x}$, ψ_γ a scalar function applied to the embedding function $\boldsymbol{\phi}_\theta$ and $\mathcal{A}$ an activation function of choice.

We propose an extension to the projection mechanism, namely to use *multiple projections* in the discriminator. We hypothesize that this could help utilizing protein sequence information at the different abstraction levels of the convolutional layers. In addition to the previous notations introduced in this section, let us assume that we have now k projections. Let $\{g_i\}_{i=1}^k$ be k neural networks, which can be decomposed in n_i layers $g_i = l_{n_i}^i \circ \boldsymbol{l}_{n_i-1}^i \circ \cdots \boldsymbol{l}_2^i \circ \boldsymbol{l}_1^i$. Let $\{p_i\}_{i=1}^k$ be the layer number at which the inner product with the output of the linear projection $\{\boldsymbol{v}_i\}_{i=1}^k$ occurs in each neural network. The projections obey a tree-like branching structure, where all layers $p \leq p_i$ of the neural network i are shared with the neural networks j for which $p_i < p_j$, and the branching of a different projection is always done at a different layer number. The discriminator with multiple projections can then be written as $D(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{A}(\sum_{i=1}^k (\boldsymbol{v}_i(\boldsymbol{y})^T \boldsymbol{l}_{p_i}^i \circ \cdots \boldsymbol{l}_1^i(\boldsymbol{x}) + g_i(\boldsymbol{x})))$.

We further propose to include an *auxiliary classifier* (AC) (Odena *et al.*, 2017) C_D to the objective function in addition to the projections, combining two previously independently used conditioning mechanisms. The AC shares parameters with the discriminator except a label classification output layer and adds a classification loss term to both the generator and discriminator: $\gamma\, \mathbb{E}_{q(\boldsymbol{x}|\boldsymbol{y})}[ce(C_D(\boldsymbol{x}), \boldsymbol{y})] + \gamma\, \mathbb{E}_{p(\boldsymbol{x}|\boldsymbol{y})}[ce(C_D(\boldsymbol{x}), \boldsymbol{y})]$, where ce denotes the cross-entropy function, $\boldsymbol{x} \sim q(\boldsymbol{x}|\boldsymbol{y})$ the learned conditional distribution and $\boldsymbol{x} \sim p(\boldsymbol{x}|\boldsymbol{y})$ the conditional distribution of the data. C_D is trained together with the discriminator loss and predicts the labels of the real or generated sequences. The conditioning mechanisms are further explained in the supplementary Section B.

2.2.3 Hierarchical label encoding and chemophysical amino acid properties

Given the hierarchical structure of the functional labels we allow for three types of label encodings $\boldsymbol{y}$: (i) one-hot encoding, as a common encoding for labels, (ii) Poincaré encoding (O'neill, 2014), which embeds labels in a hyperbolic space that is well-suited for hierarchical data and (iii) node2vec encoding (Grover and Leskovec, 2016), which preserves neighbourhood relationships by encoding the nodes of the GO directed acyclic graph (DAG) based on random walks. All of these encodings capture the relations between labels in the GO DAG and that way incorporate the information into the GAN. If a protein has multiple GO labels, the label encodings are summed to represent the set of assigned GO labels. We further allow to concatenate chemophysical properties of the respective amino-acids to the encoding of the sequences. These are obtained from the AAIndex (Kawashima *et al.*, 2007) (`https://www.genome.jp/aaindex`), a list with accession numbers can be found in Table S1.

Table 1. Classification results of SVMs trained on different embeddings, for the structural classes of the CATH database (balanced accuracy), and for 50 functional classes of the GO (F_{max} score). All values in percent.

Embedding	C	A	T	H	GO
ProFET	57 ± 4	42 ± 7	49 ± 2	31 ± 2	51 ± 1
Spectrum	67 ± 4	54 ± 4	57 ± 2	49 ± 2	56 ± 1
UniRep	74 ± 3	75 ± 5	67 ± 2	58 ± 2	72 ± 1

3 Implementation

3.1 Data

Sequence data and GO labels were obtained from the UniProt Knowledgebase (UniProtKB, Consortium (2019)) and filtered for experimental evidence, at least one existing GO annotation, standard amino acids and a maximum length of 2.048. It resulted in 150.000 sequences total. We restricted functional labels to a total number of 50, imposing a minimum threshold of ca. 5.000 sequences per label. In the GO DAG, sequences automatically inherit the labels of their parents, therefore such missing labels were imputed to complete the hierarchical information. We note that sequences exhibiting one of five manually selected label combinations (named A-E) were held out from the training data, to test the OOD performance of our model.

We randomly split the dataset in training, validation and test sets keeping ca. 15.000 (10%) sequences in both the validation and test sets. During hyperparameter optimization, we use smaller splits with ca. 3.000 sequences each. We ensure that all labels have a minimum amount of samples in the test and validation sets, and use the same number of sequences per class for the calculation of our MRR measure (1.300 and 300 sequences, respectively). Further details about the dataset and splits are available in the supplementary Section C and Figure S1.

Since we do two-sample tests we do not remove homologous sequences from the test set, but for completeness we report our results with homology control up to 50% in supplemental section M (compare also the approach of Bileschi *et al.* (2019)).

3.2 Baseline comparisons

We compare our model ProteoGAN with classic probabilistic language models and a more recent deep-learning model for protein sequence generation based on the Variational Autoencoder (VAE) framework:

- *HMM*: A profile HMM (Eddy, 2004) for each individual label (marked OpL for 'one-per-label') and for each label combination (marked OpC for 'one-per-combination'). The former discards multi-label information and totals 50 models, the latter accounts for ca. 1.800 models, one model for each label combination present in the training set.
- *n-gram*: An n-gram model ($n = 3$) for each individual label (marked OpL, discards multi-label information, total of 50 models) and for each label combination (marked OpC, total of ca. 1.800 models).
- *CVAE*: A conditional Variational Autoencoder for proteins from Greener *et al.* (2018). We adjusted the model to incorporate the 50 labels of our problem setting and performed a Bayesian optimization hyperparameter search.

We also perform ablation studies on ProteoGAN to understand the influence of several aspects of the model:

- *one-per-label GAN (OpL-GAN)*: One instance of ProteoGAN for every label, with the conditioning mechanism removed (total of 50 models). Sequences for a target label are generated by sampling them from the GAN trained on the sequences annotated with the same label. With this model, we assess whether training 50 models can be replaced by a conditioning mechanism.

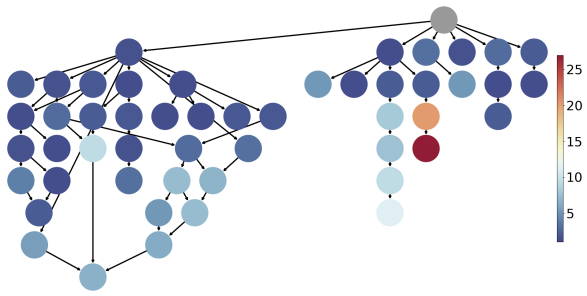


Fig. 2: Mean rank of each individual label in Spectrum MRR over five data splits. The structure represents the relations of the 50 labels of interest in the GO DAG. Nodes are colored by how well the model can target them. Dark blue indicates that the model can target the function well, the worst targeted label (red) is "kinase activity".

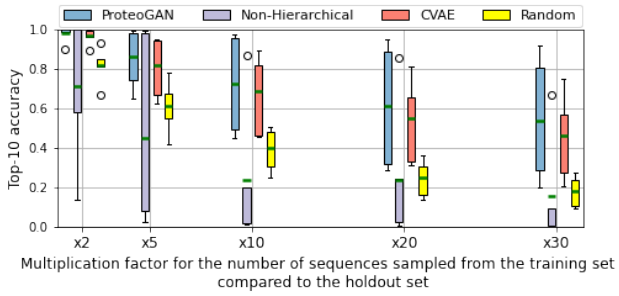


Fig. 3: Top-10 accuracy (in %) with the Spectrum embedding for OOD-capable models. The boxplots cover the 5 OOD sets, A to E, the green line represents the average. We add a random baseline for comparison, where generated sequences are sampled uniformly at random from the training set. Complementary results for the other embeddings can be found in Figure S13

- *Predictor-Guided*: ProteoGAN without conditioning mechanism, which results in a single GAN trained on the full data. The generated sequences are then annotated with the labels predicted by a state-of-the-art predictor NetGO (You *et al.*, 2019). Comparing to this model allows us to investigate how a predictor model guiding the GAN compares to a conditional GAN.
- *Non-Hierarchical*: Same as ProteoGAN, but trained without the hierarchical multi-label information. Each sequence is included multiple times with each of its original labels separately. For fairness, we keep the number of gradient updates the same as for the other models. With this model we explore the usefulness of accounting for the GO hierarchy.

We refer the reader to the respective papers and to the supplementary Section D for further information on the baseline models.

3.3 Hyperparameter optimization

We conducted two Bayesian Optimization and HyperBand (BOHB) searches (Falkner *et al.*, 2018) on six Nvidia GeForce GTX 1080, first a broad search among 23 hyperparameters (1.000 models) and a second, smaller and more selective, among 9 selected hyperparameters (1.000 models) on a maximum of 27 epochs. The optimization objective was set to maximize the ratio of our evaluation measures MRR/MMD to balance between distribution similarity and conditional consistency of the generated sequences. Both searches were complemented by a functional analysis of variance (fANOVA) (Hutter *et al.*, 2014). The 27 best selected models of the second hyperparameter search were then trained for a prolonged duration of 100 epochs, the best performing model of these (i.e. ProteoGAN) then for 300 epochs. Further details about hyperparameter optimization are available in the supplementary Section E.

4 Discussion

4.1 Meta-evaluation of metrics: Spectrum MMD is an efficient metric for Protein Design

Different embeddings capture different aspects of the original data. We were interested whether the relatively simple Spectrum kernel embedding would be sufficient to assess distribution similarity and conditional consistency, and hence compared it to two biologically founded embeddings: ProFET (Ofer and Linial, 2015), a handcrafted selection of sequence features relating mostly to biophysical properties of single amino acids or sequence motifs, and UniRep (Alley *et al.*, 2019), an LSTM-based learned embedding that was shown to recover various aspects of proteins, including structural and functional properties.

We compared these embeddings by scoring their ability to classify protein structure and function, for which we trained Support Vector Machines (SVM) trained on each of the embeddings. We classify domains of the CATH protein structure classification database (10.000 sampled out of 500.000, 10 repetitions, $80-20$ train-test split), where we report balanced accuracy (Table 1), and we classify the 50 GO functional terms of our dataset (10.000 sampled out of 150.000, 10 repetitions, $80-20$ train-test split), where we report the $F_{\max}$ score used in the CAFA challenge, compare Zhou *et al.* (2019) and supplemental Section A) (Table 1).

The UniRep embedding is arguably the most powerful in this comparison and expectedly achieved the best scores. Notably though, the Spectrum kernel embedding is also considerably well suited to assess aspects of proteins on the structure and function level as compared to ProFET and UniRep, while being orders of magnitudes faster to compute and requiring less compute resources. This makes it more fit for the requirements on performance during evaluation or hyperparameter optimization of neural networks and other models. We therefore propose it as the primary feature map in our evaluation metrics, and confirm it with evaluations based on the other embeddings (Tables S6,S7). To validate MMD itself as a well-suited test statistic for protein design we confirm it with feature-wise KS statistics (Figures S10,S11,S12).

4.2 Hyperparameter analysis: The conditional discriminator of ProteoGAN is most critical to its performance

We tested a wide range of hyperparameters and architectural choices for cGANs and analysed them in a fANOVA framework with respect to the protein design performance metrics MMD and MRR. To inform subsequent work on these models, we could empirically derive several design principles for GANs specifically for protein design (please refer to supplemental Section F for the raw marginal predictions of all parameters from which we deduce the following statements).

To begin with, smaller architectures performed much better than networks with more than four hidden layers. This size seems to be sufficient to model proteins, although of course the optimization places a selective pressure towards fast converging (small) models. The generator was less sensitive to its learning rate, while the discriminator showed strong preference towards learning rates below 1e-3. This may arise from the increased burden on the discriminator from the secondary training objective for classifying labels. It follows that it is more important that the discriminator arrives at an optimal solution rather than at local optima often found by larger learning rates.

Table 2. Evaluation of ProteoGAN and various baselines with MMD, MRR and diversity metrics based on the Spectrum kernel embedding (the results of other embeddings can be found in Table S6 and S7). An arrow indicates that lower (↓) or higher (↑) is better. Best results in bold, second best underlined. Given are mean values and standard deviation over five data splits. Due to the computational effort, the OpL-GAN model was only trained on one split. The second and third panel show models presented in Section 3.2. The fourth panel shows variants of ProteoGAN that were trained on different datasets with more labels (100 or 200).

	MMD↓	gauss. MMD↓	MRR↑	MRR_B ↑	ΔEntropy	ΔDistance
Positive Control	0.011 ± 0.000	0.009 ± 0.000	0.893 ± 0.016	0.966 ± 0.018	0.002 ± 0.006	-0.000 ± 0.001
Negative Control	1.016 ± 0.000	0.803 ± 0.000	0.090 ± 0.000	0.099 ± 0.001	0.728 ± 0.006	1.843 ± 0.001
ProteoGAN	0.043 ± 0.001	0.028 ± 0.001	**0.554 ± 0.031**	**0.709 ± 0.034**	**-0.010 ± 0.010**	0.012 ± 0.004
Predictorguided	**0.026 ± 0.001**	**0.018 ± 0.001**	0.114 ± 0.007	0.136 ± 0.016	0.014 ± 0.009	**0.001 ± 0.003**
Non-Hierarchical	0.337 ± 0.118	0.238 ± 0.088	0.306 ± 0.034	0.406 ± 0.039	-0.352 ± 0.178	0.290 ± 0.171
CVAE	0.232 ± 0.078	0.155 ± 0.056	0.301 ± 0.053	0.424 ± 0.083	0.247 ± 0.027	0.145 ± 0.085
OpC-ngram	0.056 ± 0.001	0.037 ± 0.001	0.402 ± 0.018	0.505 ± 0.034	0.208 ± 0.006	-0.050 ± 0.002
OpC-HMM	0.170 ± 0.003	0.114 ± 0.002	0.095 ± 0.001	0.143 ± 0.002	-0.579 ± 0.014	0.199 ± 0.004
OpL-GAN	**0.036**	**0.024**	**0.597**	**0.747**	**-0.062**	**0.022**
OpL-ngram	0.060 ± 0.001	0.039 ± 0.001	0.329 ± 0.009	0.396 ± 0.009	0.232 ± 0.007	-0.053 ± 0.002
OpL-HMM	0.195 ± 0.002	0.132 ± 0.002	0.100 ± 0.003	0.147 ± 0.002	-0.654 ± 0.015	0.244 ± 0.004
ProteoGAN (100 labels)	0.036	0.025	0.585	0.736	-0.026	0.019
ProteoGAN (200 labels)	0.162	0.113	0.374	0.524	0.104	0.051

Table 3. Comparison (MMD) of ProteoGAN and random mutagenesis with same or reduced mutation rate. Homolog MRR is computed by replacing labels of generated sequences with labels of their closest homolog.

	Spectrum	ProFET	UniRep
ProteoGAN	0.04	0.17	1.50
Random Mutagenesis	0.18	1.27	2.31
Random Mutagenesis (-50%)	0.11	0.66	1.92
Random Mutagenesis (-70%)	0.08	0.40	1.46
Homolog MRR	0.38	0.43	0.19

We observed a trade-off between distribution similarity and conditional consistency. This manifested in increasing MRR and decreasing MMD performance when weighing stronger the training loss term of the auxiliary classifier, and also when switching between the different conditioning mechanisms. While we could not show significant impact of our proposed multiple projections, the combination of both conditioning mechanisms showed clear improvements in conditional performance.

With respect to different label embeddings capturing the hierarchical relationships in the model, as well as to appending biophysical feature vectors to the sequence embedding, we surprisingly found that a simple one-hot encoding and the raw sequence without additional features showed the best results. The amino acid identity, rather than its properties, appears to be more critical to sequence modeling.

Moreover the discrete one-hot label embedding seems to be easier to interpret for the model than the continuous node2vec embeddings or the hyperbolic Poincaré embeddings. While these embeddings contain more information, the one-hot encoding presents them in a more accessible form. Also, hyperbolic spaces require special operators for many basic concepts that a neural network would need to learn first (Ganea *et al.*, 2018).

Other popular extensions to the GAN framework such as input noise, label smoothing or training ratios did not significantly affect model performance in our context (compare Figures S5, S6). Summarizing, a small model with both conditioning mechanisms and no further sequence or label augmentation worked best. Further improvements to the architecture should focus on improving the discriminator, as hyperparameters affecting it showed the most impact (Figure S5). Our final model ProteoGAN is the best performing model of the optimization and has multiple projections, an auxiliary classifier, no biophysical features and one-hot encoding of label information.

4.3 Baseline comparisons: ProteoGAN outperforms other methods

Based on the proposed metrics for distribution similarity, conditional consistency and diversity we assess the performance of ProteoGAN and compare it to several baselines. The results are consolidated by an evaluation with the biological embeddings ProFET and UniRep, as well as feature-wise KS-statistics of the embeddings (Tables 2,S6,S7, Figures S10,S11,S12).

ProteoGAN clearly outperforms the HMM, n-gram model and CVAE on all metrics and embeddings. The same applies to the OpL versions which are trained once per label. The different embeddings (Tables 2,S6,S7) largely agree with each other, although it should be noted that the n-gram model almost directly optimizes the MMD-based metrics and is hence overestimated there, ProFET and UniRep provide a better picture in this case.

4.4 Ablation models: Hierarchical information dramatically improves conditioning

We also investigated several ablation models to demonstrate the advantages of conditioning on hierarchical labels. To begin with, the predictor-guided model had a very low conditional performance (MRR=0.114) and hence the original model exceeded it by a large margin (MRR=0.554). This shows that general function predictors for proteins are not (yet) suited to guide generative models at evaluation time. Continuous guidance during training might improve this result, but is time-wise prohibitive. The better MMD scores of the predictor model is likely due to the missing burden of the conditioning mechanism. We also observed this trade-off between MMD and MRR in the hyperparameter optimization.

Similarly, the non-hierarchical model (MMD=0.337, MRR=0.306) is clearly outperformed by the original ProteoGAN. The hierarchical information drastically helps model performance, presumably because the label structure can be transferred to the underlying sequence data structure, and because such a model does not need to learn each label marginal distribution independently.

The OpL-GAN separates the conditional learning problem into several sub-problems, and was in fact slightly better than ProteoGAN. Yet, ProteoGAN could achieve the result of 50 independent models by training

a single conditional model with a minor trade-off in performance. Besides the lower training effort of ProteoGAN, the conditioning mechanism has the advantage to allow for functional OOD generation, as discussed below.

Figure 2 breaks down the conditional performance of ProteoGAN with respect to the individual labels. 27 of the 50 labels were on average ranked first or second and hence could very well be targeted. 33 of the 50 labels were ranked at least third. Certain branches of the ontology were more difficult to model (details available in Figure S9).

We asked how well our model could scale beyond 50 labels and trained it without any further tuning on 100 and 200 labels. While performance even gets better with 100 labels, once more showing that label information is advantageous, it starts to drop at 200 labels. Any scaling beyond this point will require hyperparameter tuning and likely an increase in parameter capacity to model the additional labels. Also, the amount of available training samples per class drops rapidly with increasing specificity of the labels. However the functional diversity we consider here would already enable many applications in *de novo* protein design.

4.5 Applicability: ProteoGAN can support protein screenings with a larger sequence space

It is difficult to proof biological validity without wet lab validation, and we do not claim to do so here. We acknowledge that MMD values still show significant difference to the positive control, and that corresponding p-values (Table S8) were inconclusive in this regard. Hence it is likely that generated sequences are not immediately usable out of the box, but need some experimental tuning as in directed evolution. Here we see the main application of ProteoGAN at this time: The extension of protein screenings with candidates that are further away from the known sequence space than previously possible, yet more likely to be functional than comparably novel candidates of other methods. To this end we compare ProteoGAN to random mutagenesis, the traditional method to produce candidates for such screenings, by determining the number of edit operations necessary to obtain a generated sequence from its closest naturally occurring homolog. We then replace them by random mutations and determine MMD, gradually reducing the mutation rate until it matches the result of ProteoGAN (Table 3).

We first observed that generated sequences had an average percent identity of $56\% \pm 2\%$, indicating that we are not simply reproducing training examples and sequences are novel. We refer in passing to Repecka *et al.* (2019) who had great success in validating GAN-generated proteins in vitro with a similar identity. Random mutagenesis achieved ProteoGAN's results after reducing the mutation rate by 70%. We hence conclude that our model is about three times more efficient than random mutagenesis. Additionally we investigated the biological plausibility of the conditional generation by replacing the labels of generated sequences with the labels of their closest homologs. The MRR results (Table 3) show that good conditional performance is maintained also when judging from sequence homology to real proteins.

4.6 Outlook: Conditioning may enable the design of novel protein functions

As an interesting outlook we provide first evaluations with respect to OOD generation. We hypothesize that the conditioning mechanism may be used to combine previously unrelated functional labels into one protein, which would enable the design of completely novel kinds of proteins with previously unseen functionality. While also here, biological implementation is inevitable to proof this concept, we can report that ProteoGAN and CVAE both showed promising TopX accuracies on five held out label combinations (Figures 3, S13). Further development of this concept will provide new tools for biotechnology.

4.7 Conclusion

We provide and evaluate broadly applicable metrics for assessing distribution similarity, conditional consistency and diversity in generative protein design. Due to their computational efficiency they can be used to compare and develop generative models at scale. With these metrics we hope to simplify the process of developing generative models in the domain of protein sequences. We further present ProteoGAN, a GAN conditioned on hierarchical labels from the Gene Ontology, which outperforms classic and state-of-the-art models for (conditional) protein sequence generation. We envision that ProteoGAN may be used to exploit promising regions of the protein sequence space that are inaccessible by experimental random mutagenesis. It is universally applicable in various contexts that require different protein functions, and is even able to provide sequence candidates for never seen proteins. Extensions to this framework could incorporate other conditional information, such as structure motifs, binding partners, or other types of ontologies. Further development of such models may make proteins available as universal molecular machines that can be purely computationally designed *ad hoc* for any given biotechnological application.

Acknowledgements

We thank Karsten Borgwardt for insightful discussions.

References

Alford, R. F. *et al.* (2017). The rosetta all-atom energy function for macromolecular modeling and design. *Journal of chemical theory and computation*, **13**(6), 3031–3048.

Alley, E. C. *et al.* (2019). Unified rational protein engineering with sequence-based deep representation learning. *Nature methods*, **16**(12), 1315–1322.

Angermueller, C. *et al.* (2019). Model-based reinforcement learning for biological sequence design. In *International Conference on Learning Representations*.

Arjovsky, M. *et al.* (2017). Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 214–223.

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

Bileschi, M. L. *et al.* (2019). Using deep learning to annotate the protein universe. *bioRxiv*, page 626507.

Borgwardt, K. M. *et al.* (2006). Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, **22**(14), e49–e57.

Brookes, D. *et al.* (2019). Conditioning by adaptive sampling for robust design. volume 97 of *Proceedings of Machine Learning Research*, pages 773–782, Long Beach, California, USA. PMLR.

Brown, T. B. *et al.* (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Chhibbar, P. and Joshi, A. (2019). Generating protein sequences from antibiotic resistance genes data using generative adversarial networks. *arXiv preprint arXiv:1904.13240*.

Consortium, U. (2019). Uniprot: a worldwide hub of protein knowledge. *Nucleic acids research*, **47**(D1), D506–D515.

Das, P. *et al.* (2018). Pepcvae: Semi-supervised targeted design of antimicrobial peptide sequences. *arXiv preprint arXiv:1810.07743*.

Davidsen, K. *et al.* (2019). Deep generative models for t cell receptor protein sequences. *Elife*, **8**, e46935.

Deng, L. *et al.* (2019). Madoka: an ultra-fast approach for large-scale protein structure similarity searching. *BMC bioinformatics*, **20**(19), 1–10.

DeVries, T. *et al.* (2019). On the evaluation of conditional gans. *arXiv preprint arXiv:1907.08175*.

Dill, K. A. and MacCallum, J. L. (2012). The protein-folding problem, 50 years on. *Science*, **338**(6110), 1042–1046.

Durbin, R. *et al.* (1998). Biological sequence analysis. probabilistic models of proteins and nucleic acids.

Eddy, S. R. (2004). What is a hidden markov model? *Nature biotechnology*, **22**(10), 1315–1316.

Falkner, S. *et al.* (2018). Bohb: Robust and efficient hyperparameter optimization at scale. In *ICML*.

Gane, A. *et al.* (2019). A comparison of generative models for sequence design. *Machine Learning in Computational Biology Workshop*.

Ganea, O.-E. *et al.* (2018). Hyperbolic neural networks. *arXiv preprint arXiv:1805.09112*.

Goodfellow, I. *et al.* (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Greener, J. G. *et al.* (2018). Design of metalloproteins and novel protein folds using variational autoencoders. *Scientific reports*, **8**(1), 1–12.

Gretton, A. *et al.* (2012). A kernel two-sample test. *The Journal of Machine Learning Research*, **13**(1), 723–773.

Grnarova, P. *et al.* (2019). A domain agnostic measure for monitoring and evaluating gans. In *Advances in Neural Information Processing Systems*, pages 12092–12102.

Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864.

Gulrajani, I. *et al.* (2017). Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777.

Gupta, A. and Zou, J. (2019). Feedback gan for dna optimizes protein functions. *Nature Machine Intelligence*, **1**(2), 105–111.

Heusel, M. *et al.* (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637.

Huang, P.-S. *et al.* (2016). The coming of age of de novo protein design. *Nature*, **537**(7620), 320–327.

Hutter, F. *et al.* (2014). An efficient approach for assessing hyperparameter importance. In *International conference on machine learning*, pages 754–762. PMLR.

Ingraham, J. *et al.* (2019). Generative models for graph-based protein design. In *Advances in Neural Information Processing Systems*, pages 15820–15831.

Jamieson, K. and Talwalkar, A. (2016). Non-stochastic best arm identification and hyperparameter optimization. In *Artificial Intelligence and Statistics*, pages 240–248.

Karimi, M. *et al.* (2019). De novo protein design for novel folds using guided conditional wasserstein generative adversarial networks (gcwgan). *bioRxiv*, page 769919.

Kawashima, S. *et al.* (2007). AAindex: Amino acid index database, progress report 2008. *Nucleic acids research*, **36**(suppl_1), D202–D205.

Killoran, N. *et al.* (2017). Generating and designing dna with deep generative models. *arXiv preprint arXiv:1712.06148*.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In Y. Bengio and Y. LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada*.

Kynkäänniemi, T. *et al.* (2019). Improved precision and recall metric for assessing generative models. In *Advances in Neural Information Processing Systems*, pages 3927–3936.

Leslie, C. *et al.* (2001). The spectrum kernel: A string kernel for svm protein classification. In *Biocomputing 2002*, pages 564–575. World Scientific.

Li, C.-L. *et al.* (2017a). Mmd gan: Towards deeper understanding of moment matching network. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 2200–2210.

Li, L. *et al.* (2017b). Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, **18**(1), 6765–6816.

Madani, A. *et al.* (2020). Progen: Language modeling for protein generation. *arXiv preprint arXiv:2004.03497*.

Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Miyato, T. and Koyama, M. (2018). cgans with projection discriminator. *International Conference on Learning Representations*.

Mueller, A. T. *et al.* (2018). Recurrent neural network model for constructive peptide design. *Journal of chemical information and modeling*, **58**(2), 472–479.

Nalisnick, E. *et al.* (2019). Detecting out-of-distribution inputs to deep generative models using a test for typicality. *arXiv preprint arXiv:1906.02994*, **5**, 5.

Odena, A. *et al.* (2017). Conditional image synthesis with auxiliary classifier GANs. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2642–2651. JMLR. org.

Ofer, D. and Linial, M. (2015). Profet: Feature engineering captures high-level protein functions. *Bioinformatics*, **31**(21), 3429–3436.

O'neill, B. (2014). *Elementary differential geometry*. Academic press.

Papineni, K. *et al.* (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Radford, A. *et al.* (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, **abs/1511.06434**.

Radivojac, P. *et al.* (2013). A large-scale evaluation of computational protein function prediction. *Nature methods*, **10**(3), 221–227.

Ren, J. *et al.* (2019). Likelihood ratios for out-of-distribution detection. *arXiv preprint arXiv:1906.02845*.

Repecka, D. *et al.* (2019). Expanding functional protein sequence space using generative adversarial networks. *bioRxiv*, page 789719.

Rezende, D. J. *et al.* (2014). Stochastic backpropagation and approximate inference in deep generative models. *International Conference in Machine Learning*.

Riesselman, A. *et al.* (2019). Accelerating protein design using autoregressive generative models. *BioRxiv*, page 757252.

Salimans, T. *et al.* (2016). Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242.

Shmelkov, K. *et al.* (2018). How good is my gan? In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 213–229.

Vaswani, A. *et al.* (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Vegas, E. *et al.* (2016). Inferring differentially expressed pathways using kernel maximum mean discrepancy-based test. *BMC bioinformatics*, **17**(5), 399–405.

Yang, J. *et al.* (2020). Improved protein structure prediction using predicted interresidue orientations. *Proceedings of the National Academy of Sciences*, **117**(3), 1496–1503.

You, R. *et al.* (2019). Netgo: improving large-scale protein function prediction with massive network information. *Nucleic acids research*, **47**(W1), W379–W387.

Zhang, Y. and Skolnick, J. (2004). Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function, and Bioinformatics*, **57**(4), 702–710.

Zhou, N. *et al.* (2019). The cafa challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens. *Genome biology*, **20**(1), 1–23.

Supplementary material

A Definition of $F_{\max}$-score

The CAFA challenge (Zhou *et al.*, 2019) uses the $F_{\max}$-score to compare the performance of protein function predictors:

$$
\begin{aligned}
pr(\tau) &= \frac{1}{m(\tau)} \sum_{i=1}^{m(\tau)} \frac{\sum_f \mathbb{1}(f \in P_i(\tau) \wedge f \in T_i)}{\sum_f \mathbb{1}(f \in P_i(\tau))}, \\
rc(\tau) &= \frac{1}{n} \sum_{i=1}^{n} \frac{\sum_f \mathbb{1}(f \in P_i(\tau) \wedge f \in T_i)}{\sum_f \mathbb{1}(f \in T_i)}, \\
F_{max} &= \max_{\tau} \left\{ \frac{2 \cdot pr(\tau) \cdot rc(\tau)}{pr(\tau) + rc(\tau)} \right\},
\end{aligned}
\tag{3}
$$

where f is a GO term, n is the number of sequences, $m(\tau)$ is the number of sequences with a score greater than or equal to τ, $P_i(\tau)$ is the set of predicted terms with a score greater than or equal to τ for a protein sequence i, T_i denotes the corresponding ground truth set of terms for that sequence, and $\mathbb{1}(\cdot)$ an indicator function.

B Conditioning mechanism

In this section, we first detail how we adapted the Wasserstein loss to the conditional setting, then we describe state-of-the-art conditional GAN objective functions and variants used in this project.

Loss function of conditional GANs: Our models are trained with the Wasserstein objective with gradient penalty from (Gulrajani *et al.*, 2017). As a reminder, the WGAN-GP losses can be written as follows:

$$\begin{aligned}\mathcal{L}_D &= \mathbb{E}_{q(\boldsymbol{x})}[D(\boldsymbol{x})] - \mathbb{E}_{p(\boldsymbol{x})}[D(\boldsymbol{x})] \\ &\quad + \lambda\, \mathbb{E}_{m(\hat{\boldsymbol{x}})}[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2] \\ \mathcal{L}_G &= -\,\mathbb{E}_{q(\boldsymbol{x})}[D(\boldsymbol{x})]\end{aligned} \tag{4}$$

where $\boldsymbol{x} \sim p(\boldsymbol{x})$ is the data distribution and $\boldsymbol{x} \sim q(\boldsymbol{x})$ is the generator model distribution, $\hat{\boldsymbol{x}}$ is an interpolated sample between a real sequence and a generated one, m is the distribution of interpolated samples, D is the discriminator (or critic), $\mathcal{L}_D$ the loss of the discriminator and $\mathcal{L}_G$ the loss of the generator. The term $\mathbb{E}_{m(\hat{\boldsymbol{x}})}[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2]$ ensures that the discriminator is Lipschitz continuous.

To be able to use the Wasserstein objective with gradient penalty in the projection discriminator (Miyato and Koyama, 2018) (see below), we had to adapt the objective formula to include the label information. Let D be the discriminator and G the generator. Let $(\boldsymbol{x}, \boldsymbol{y}) \sim p$ be a sample from the dataset, where $\boldsymbol{x}$ is a one-hot encoded sequence, and $\boldsymbol{y}$ an encoding of choice of the categorical label. Let q be the generator model distribution, such that $\boldsymbol{y} \sim q(\boldsymbol{y})$ is defined by the user and, in practice, follows the distribution of the labels in the dataset, and $\boldsymbol{x} \sim q(\boldsymbol{x}|\boldsymbol{y})$ is learned. Let $\hat{\boldsymbol{x}}$ be a linear interpolation between a real sequence and a generated one, with interpolation parameter chosen uniformly at random between 0 and 1. We call $\hat{\boldsymbol{x}} \sim m(\hat{\boldsymbol{x}}|\boldsymbol{y})$ the conditional distribution of interpolated sequences given a label encoding $\boldsymbol{y}$. Let λ be a weighing factor introduced in (Gulrajani *et al.*, 2017). The discriminator and generator losses can be written as follows:

$$\begin{aligned}\mathcal{L}_D &= \mathbb{E}_{q(\boldsymbol{y})}[\mathbb{E}_{q(\boldsymbol{x}|\boldsymbol{y})}[D(\boldsymbol{x}, \boldsymbol{y})]] - \mathbb{E}_{p(\boldsymbol{y})}[\mathbb{E}_{p(\boldsymbol{x}|\boldsymbol{y})}[D(\boldsymbol{x}, \boldsymbol{y})]] \\ &\quad + \lambda\, \mathbb{E}_{p(\boldsymbol{y})}[\mathbb{E}_{m(\hat{\boldsymbol{x}}|\boldsymbol{y})}[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}}, \boldsymbol{y})\|_2 - 1)^2]], \\ \mathcal{L}_G &= -\,\mathbb{E}_{q(\boldsymbol{y})}[\mathbb{E}_{q(\boldsymbol{x}|\boldsymbol{y})}[D(\boldsymbol{x}, \boldsymbol{y})]].\end{aligned} \tag{5}$$

This formulation ensures that the Lipschitz constraints imposed on the discriminator in the unconditional WGAN-GP objective holds for each class.

Case of the cGAN with projection discriminator (Miyato and Koyama, 2018): In the conditional GAN with projection discriminator model, the discriminator is decomposed into a sum of two terms, one being the inner product between a label embedding and an intermediate transformation of the input, and the second term being solely depending on the input sequence $\boldsymbol{x}$. The new expression of the projection discriminator can be derived by assuming that the label is categorical and that both the log-likelihoods of the data and target distribution can be written as log linear models. Let $\boldsymbol{y} \rightarrow \boldsymbol{v}(\boldsymbol{y})$ be a linear projection of the label encoding. Let $\boldsymbol{\phi}_\theta$ be an embedding function applied to the input $\boldsymbol{x}$ and ψ_γ a scalar function applied to the embedding function $\boldsymbol{\phi}_\theta$. Let $\mathcal{A}$ be an activation function of choice. The projection discriminator in (Miyato and Koyama, 2018) can therefore be written as:

$$D(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{A}(\boldsymbol{v}(\boldsymbol{y})^T \boldsymbol{\phi}_\theta(\boldsymbol{x}) + \psi_\gamma(\boldsymbol{\phi}_\theta(\boldsymbol{x}))) \tag{6}$$

The label information is therefore introduced via an inner-product. This formulation leads to a more stable algorithm compared to a simple concatenation of the label with the input (Mirza and Osindero, 2014), potentially thanks to the introduction of a form of regularization on the discriminator.

In this project, we tested the possibility to include several projections in the discriminator. In addition to the previous notations introduced in this section, let us assume that we have now k projections. Let $\{g_i\}_{i=1}^k$ be k neural networks, which can be decomposed in n_i layers $g_i = l^i_{n_i} \circ \boldsymbol{l}^i_{n_i-1} \circ \cdots \boldsymbol{l}^i_2 \circ \boldsymbol{l}^i_1$. Let $\{p_i\}_{i=1}^k$ be the layer number at which the inner product with the output of the linear projection $\{\boldsymbol{v}_i\}_{i=1}^k$ occurs in each neural network. The projections obey a tree-like branching structure, where all layers $p \leq p_i$ of the neural network i are shared with the neural networks j for which $p_i < p_j$ and the branching of a different projection is always done at a different layer number. The discriminator with multiple projections can then be written as:

$$D(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{A}(\sum_{i=1}^{k} (\boldsymbol{v}_i(\boldsymbol{y})^T \boldsymbol{l}^i_{p_i} \circ \cdots \boldsymbol{l}^i_1(\boldsymbol{x}) + g_i(\boldsymbol{x}))) \tag{7}$$

In practice we allow for up to four projections. Our BOHB hyperparameter searches did not show evidence of the superiority of projection mechanisms for conditioning purposes when they are the unique type of conditional mechanism in the network. However, the projection models were able to generate sequences similar to naturally occurring ones (low MMD), and the resulting model of the optimization had two projections.

Case of the cGAN model with auxiliary classifier (Odena *et al.*, 2017): As opposed to cGANs with projection discriminator, cGANs with auxiliary classifier add a term to the generator and discriminator losses to incorporate the log-likelihood of the correct labels (compare Equation 5). In addition to notations introduced for Equation 5, let C_D be the auxiliary classifier, ce the cross entropy and γ a weighting factor. For each label $\boldsymbol{y}$, the loss function of cGANs with auxiliary classifiers can be written as:

$$\begin{aligned}\mathcal{L}_D &= \mathbb{E}_{q(\boldsymbol{y})}[\mathbb{E}_{q(\boldsymbol{x}|\boldsymbol{y})}[D(\boldsymbol{x}, \boldsymbol{y})]] - \mathbb{E}_{p(\boldsymbol{y})}[\mathbb{E}_{p(\boldsymbol{x}|\boldsymbol{y})}[D(\boldsymbol{x}, \boldsymbol{y})]] \\ &\quad + \lambda\, \mathbb{E}_{p(\boldsymbol{y})}[\mathbb{E}_{m(\hat{\boldsymbol{x}}|\boldsymbol{y})}[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}}, \boldsymbol{y})\|_2 - 1)^2]] \\ &\quad + \gamma\, \mathbb{E}_{p(\boldsymbol{x}|\boldsymbol{y})}[ce(C_D(\boldsymbol{x}), \boldsymbol{y})] + \gamma\, \mathbb{E}_{q(\boldsymbol{x}|\boldsymbol{y})}[ce(C_D(\boldsymbol{x}), \boldsymbol{y})], \\ \mathcal{L}_G &= -\,\mathbb{E}_{q(\boldsymbol{y})}[\mathbb{E}_{q(\boldsymbol{x}|\boldsymbol{y})}[D(\boldsymbol{x}, \boldsymbol{y})]] + \gamma\, \mathbb{E}_{p(\boldsymbol{x}|\boldsymbol{y})}[ce(C_D(\boldsymbol{x}), \boldsymbol{y})] + \gamma\, \mathbb{E}_{q(\boldsymbol{x}|\boldsymbol{y})}[ce(C_D(\boldsymbol{x}), \boldsymbol{y})]\end{aligned} \tag{8}$$

C_D typically shares weights with D and is trained when minimising $\mathcal{L}_D$ but is fixed when minimising $\mathcal{L}_G$.

In our work, we compare both types of conditional GANs (GAN equipped with auxiliary classifier or with multiple projections at several layers (see Equation 7)) to a third proposed model that combines both mechanisms. It is important to note that in this case the label information introduced in the projection may not be shared with the auxiliary classifier. The fANOVA analysis performed on the second BOHB optimization results (Figure S4) shows that the combination of both mechanisms helps to obtain a better performing conditioning mechanism, as measured by MRR.

C Data

Sequence data was obtained from the UniProt Knowledgebase on May 23rd, 2020 via the query `(existence:"evidence at transcript level" OR existence:"evidence at protein level") goa:(evidence:manual) go:0003674`. For this proof-of-concept work we selected highly confident annotations and sequences, future work may profit from e.g. pretraining with a larger sequence pool.

Functional labels were collected from the same database. The gene ontology (GO) resource is composed of three branches, molecular function, cellular component and biological process. We focus on the molecular function ontology which contains thousands of terms ranging from description like *binding* (GO:0005488) to very specific terms such as *microtubule-severing ATPase activity* (GO:0008568). Each protein is annotated with a set of GO labels describing the molecular function of a protein in modular way. The ontology is structured as a directed acyclic graph with a single root. Further, labels have a hierarchical relationship, i.e. protein with a given functional label inherits automatically the labels of its parents in the DAG (*is-a* relationship). The molecular function ontology resource currently contains more than ten thousand labels, many of which have a highly specific meaning and only few representatives. We therefore restrict the number of labels to the 50 largest classes. We argue fifty labels is sufficient for a proof-of-principle and would even enable the design of experimental assays for validation. Figure S1 illustrates the selected subset of labels and their relationships. The ontology itself (the DAG) was downloaded at the same time from a separate source: `http://purl.obolibrary.org/obo/go/go-basic.obo`

Train, validation and test splits were created to preferably represent all labels uniformly in the test and evaluation sets. This is complicated by the multi-label structure of the data and we hence resort to a heuristic approach. To create the sets, we randomly sample sequences until there is at least 1.300 (300 in the optimization) sequences per class, where at each step we only sample from sequences that have at least one underrepresented label. The selection of hyperparameters by the BOHB hyperparameter optimizations for ProteoGAN and by the hyperparameter searches for the baselines are done on the validation set, while the results presented in the main text were acquired on the test set. For sequence sample generation the model was conditioned on the label combinations of the evaluation/test set and the resulting sequences then compared with the respective set by MMD and MRR.

OOD holdout sets were manually selected based on their usefulness in real-world application and the amount of samples they comprise (Table S2).

Table S1. Chemophysical properties and their accession numbers in the AAIndex.

Property name	Accession number
Molecular Weight	FASG760101
Hydrophobicity	FASG890101
Hydrophilicity	HOPT810101
Polarity	RADA880108
Amphiphilicity	MITS020101
Positive Charge	FAUJ880111
Negative Charge	FAUJ880112
pK	JOND750102
Isoelectric Point	ZIMJ680104
Probability Helix	KANM800101
Probability Sheet	KANM800102
Average Accessible Surface Area	JANJ780101
Buriability	ZHOH040103
Linker Index	BAEK050101

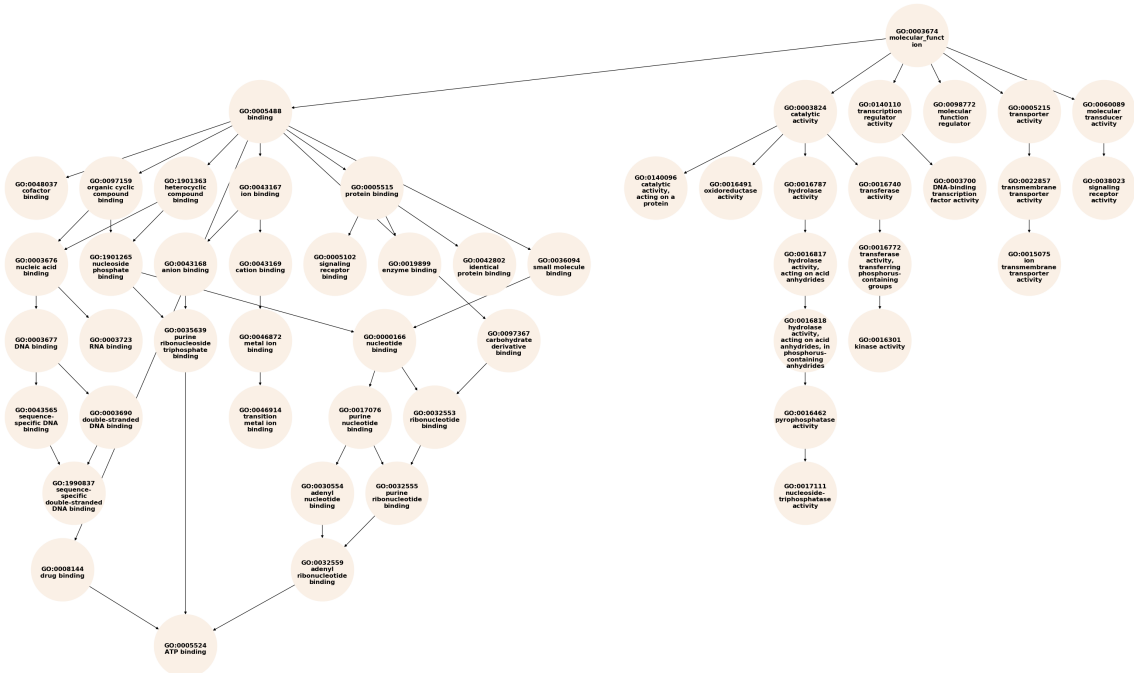


Fig. S1: GO DAG of the 50 labels selected for this project.

Table S2. Label combinations held-out from the training dataset for OOD evaluation.

ID	Labels	$\text{Size}_{\text{Train}}$	$\text{Size}_{\text{Holdout}}$
A	drug binding & transmembrane transporter activity	10446	1678
B	catalytic activity & cofactor binding & anion binding	5427	4027
C	small molecule binding & signaling receptor activity & catalytic activity, acting on a protein & kinase activity	7676	1079
D	ion transmembrane transporter activity & cation binding	7945	1109
E	DNA binding & RNA binding	13207	1435

$\text{Size}_{\text{Train}}$ = Size of smallest class in X_{Train}, $\text{Size}_{\text{Holdout}}$ = Size of X_{Holdout}

D Baselines

We implement several baselines to put the performance of our model into perspective. In this section, we would like to give additional details concerning the baselines that we gathered from the literature.

CVAE (Greener *et al.*, 2018) uses a conditional VAE (CVAE) in order to generate either metalloproteins with desired metal binding sites or fold properties. In the case of fold properties, the authors introduce iterative sampling and guidance steps in the latent space. The decoder and encoder are both MLPs and the number of layers is chosen with hyperparameter search. We introduced a KL-balancing term to stabilize training. We performed a Bayesian Optimization hyperparameter search for which we tried $1,000$ combinations of hyperparameters. Notably, we allowed for an optimization of network architecture by optimizing over the layer numbers for both encoder and decoder, and by optimizing the number of units in the first layer of the encoder and the last layer of the decoder. The unit number then halved towards the latent space with each layer. The hyperparameters and their value ranges, as well as the final model configuration can be found in Table S3. We refer the reader to Greener *et al.* (2018) for more information on the model.

The HMM baselines were implemented based on HMMER (Eddy, 2004). For *OpC-HMM*, all sequences in the training dataset containing a specific *label combination* were aggregated, for each of the ca. 1.800 label combinations of the test set. For *OpL-HMM*, all sequences in the training dataset containing a specific *label* were aggregated, for each of the 50 labels. The resulting sequence sets were aligned with MAFFT (with parameters `-retree 1 -maxiterate 0 -ep 0.123`). Because of the time-intensive multiple sequence alignment the sequences sets were randomly sampled to have a maximum size of 5000 sequences. From the alignment, a profileHMM was built with HMMER which was then sampled to generate a sequence.

In the n-gram baseline also, sequences were selected according to label sets (OpC) and single labels (OpL). Here the full data was used. n was set to 3. The sequence lengths were sampled from the sequence length distribution of the training data.

Table S3. CVAE of Greener et al. hyperparameters subject to BO optimization.

Name		Values	Final Value
Learning rate		[1e-5,1e-2]	7.8e-4
Pretrain start		[1,5000]	2598
Pretrain end		[1,5000]	1251
Latent dimension		[10,1000]†	761
KL balancing	β	[1e-3,100]†	1.1e-3
Encoder layer number		[1,5]	3
Decoder layer number		[1,5]	1
Log2(Encoder first layer units)		[4,10]	7
Log2(Decoder last layer units)		[4,10]	9

† Values were sampled on a logarithmic scale.

E Hyperparameter search

For ProteoGAN we conducted hyperparameter searches with the Bayesian Optimization and HyperBand (BOHB) algorithm. The Hyperband (Li *et al.*, 2017b) algorithm uses successive halving (Jamieson and Talwalkar, 2016) to evaluate a number of models on a given budget of resources. The better half of the models are then evaluated on twice the budget, et cetera. Hyperband is an independent optimization algorithm that has been combined with Bayesian optimization to form Bayesian optimization and Hyperband (BOHB) (Falkner *et al.*, 2018), the optimization strategy used in this project.

We conducted two BOHB optimizations. For both we evaluated $1,000$ models. All networks were trained with the Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0$ and $\beta_2 = 0.9$ (following (Gulrajani *et al.*, 2017)). The optimization consisted first of a broad search among 23 hyperparameters and second, of a smaller and more specific search, among 9 selected hyperparameters. For the first BOHB optimization, an optimization iteration was defined as two epochs which we found through pilot experiments was the minimum time to observe a viable trend in the metrics. The parameters R and η (in the notation of Li *et al.* (2017b)) were set to 9 and 3, respectively, which allowed for a maximum training time of 18 epochs (22.5K gradient updates). The optimization objective was to maximize the ratio of metrics MMD/MRR on the validation set (the metrics are introduced in the main document). During the optimization, BOHB selected the models based on evaluations at the end of a training period. For the second optimization, we reduced the number of hyperparameters to only 9. We selected values for the other hyperparameters based on the analysis of the hyperparameter importance of the first optimization (see paragraph below). The hyperparameters that showed either no importance or that were detrimental to training were removed. For this second optimization, the smaller network size allowed for 3 epochs per iteration, resulting in a maximum training time of 27 epochs (1.2K gradient updates). The list of hyperparameters of the two BOHB optimizations and their ranges is presented in Table S4. The parameters of the best models selected by the two BOHB optimizations are presented Table S5.

Table S4. Hyperparameters subject to BOHB optimization.

Name	Symbol	Values
Use chemophysical properties		Yes, No, only
Label embedding		one-hot, node2vec, Poincaré
Conditioning mechanism		projection, AC, both
AC weighting factor	γ	[1, 1000]†
Label smoothing factor	θ	[0, 0.5]
Latent noise dimension	d_Z	[1, 1000]†
Input noise standard deviation	σ	[0, 1]
Generator learning rate	η_G	[1e-5, 1e-2]
Generator learning rate 2	η_{G2}	[1e-5, 1e-2]
Discriminator learning rate	η_D	[1e-5, 1e-2]
Discriminator learning rate 2	η_{D2}	[1e-5, 1e-2]
Training ratio	n_{critic}	[1, 10]
Learning rate schedule		constant, cosine, exponential
Schedule interval (in epochs)	i	[1, 18]
Generator layer number	n_G	[1, 10]
Discriminator layer number	n_D	[1, 10]
Strides	s	1, 2, 4, 8
Filter size	f	[3, 12]
Generator skip	h_G	[0, 10]
Discriminator skip	h_D	[0, 10]
Number of projections	n_P	[1, 5]
Output source layers	o_S	[1, 3]
Output label layers	o_L	[1, 3]

† Values were sampled on a logarithmic scale. AC = auxiliary classifier.

Table S5. Hyperparameters found in the first and second BOHB optimization. Values with an asterisk indicate the preset configurations in the second optimization.

Name	First	Second
Use chemophysical properties	Yes	No
Label embedding	one-hot	one-hot
Conditioning mechanism	both	both
AC weighting factor	178	135
Label smoothing factor	0.28	-*
Latent noise dimension	91	100*
Input noise standard deviation	0.29	-*
Generator learning rate	2.0e-3	4.1e-4
Generator learning rate 2	-	-*
Discriminator learning rate	8.5e-4	4.0e-4
Discriminator learning rate 2	-	-*
Training ratio	1	1*
Learning rate schedule	constant	constant*
Schedule interval (in epochs)	-	-*
Generator layer number	2	2*
Discriminator layer number	3	2*
Strides	4	8
Filter size	8	12
Generator skip	-	-*
Discriminator skip	-	-*
Number of projections	1	2
Output source layers	-	1*
Output label layers	2	1*

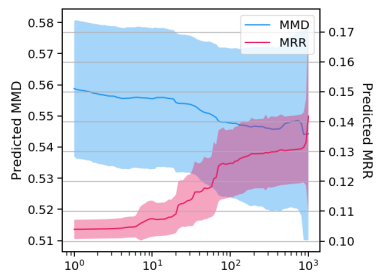
AC = auxiliary classifier.

F Hyperparameter analysis

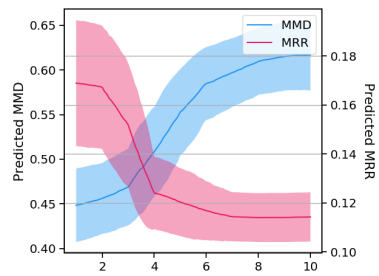
After the optimization we analyzed hyperparameter importance with the approach presented in (Hutter *et al.*, 2014). A surrogate model (random forest) is trained on the parameter configurations and the respective evaluation scores. This enables a functional analysis of variance (fANOVA) which allows for a quantification of hyperparameter importance in terms of the variance they explain. It also provides marginal predictions for each hyperparameter which gives insights about their optimal value setting. For the random forest, we used $1,000$ trees with a maximum depth of 64, and repeat the estimation 100 times. We do so for all evaluated models of the first and second BOHB optimizations. The hyperparameter importances obtained from the first optimization (and resp. second optimization) are presented in Figure S5 (resp. Figure S6). The first fANOVA showed that parameters related to the discriminator (learning rate, number of layers) are most important for model performance,[1] and helped to select potentially important hyperparameters for the second analysis. Noticeably, the best model of the first optimization was already a well-performing model, but we chose to run a second optimization to better understand the role of key hyperparameters to gain insight in potential good practices when designing conditional generative adversarial networks. The second fANOVA clarified the importance of the remaining hyperparameters, such as use of chemophysical features and label embeddings among others (Figure S6).

We also show marginal predictions for hyperparameters of the first optimization in Figure S2, and for the second optimization in Figure S3 and Figure S4.

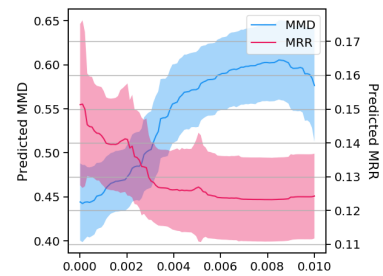
[1] Some other important factors were learning rate schedule-related parameters such as *Generator learning rate 2* or *schedule*. We realized that these were detrimental to model performance as the short duration of training in the optimization did not allow to estimate long term effects seen in the selected models that were trained for 100 epochs.



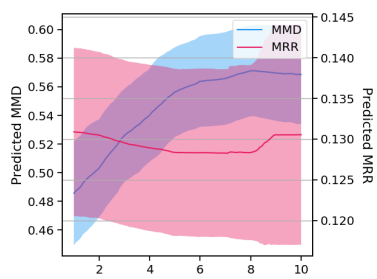
(a) Auxiliary Classifier weighing factor



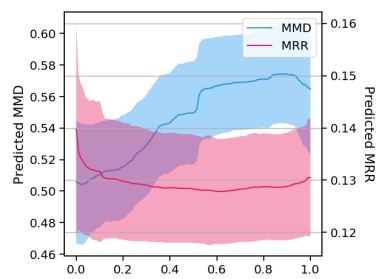
(b) Discriminator layer number



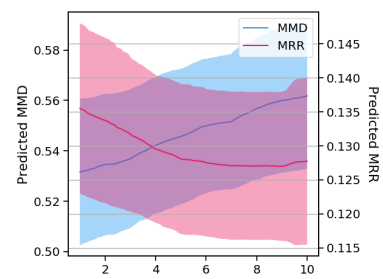
(c) Discriminator learning rate



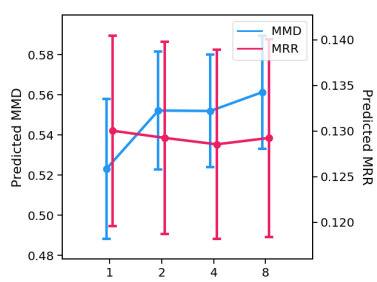
(d) Generator layer number



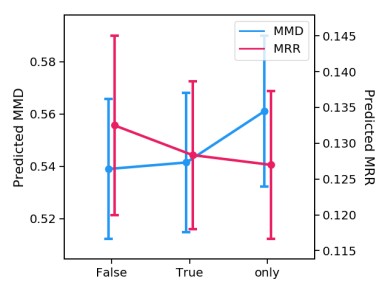
(e) Input noise



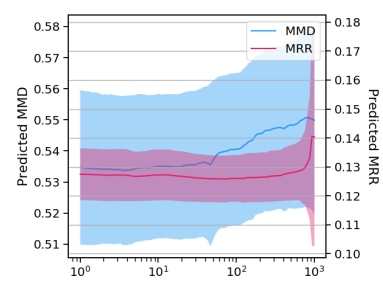
(f) Training ratio



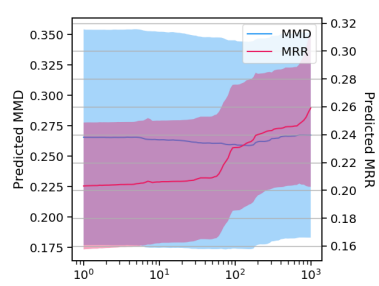
(g) Strides



(h) Use chemophysical features

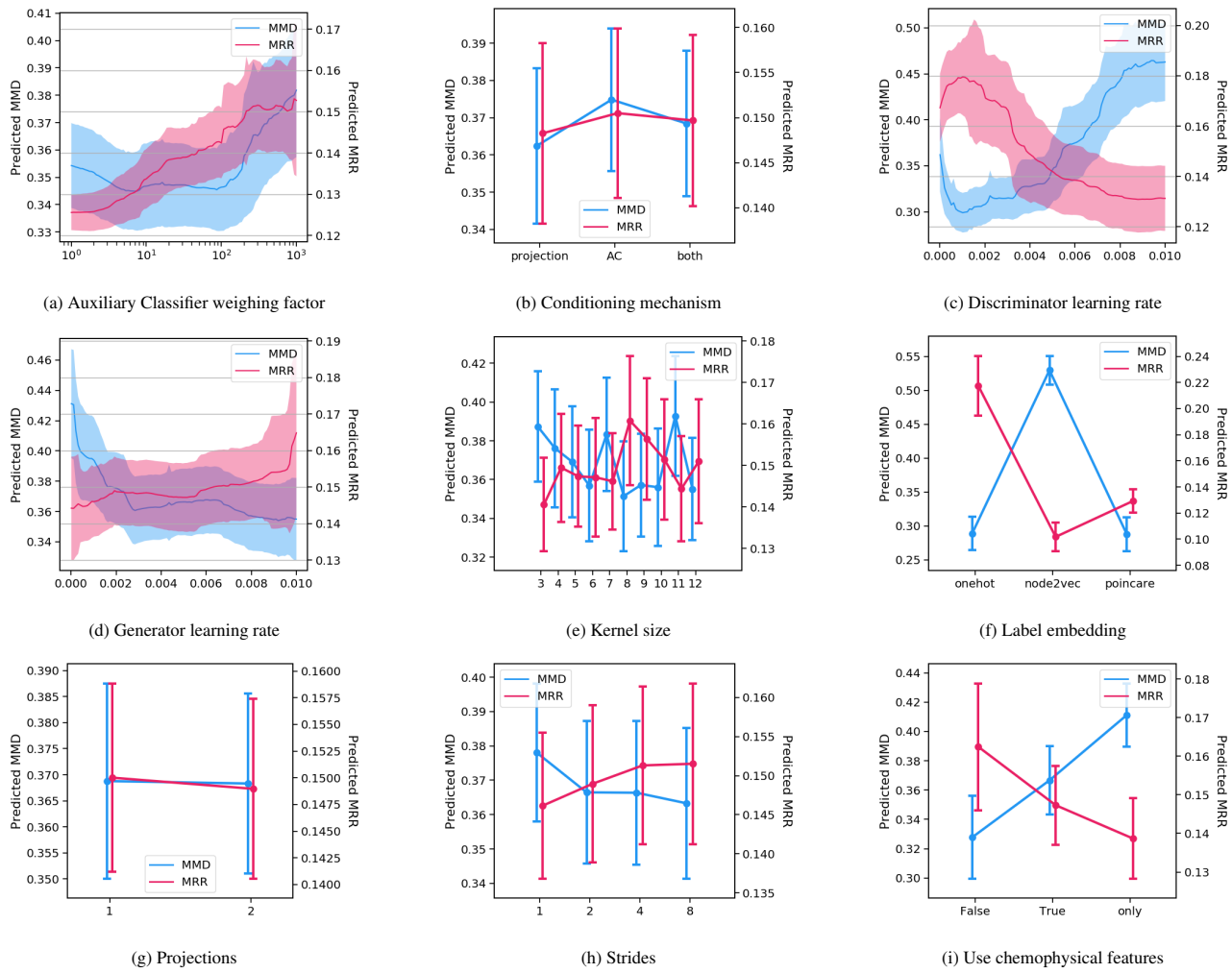


(i) Latent dimensionality



(j) Auxiliary Classifier weighing factor (Selected)

Fig. S2: Marginal predictions of hyperparameters based on data in the first optimization. We show some selected predictions that allowed for interpretation, all others were inconclusive. If not otherwise noted, data comes from all trials in the optimization. Predictions were obtained training on MMD and MRR. Note that for MMD, lower is better.



(a) Auxiliary Classifier weighing factor

(b) Conditioning mechanism

(c) Discriminator learning rate

(d) Generator learning rate

(e) Kernel size

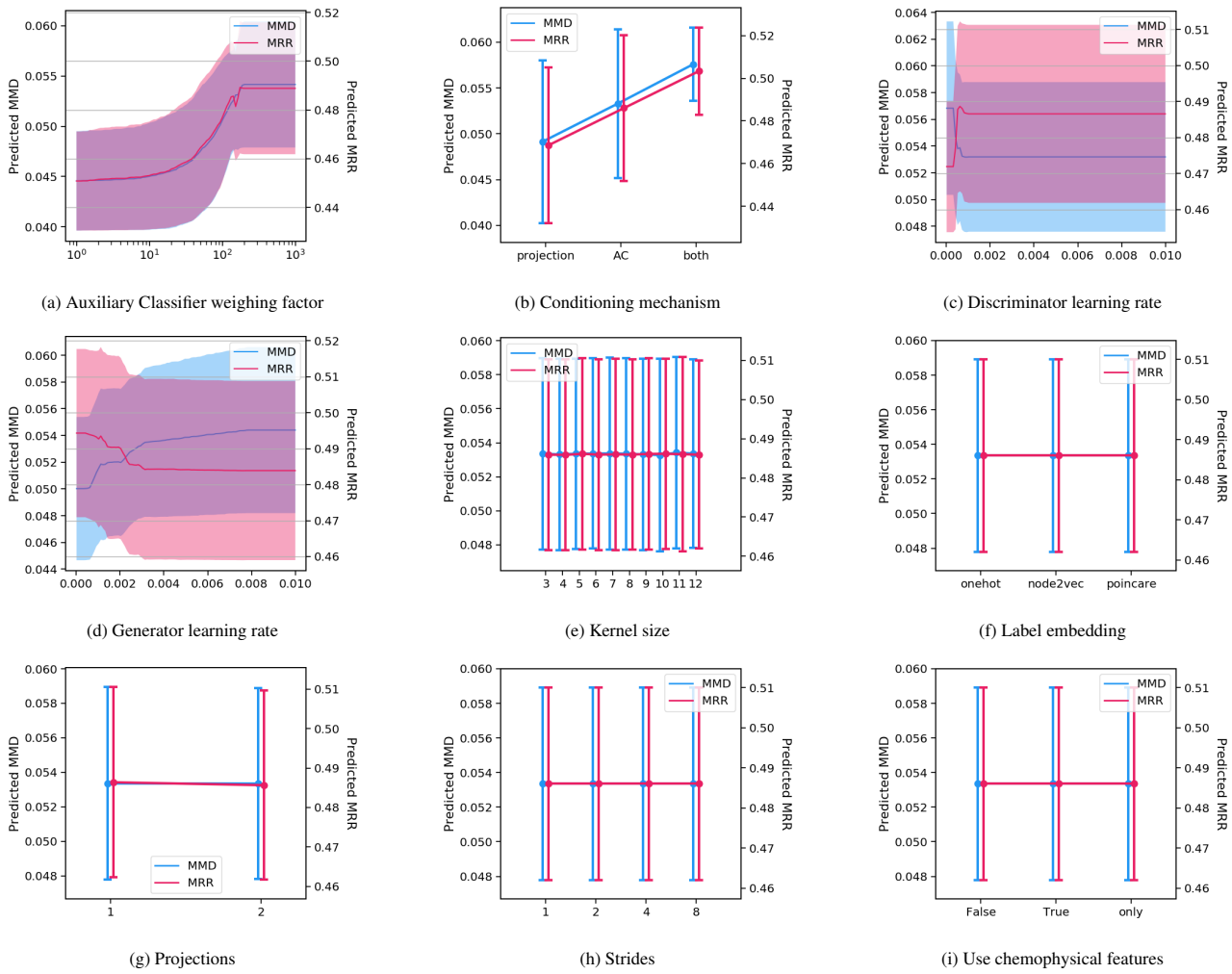
(f) Label embedding

(g) Projections

(h) Strides

(i) Use chemophysical features

Fig. S3: Marginal predictions of hyperparameters based on optimization data in the second optimization. Predictions were obtained training on MMD and MRR. Note that for MMD, lower is better.



(a) Auxiliary Classifier weighing factor

(b) Conditioning mechanism

(c) Discriminator learning rate

(d) Generator learning rate

(e) Kernel size

(f) Label embedding

(g) Projections

(h) Strides

(i) Use chemophysical features

Fig. S4: Marginal predictions of hyperparameters based on the data of the 27 best selected models in the second optimization. Predictions were obtained training on MMD and MRR. Note that for MMD, lower is better.

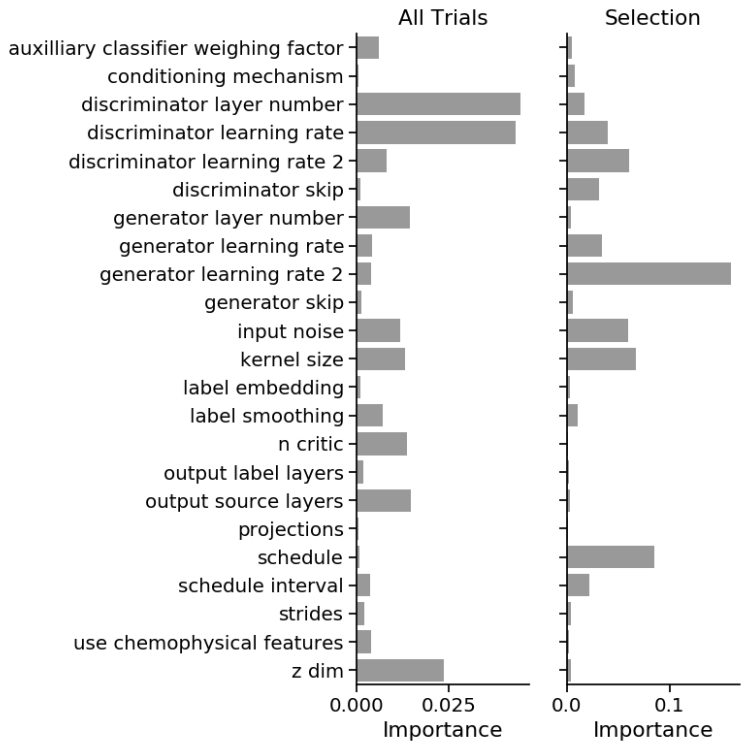


Fig. S5: Hyperparameter importance for the first BOHB optimization. Shown are all hyperparameters subject to optimization for all models (left), and a manual selection of models that was trained for 100 epochs (right).

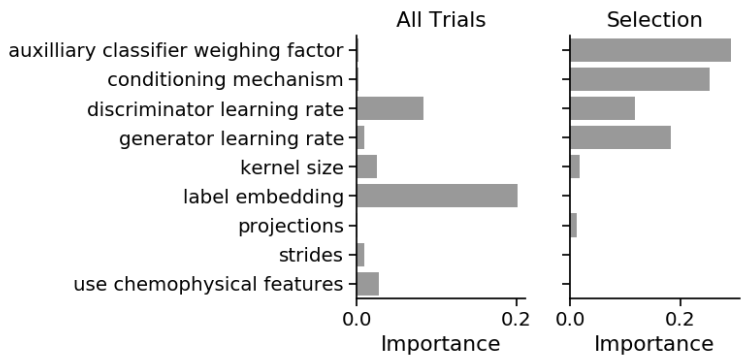


Fig. S6: Hyperparameter importance for the second BOHB optimization. The bars show individual importance of each hyperparameter in terms of the variance they explain. We conducted the analysis for all trials of the optimization (left) and for the selected models that were trained for prolonged time (100 epochs) (right). The total variance explained by the main effects was 36% and 88%, respectively.

G Losses, Duality Gap and real-time evaluation of ProteoGAN

The loss function of the final model presented in the main document, combining projection and auxiliary classifier, is shown Figure S7. We monitored the duality gap (red), for which we split the training data into an adversary finding set and a test set of 1% of the train set each (these sets are required additionally to estimate the duality gap, compare Grnarova *et al.* (2019)). The duality gap is well-behaved, with a fast convergence to 0, indicating that there is no mode collapse and suggesting that the samples are of reasonable quality. Also, the evaluations of MMD and MRR can be seen during training (evaluated twice per epoch) which provides valuable information for model selection and early stopping.

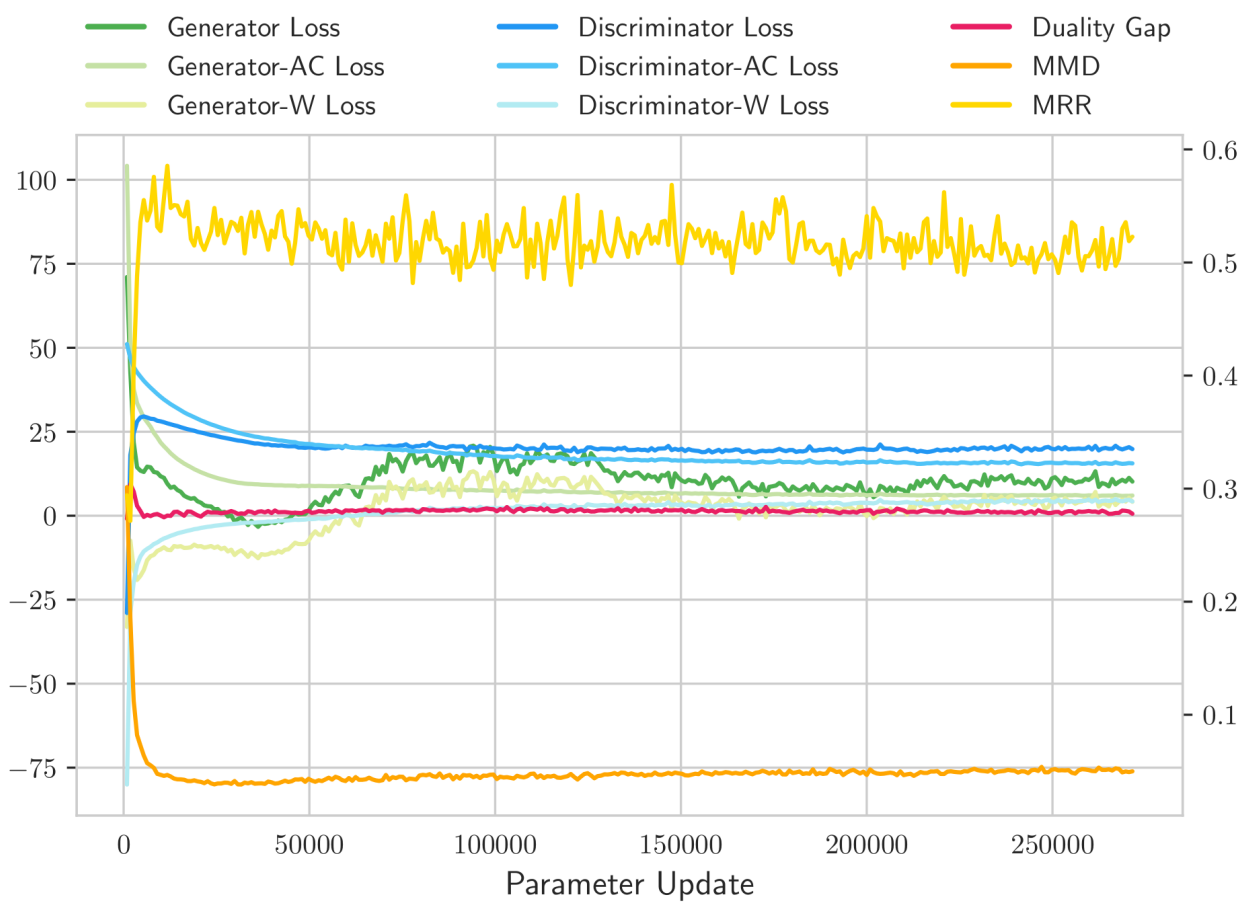


Fig. S7: Losses and evaluations at training time. W = Wasserstein, AC = Auxiliary Classifier

H Control for overfitting

To control that our model is not simply reproducing training examples we computed pairwise distances in feature space and verified that generated sequences were not closer to the training set than the test set (Figure S8).

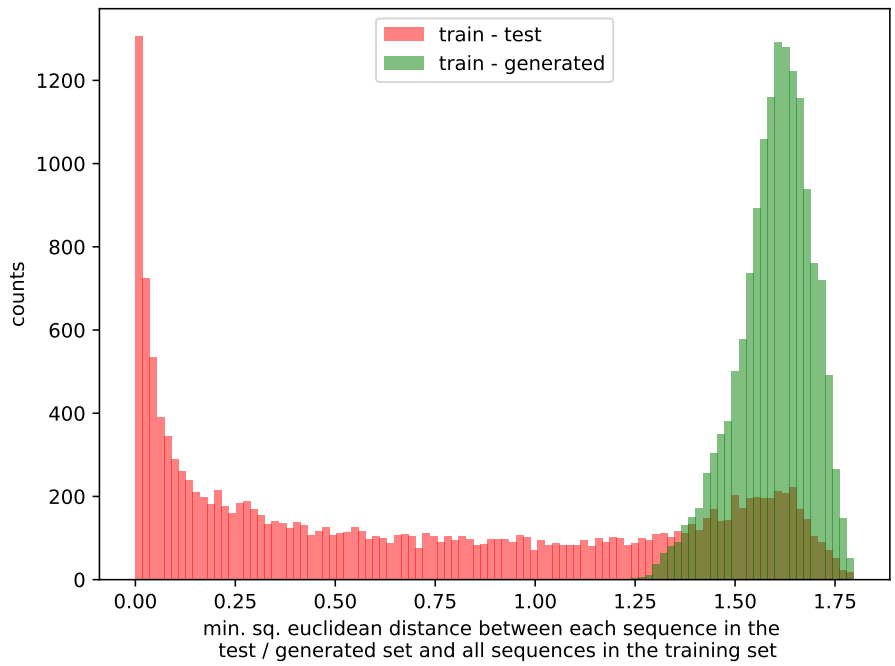


Fig. S8: Distributions of pairwise distances in Spectrum kernel feature space between a test set and the training set (red) and a generated set of ProteoGAN and the training set (green). It can be seen that the generated sequences are not closer to the training set than the testset (which would indicate overfitting). Further the generated sequences are about as far, but not further, away from the training set than the testset.

I Breakdown of conditional performance per label

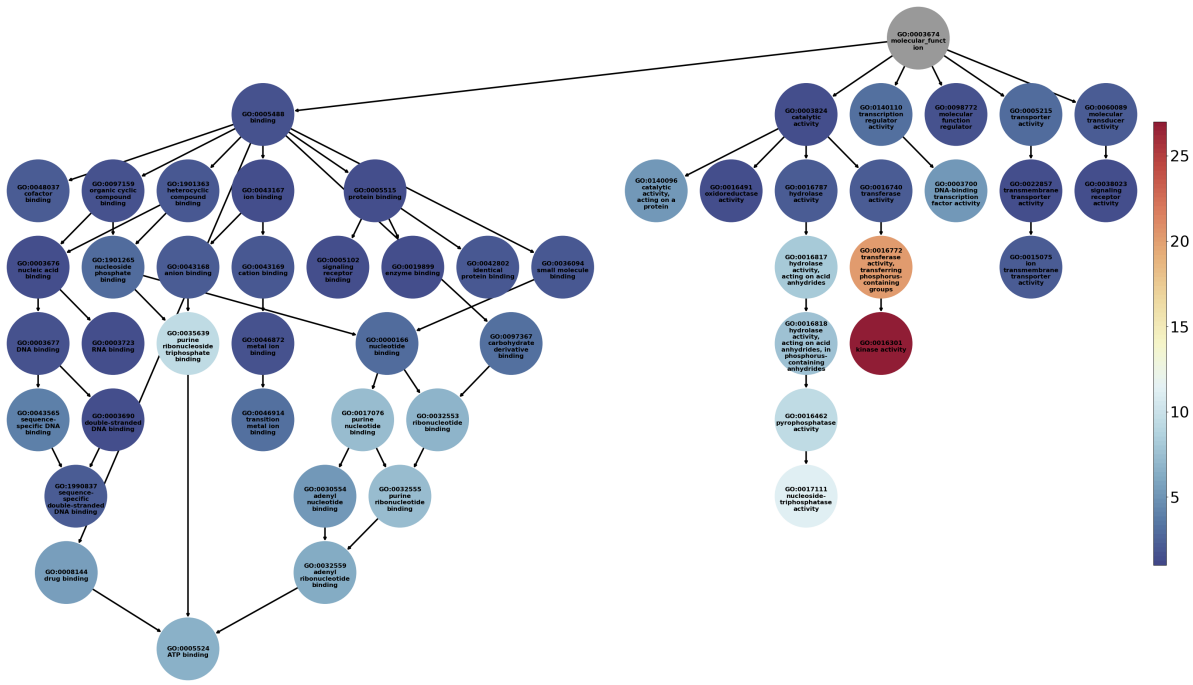


Fig. S9: Mean rank of each individual label in Spectrum MRR over five data splits. The structure represents the relations of the 50 labels of interest in the GO DAG. Nodes are colored by how well the model can target them. Dark blue indicates that the model can target the function well.

J Model evaluation with ProFET and UniRep

Table S6. Evaluation of ProteoGAN and other models with proposed metrics, based on the ProFET embedding (compare results in the main text).

	MMD↓	gauss. MMD↓	MRR↑	MRR_B ↑	ΔEntropy	ΔDistance
Positive Control	0.020 ± 0.005	0.011 ± 0.000	0.830 ± 0.027	0.924 ± 0.019	-0.004 ± 0.006	-0.010 ± 0.060
Negative Control	11.704 ± 0.005	1.037 ± 0.000	0.090 ± 0.000	0.119 ± 0.000	7.208 ± 0.010	6.374 ± 0.056
ProteoGAN	0.163 ± 0.014	0.060 ± 0.004	0.594 ± 0.050	0.766 ± 0.063	-0.047 ± 0.025	-0.328 ± 0.217
Predictorguided	0.091 ± 0.005	0.034 ± 0.002	0.131 ± 0.016	0.168 ± 0.018	0.032 ± 0.023	0.151 ± 0.147
Non-Hierarchical	1.144 ± 0.242	0.262 ± 0.031	0.288 ± 0.045	0.402 ± 0.075	-0.215 ± 0.231	-7.006 ± 4.127
CVAE	1.086 ± 0.281	0.211 ± 0.018	0.287 ± 0.035	0.408 ± 0.039	-0.145 ± 0.106	-2.841 ± 2.097
OpC-ngram	0.548 ± 0.014	0.069 ± 0.001	0.257 ± 0.016	0.340 ± 0.024	-0.005 ± 0.009	-6.804 ± 0.264
OpC-HMM	0.689 ± 0.012	0.289 ± 0.003	0.170 ± 0.012	0.235 ± 0.013	1.135 ± 0.013	2.937 ± 0.057
OpL-GAN	0.161	0.069	0.644	0.807	0.196	0.981
OpL-ngram	0.580 ± 0.015	0.083 ± 0.001	0.267 ± 0.022	0.325 ± 0.021	0.022 ± 0.008	-6.738 ± 0.327
OpL-HMM	0.798 ± 0.013	0.323 ± 0.004	0.176 ± 0.013	0.236 ± 0.014	1.330 ± 0.006	3.089 ± 0.038
ProteoGAN (100 labels)	0.124	0.057	0.537	0.746	-0.041	-0.547
ProteoGAN (200 labels)	0.245	0.109	0.297	0.412	0.314	0.856

Table S7. Evaluation of ProteoGAN and other models with proposed metrics, based on the UniRep embedding (compare results in the main text).

	MMD↓	gauss. MMD↓	MRR↑	MRR_B ↑	ΔEntropy	ΔDistance
Positive Control	0.038 ± 0.005	0.011 ± 0.000	0.855 ± 0.040	0.946 ± 0.034	-0.000 ± 0.004	-0.050 ± 0.185
Negative Control	10.618 ± 0.002	1.002 ± 0.000	0.090 ± 0.000	0.119 ± 0.000	8.121 ± 0.007	27.981 ± 0.169
ProteoGAN	1.512 ± 0.019	0.130 ± 0.006	0.244 ± 0.013	0.300 ± 0.011	0.866 ± 0.027	12.611 ± 0.731
Predictorguided	1.591 ± 0.012	0.151 ± 0.004	0.097 ± 0.004	0.121 ± 0.005	0.964 ± 0.017	15.004 ± 0.263
Non-Hierarchical	2.408 ± 0.456	0.132 ± 0.020	0.170 ± 0.037	0.225 ± 0.034	0.689 ± 0.166	4.790 ± 3.039
CVAE	1.808 ± 0.203	0.100 ± 0.023	0.213 ± 0.016	0.276 ± 0.017	0.632 ± 0.109	7.201 ± 3.752
OpC-ngram	2.062 ± 0.009	0.209 ± 0.002	0.147 ± 0.005	0.192 ± 0.014	0.943 ± 0.006	16.176 ± 0.161
OpC-HMM	2.083 ± 0.021	0.313 ± 0.003	0.116 ± 0.007	0.146 ± 0.011	1.529 ± 0.011	18.765 ± 0.182
OpL-GAN	1.721	0.194	0.224	0.278	1.108	16.853
OpL-ngram	2.154 ± 0.008	0.240 ± 0.002	0.159 ± 0.006	0.224 ± 0.008	1.025 ± 0.011	17.424 ± 0.185
OpL-HMM	2.263 ± 0.026	0.369 ± 0.007	0.120 ± 0.018	0.150 ± 0.017	1.749 ± 0.028	19.904 ± 0.428
ProteoGAN (100 labels)	1.446	0.113	0.225	0.294	0.820	10.707
ProteoGAN (200 labels)	2.059	0.286	0.126	0.177	1.393	21.001

K P-values corresponding to MMD tests

Table S8. Empirical p-values (Monte-Carlo simulation, n=1000) to the corresponding MMD tests for three embeddings and all models of the main results (mean and standard deviation over five data splits). Recall that in the case of a good generative model we would like to confirm the null hypothesis that the two samples originate from the same distribution, and thus expect a high p-value.

Model	Spectrum	ProFET	UniRep
Positive Control	0.435 ± 0.360	0.523 ± 0.419	0.675 ± 0.255
Negative Control	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
ProteoGAN	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
Predictorguided	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
Non-Hierarchical	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
CVAE	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
OpC-ngram	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
OpC-HMM	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
OpL-GAN	0.000	0.000	0.000
OpL-ngram	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
OpL-HMM	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
ProteoGAN (100 labels)	0.000	0.000	0.000
ProteoGAN (200 labels)	0.000	0.000	0.000

L Kolmogorov-Smirnov statistics

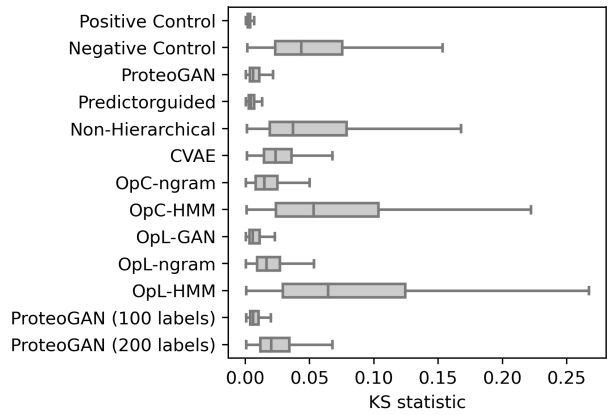


Fig. S10: KS statistics over 8.000 Spectrum kernel features, lower is better.

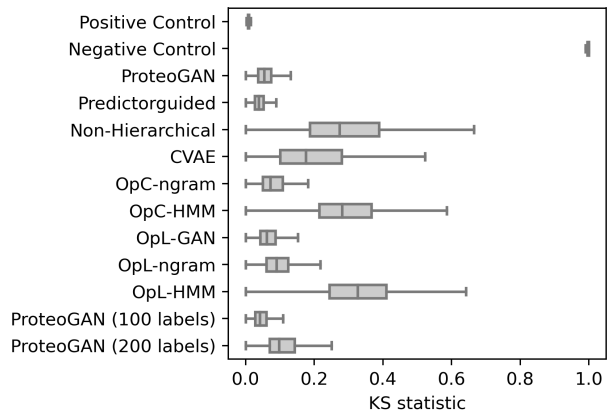


Fig. S11: KS statistics over ca. 500 ProFET biological features, lower is better.

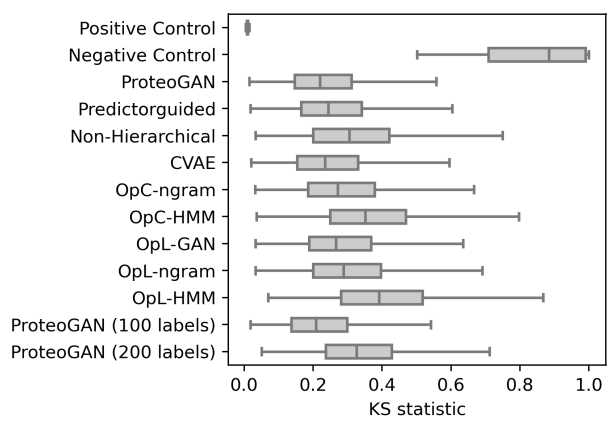


Fig. S12: KS statistics over 1.900 UniRep biological features, lower is better.

M Model evaluation with homology-controlled test sets

Table S9. Model evaluation with homology controlled test sets and the Spectrum embedding. Sequences were allowed to have up to 90% homology with the training set. Compare main result tables.

Model	MMD↓	gauss. MMD↓	MRR↑	MRR_B↑	ΔEntropy	ΔDistance
Positive Control	0.018 ± 0.001	0.013 ± 0.000	0.769 ± 0.021	0.902 ± 0.023	-0.037 ± 0.009	0.005 ± 0.002
Negative Control	1.014 ± 0.000	0.802 ± 0.000	0.090 ± 0.000	0.097 ± 0.000	0.688 ± 0.007	1.848 ± 0.002
ProteoGAN	0.045 ± 0.002	0.030 ± 0.001	0.527 ± 0.015	0.702 ± 0.035	-0.056 ± 0.012	0.017 ± 0.004
Predictorguided	0.030 ± 0.001	0.021 ± 0.001	0.114 ± 0.004	0.141 ± 0.007	-0.028 ± 0.010	0.007 ± 0.003
Non-Hierarchical	0.336 ± 0.118	0.238 ± 0.088	0.298 ± 0.048	0.421 ± 0.057	-0.416 ± 0.201	0.295 ± 0.170
CVAE	0.231 ± 0.077	0.155 ± 0.055	0.282 ± 0.019	0.398 ± 0.035	0.205 ± 0.027	0.151 ± 0.086
OpC-ngram	0.055 ± 0.002	0.036 ± 0.001	0.338 ± 0.016	0.408 ± 0.020	0.196 ± 0.007	-0.045 ± 0.002
OpC-HMM	0.175 ± 0.003	0.117 ± 0.002	0.099 ± 0.008	0.145 ± 0.007	-0.686 ± 0.018	0.204 ± 0.005
OpL-GAN	0.042	0.028	0.493	0.686	-0.107	0.027
OpL-ngram	0.058 ± 0.002	0.038 ± 0.001	0.324 ± 0.011	0.384 ± 0.014	0.220 ± 0.009	-0.047 ± 0.002
OpL-HMM	0.200 ± 0.002	0.135 ± 0.001	0.098 ± 0.009	0.145 ± 0.009	-0.784 ± 0.013	0.249 ± 0.004
ProteoGAN (100 labels)	0.040	0.027	0.574	0.769	-0.072	0.024
ProteoGAN (200 labels)	0.163	0.113	0.298	0.455	0.056	0.056

Table S10. Model evaluation with homology controlled test sets and the ProFET embedding. Sequences were allowed to have up to 90% homology with the training set. Compare main result tables.

Model	MMD↓	gauss. MMD↓	MRR↑	MRR_B↑	ΔEntropy	ΔDistance
Positive Control	0.066 ± 0.011	0.020 ± 0.002	0.723 ± 0.035	0.848 ± 0.033	0.039 ± 0.024	0.100 ± 0.096
Negative Control	11.673 ± 0.007	1.037 ± 0.000	0.090 ± 0.000	0.119 ± 0.000	7.340 ± 0.009	6.484 ± 0.090
ProteoGAN	0.184 ± 0.014	0.066 ± 0.004	0.507 ± 0.022	0.682 ± 0.030	-0.448 ± 0.041	-0.217 ± 0.247
Predictorguided	0.138 ± 0.006	0.043 ± 0.002	0.130 ± 0.016	0.166 ± 0.033	0.018 ± 0.023	0.261 ± 0.177
Non-Hierarchical	1.152 ± 0.259	0.262 ± 0.031	0.290 ± 0.044	0.403 ± 0.075	-0.346 ± 0.224	-6.896 ± 4.099
CVAE	1.070 ± 0.270	0.208 ± 0.017	0.286 ± 0.014	0.395 ± 0.025	-0.523 ± 0.031	-2.730 ± 2.080
OpC-ngram	0.526 ± 0.015	0.070 ± 0.001	0.261 ± 0.009	0.329 ± 0.010	0.062 ± 0.015	-6.693 ± 0.273
OpC-HMM	0.736 ± 0.015	0.297 ± 0.003	0.177 ± 0.011	0.255 ± 0.017	-0.324 ± 0.044	3.048 ± 0.090
OpL-GAN	0.220	0.080	0.581	0.730	-0.138	1.109
OpL-ngram	0.561 ± 0.017	0.084 ± 0.001	0.259 ± 0.008	0.319 ± 0.010	0.079 ± 0.020	-6.627 ± 0.333
OpL-HMM	0.846 ± 0.010	0.331 ± 0.003	0.198 ± 0.008	0.261 ± 0.013	-0.620 ± 0.017	3.200 ± 0.071
ProteoGAN (100 labels)	0.150	0.059	0.500	0.699	-0.446	-0.419
ProteoGAN (200 labels)	0.253	0.113	0.277	0.399	-0.428	0.984

Table S11. Model evaluation with homology controlled test sets and the UniRep embedding. Sequences were allowed to have up to 90% homology with the training set. Compare main result tables.

Model	MMD↓	gauss. MMD↓	MRR↑	MRR_B↑	ΔEntropy	ΔDistance
Positive Control	0.116 ± 0.018	0.017 ± 0.001	0.766 ± 0.034	0.892 ± 0.020	0.020 ± 0.008	0.255 ± 0.218
Negative Control	10.627 ± 0.003	1.002 ± 0.000	0.090 ± 0.000	0.116 ± 0.002	8.144 ± 0.013	28.286 ± 0.206
ProteoGAN	1.509 ± 0.018	0.130 ± 0.006	0.244 ± 0.009	0.305 ± 0.005	0.072 ± 0.021	12.916 ± 0.758
Predictorguided	1.592 ± 0.012	0.152 ± 0.004	0.096 ± 0.006	0.126 ± 0.008	0.419 ± 0.112	15.309 ± 0.282
Non-Hierarchical	2.437 ± 0.471	0.132 ± 0.020	0.174 ± 0.033	0.229 ± 0.035	0.098 ± 0.276	5.095 ± 3.032
CVAE	1.804 ± 0.205	0.100 ± 0.023	0.212 ± 0.019	0.267 ± 0.022	0.063 ± 0.085	7.507 ± 3.770
OpC-ngram	2.060 ± 0.007	0.209 ± 0.001	0.145 ± 0.013	0.179 ± 0.017	0.624 ± 0.015	16.481 ± 0.194
OpC-HMM	2.121 ± 0.019	0.313 ± 0.003	0.122 ± 0.003	0.155 ± 0.005	0.290 ± 0.032	19.070 ± 0.211
OpL-GAN	1.729	0.195	0.236	0.281	0.456	17.203
OpL-ngram	2.153 ± 0.006	0.240 ± 0.002	0.160 ± 0.012	0.215 ± 0.017	0.673 ± 0.007	17.729 ± 0.209
OpL-HMM	2.302 ± 0.021	0.370 ± 0.007	0.128 ± 0.007	0.160 ± 0.010	0.132 ± 0.035	20.209 ± 0.452
ProteoGAN (100 labels)	1.443	0.114	0.245	0.302	-0.073	11.057
ProteoGAN (200 labels)	2.053	0.287	0.130	0.162	0.018	21.351

Table S12. Model evaluation with homology controlled test sets and the Spectrum embedding. Sequences were allowed to have up to 70% homology with the training set. Compare main result tables.

Model	MMD↓	gauss. MMD↓	MRR↑	MRR_B↑	ΔEntropy	ΔDistance
Positive Control	0.026 ± 0.001	0.019 ± 0.000	0.545 ± 0.033	0.738 ± 0.018	-0.055 ± 0.010	0.006 ± 0.002
Negative Control	1.016 ± 0.001	0.803 ± 0.000	0.090 ± 0.000	0.098 ± 0.000	0.671 ± 0.007	1.849 ± 0.002
ProteoGAN	0.049 ± 0.002	0.032 ± 0.001	0.416 ± 0.032	0.580 ± 0.034	-0.073 ± 0.011	0.018 ± 0.004
Predictorguided	0.035 ± 0.001	0.024 ± 0.001	0.133 ± 0.006	0.169 ± 0.010	-0.046 ± 0.007	0.008 ± 0.003
Non-Hierarchical	0.337 ± 0.117	0.238 ± 0.088	0.293 ± 0.041	0.407 ± 0.054	-0.434 ± 0.200	0.296 ± 0.171
CVAE	0.235 ± 0.077	0.157 ± 0.055	0.256 ± 0.017	0.368 ± 0.013	0.188 ± 0.031	0.152 ± 0.086
OpC-ngram	0.056 ± 0.002	0.037 ± 0.001	0.298 ± 0.006	0.378 ± 0.009	0.179 ± 0.009	-0.044 ± 0.002
OpC-HMM	0.175 ± 0.003	0.117 ± 0.002	0.110 ± 0.008	0.158 ± 0.010	-0.704 ± 0.017	0.206 ± 0.005
OpL-GAN	0.044	0.030	0.368	0.533	-0.116	0.027
OpL-ngram	0.059 ± 0.002	0.039 ± 0.001	0.296 ± 0.015	0.378 ± 0.025	0.203 ± 0.009	-0.046 ± 0.002
OpL-HMM	0.201 ± 0.002	0.136 ± 0.001	0.102 ± 0.010	0.150 ± 0.011	-0.802 ± 0.014	0.251 ± 0.003
ProteoGAN (100 labels)	0.046	0.031	0.511	0.681	-0.081	0.024
ProteoGAN (200 labels)	0.163	0.113	0.280	0.417	0.047	0.056

Table S13. Model evaluation with homology controlled test sets and the ProFET embedding. Sequences were allowed to have up to 70% homology with the training set. Compare main result tables.

Model	MMD↓	gauss. MMD↓	MRR↑	MRR_B↑	ΔEntropy	ΔDistance
Positive Control	0.092 ± 0.011	0.028 ± 0.001	0.672 ± 0.054	0.807 ± 0.053	0.066 ± 0.053	0.292 ± 0.140
Negative Control	11.675 ± 0.011	1.036 ± 0.001	0.090 ± 0.000	0.119 ± 0.000	7.368 ± 0.033	6.676 ± 0.136
ProteoGAN	0.210 ± 0.012	0.070 ± 0.003	0.473 ± 0.023	0.627 ± 0.030	-0.421 ± 0.058	-0.025 ± 0.298
Predictorguided	0.140 ± 0.005	0.047 ± 0.002	0.127 ± 0.027	0.165 ± 0.025	0.045 ± 0.051	0.453 ± 0.206
Non-Hierarchical	1.159 ± 0.254	0.259 ± 0.032	0.292 ± 0.041	0.392 ± 0.066	-0.319 ± 0.217	-6.704 ± 4.068
CVAE	1.075 ± 0.272	0.206 ± 0.017	0.263 ± 0.029	0.348 ± 0.052	-0.496 ± 0.050	-2.538 ± 2.059
OpC-ngram	0.510 ± 0.019	0.070 ± 0.001	0.239 ± 0.008	0.294 ± 0.011	0.089 ± 0.034	-6.501 ± 0.294
OpC-HMM	0.732 ± 0.016	0.296 ± 0.004	0.164 ± 0.011	0.240 ± 0.013	-0.297 ± 0.042	3.239 ± 0.136
OpL-GAN	0.216	0.082	0.575	0.721	-0.139	1.280
OpL-ngram	0.541 ± 0.021	0.083 ± 0.001	0.251 ± 0.008	0.311 ± 0.012	0.106 ± 0.045	-6.436 ± 0.370
OpL-HMM	0.844 ± 0.011	0.331 ± 0.002	0.183 ± 0.010	0.255 ± 0.030	-0.593 ± 0.031	3.392 ± 0.119
ProteoGAN (100 labels)	0.178	0.062	0.476	0.641	-0.448	-0.248
ProteoGAN (200 labels)	0.239	0.112	0.276	0.386	-0.429	1.154

Table S14. Model evaluation with homology controlled test sets and the UniRep embedding. Sequences were allowed to have up to 70% homology with the training set. Compare main result tables.

Model	MMD↓	gauss. MMD↓	MRR↑	MRR_B↑	ΔEntropy	ΔDistance
Positive Control	0.163 ± 0.008	0.022 ± 0.000	0.759 ± 0.019	0.884 ± 0.024	0.015 ± 0.013	0.383 ± 0.344
Negative Control	10.628 ± 0.004	1.002 ± 0.000	0.090 ± 0.000	0.112 ± 0.003	8.139 ± 0.007	28.414 ± 0.355
ProteoGAN	1.497 ± 0.018	0.130 ± 0.007	0.228 ± 0.012	0.288 ± 0.016	0.067 ± 0.023	13.044 ± 0.874
Predictorguided	1.574 ± 0.014	0.152 ± 0.004	0.099 ± 0.006	0.125 ± 0.007	0.414 ± 0.119	15.437 ± 0.413
Non-Hierarchical	2.439 ± 0.468	0.132 ± 0.021	0.171 ± 0.029	0.228 ± 0.033	0.092 ± 0.276	5.223 ± 3.010
CVAE	1.812 ± 0.201	0.100 ± 0.024	0.209 ± 0.026	0.259 ± 0.024	0.058 ± 0.087	7.634 ± 3.876
OpC-ngram	2.047 ± 0.016	0.209 ± 0.001	0.137 ± 0.010	0.164 ± 0.013	0.618 ± 0.012	16.608 ± 0.325
OpC-HMM	2.094 ± 0.023	0.313 ± 0.003	0.122 ± 0.009	0.148 ± 0.011	0.284 ± 0.039	19.197 ± 0.361
OpL-GAN	1.725	0.196	0.212	0.274	0.454	17.548
OpL-ngram	2.137 ± 0.017	0.240 ± 0.002	0.159 ± 0.012	0.203 ± 0.009	0.668 ± 0.013	17.857 ± 0.338
OpL-HMM	2.275 ± 0.026	0.369 ± 0.007	0.123 ± 0.007	0.148 ± 0.010	0.126 ± 0.040	20.337 ± 0.532
ProteoGAN (100 labels)	1.452	0.114	0.235	0.292	-0.076	11.403
ProteoGAN (200 labels)	2.052	0.288	0.127	0.161	0.015	21.697

Table S15. Model evaluation with homology controlled test sets and the Spectrum embedding. Sequences were allowed to have up to 50% homology with the training set. Compare main result tables.

	MMD↓	gauss. MMD↓	MRR↑	MRR_B↑	ΔEntropy	ΔDistance
Positive Control	0.038 ± 0.000	0.027 ± 0.000	0.299 ± 0.013	0.447 ± 0.016	-0.081 ± 0.009	0.006 ± 0.003
Negative Control	1.018 ± 0.001	0.804 ± 0.000	0.090 ± 0.000	0.099 ± 0.001	0.645 ± 0.007	1.849 ± 0.002
ProteoGAN	0.056 ± 0.003	0.038 ± 0.002	0.310 ± 0.032	0.426 ± 0.041	-0.099 ± 0.012	0.019 ± 0.004
Predictorguided	0.042 ± 0.002	0.030 ± 0.001	0.114 ± 0.007	0.150 ± 0.014	-0.071 ± 0.009	0.008 ± 0.004
Non-Hierarchical	0.338 ± 0.117	0.239 ± 0.088	0.276 ± 0.045	0.405 ± 0.057	-0.460 ± 0.201	0.296 ± 0.169
CVAE	0.239 ± 0.076	0.160 ± 0.054	0.192 ± 0.018	0.299 ± 0.022	0.162 ± 0.031	0.152 ± 0.085
OpC-ngram	0.060 ± 0.002	0.041 ± 0.001	0.236 ± 0.008	0.321 ± 0.015	0.153 ± 0.008	-0.044 ± 0.003
OpC-HMM	0.176 ± 0.003	0.118 ± 0.002	0.103 ± 0.013	0.157 ± 0.025	-0.730 ± 0.020	0.206 ± 0.006
OpL-GAN	0.051	0.035	0.235	0.359	-0.147	0.028
OpL-ngram	0.061 ± 0.002	0.041 ± 0.002	0.272 ± 0.017	0.371 ± 0.020	0.177 ± 0.010	-0.046 ± 0.003
OpL-HMM	0.201 ± 0.002	0.136 ± 0.002	0.099 ± 0.011	0.153 ± 0.025	-0.828 ± 0.016	0.251 ± 0.004
ProteoGAN (100 labels)	0.056	0.038	0.377	0.486	-0.112	0.025
ProteoGAN (200 labels)	0.165	0.115	0.197	0.313	0.016	0.057

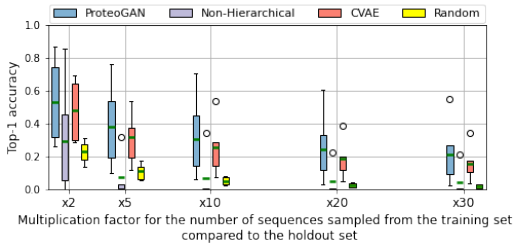
Table S16. Model evaluation with homology controlled test sets and the ProFET embedding. Sequences were allowed to have up to 50% homology with the training set. Compare main result tables.

	MMD↓	gauss. MMD↓	MRR↑	MRR_B↑	ΔEntropy	ΔDistance
Positive Control	0.136 ± 0.010	0.044 ± 0.001	0.548 ± 0.027	0.675 ± 0.012	0.084 ± 0.049	0.752 ± 0.263
Negative Control	11.668 ± 0.014	1.034 ± 0.001	0.090 ± 0.000	0.119 ± 0.000	7.386 ± 0.039	7.136 ± 0.259
ProteoGAN	0.244 ± 0.021	0.074 ± 0.005	0.445 ± 0.053	0.602 ± 0.061	-0.403 ± 0.053	0.434 ± 0.397
Predictorguided	0.158 ± 0.013	0.056 ± 0.004	0.130 ± 0.021	0.171 ± 0.035	0.063 ± 0.065	0.913 ± 0.338
Non-Hierarchical	1.163 ± 0.250	0.253 ± 0.032	0.265 ± 0.044	0.375 ± 0.077	-0.301 ± 0.203	-6.244 ± 4.028
CVAE	1.083 ± 0.277	0.202 ± 0.016	0.246 ± 0.035	0.343 ± 0.051	-0.478 ± 0.033	-2.078 ± 1.973
OpC-ngram	0.493 ± 0.020	0.075 ± 0.001	0.229 ± 0.014	0.281 ± 0.018	0.107 ± 0.032	-6.041 ± 0.413
OpC-HMM	0.730 ± 0.021	0.298 ± 0.004	0.170 ± 0.017	0.246 ± 0.025	-0.279 ± 0.017	3.699 ± 0.259
OpL-GAN	0.231	0.090	0.438	0.573	-0.131	1.769
OpL-ngram	0.519 ± 0.025	0.086 ± 0.002	0.238 ± 0.032	0.289 ± 0.037	0.124 ± 0.029	-5.976 ± 0.392
OpL-HMM	0.841 ± 0.016	0.331 ± 0.004	0.168 ± 0.027	0.227 ± 0.043	-0.575 ± 0.035	3.851 ± 0.238
ProteoGAN (100 labels)	0.230	0.070	0.423	0.576	-0.440	0.241
ProteoGAN (200 labels)	0.243	0.116	0.252	0.339	-0.421	1.644

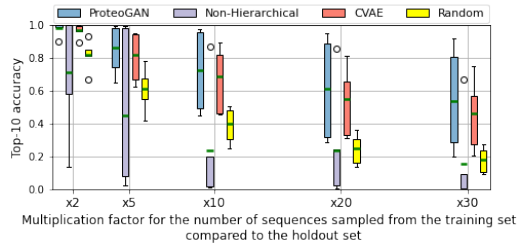
Table S17. Model evaluation with homology controlled test sets and the UniRep embedding. Sequences were allowed to have up to 50% homology with the training set. Compare main result tables.

	MMD↓	gauss. MMD↓	MRR↑	MRR_B↑	ΔEntropy	ΔDistance
Positive Control	0.292 ± 0.017	0.034 ± 0.001	0.725 ± 0.032	0.870 ± 0.022	-0.044 ± 0.018	0.577 ± 0.387
Negative Control	10.604 ± 0.004	1.002 ± 0.000	0.090 ± 0.000	0.106 ± 0.003	8.080 ± 0.013	28.608 ± 0.395
ProteoGAN	1.402 ± 0.020	0.129 ± 0.007	0.240 ± 0.018	0.289 ± 0.018	0.008 ± 0.026	13.238 ± 0.968
Predictorguided	1.474 ± 0.012	0.151 ± 0.006	0.095 ± 0.006	0.117 ± 0.006	0.355 ± 0.123	15.631 ± 0.314
Non-Hierarchical	2.369 ± 0.464	0.132 ± 0.021	0.176 ± 0.028	0.229 ± 0.028	0.034 ± 0.281	5.417 ± 2.844
CVAE	1.746 ± 0.214	0.100 ± 0.023	0.197 ± 0.013	0.246 ± 0.015	-0.001 ± 0.082	7.829 ± 3.602
OpC-ngram	1.976 ± 0.008	0.209 ± 0.003	0.134 ± 0.009	0.162 ± 0.011	0.560 ± 0.015	16.803 ± 0.342
OpC-HMM	1.981 ± 0.025	0.312 ± 0.004	0.118 ± 0.010	0.142 ± 0.009	0.226 ± 0.045	19.392 ± 0.433
OpL-GAN	1.589	0.192	0.203	0.262	0.376	17.222
OpL-ngram	2.065 ± 0.008	0.240 ± 0.003	0.149 ± 0.010	0.196 ± 0.011	0.609 ± 0.023	18.052 ± 0.444
OpL-HMM	2.162 ± 0.016	0.369 ± 0.006	0.118 ± 0.013	0.142 ± 0.013	0.067 ± 0.044	20.531 ± 0.630
ProteoGAN (100 labels)	1.334	0.111	0.250	0.308	-0.154	11.076
ProteoGAN (200 labels)	1.930	0.284	0.105	0.125	-0.063	21.371

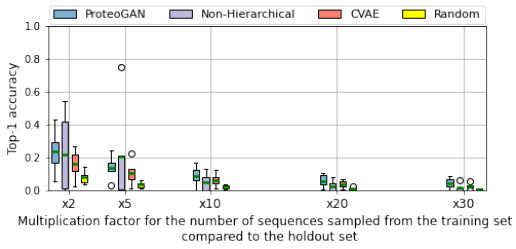
N OOD evaluation



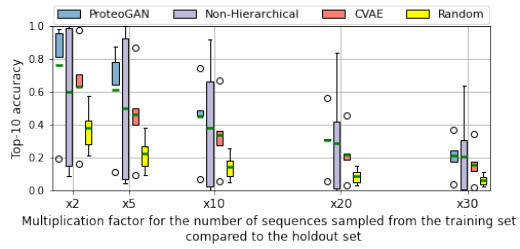
(a) Spectrum embeddings



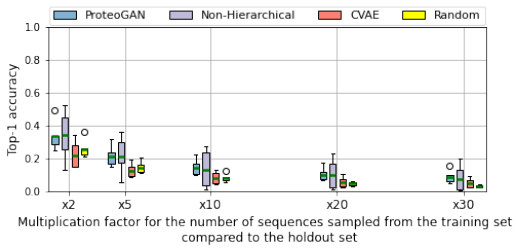
(b) Spectrum embeddings



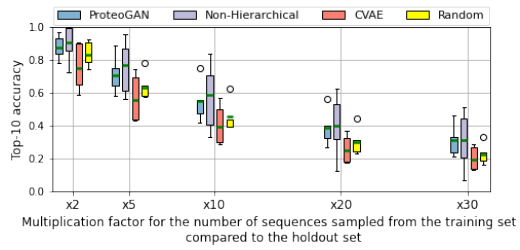
(c) Unirep embeddings



(d) Unirep embeddings



(e) ProFET embeddings



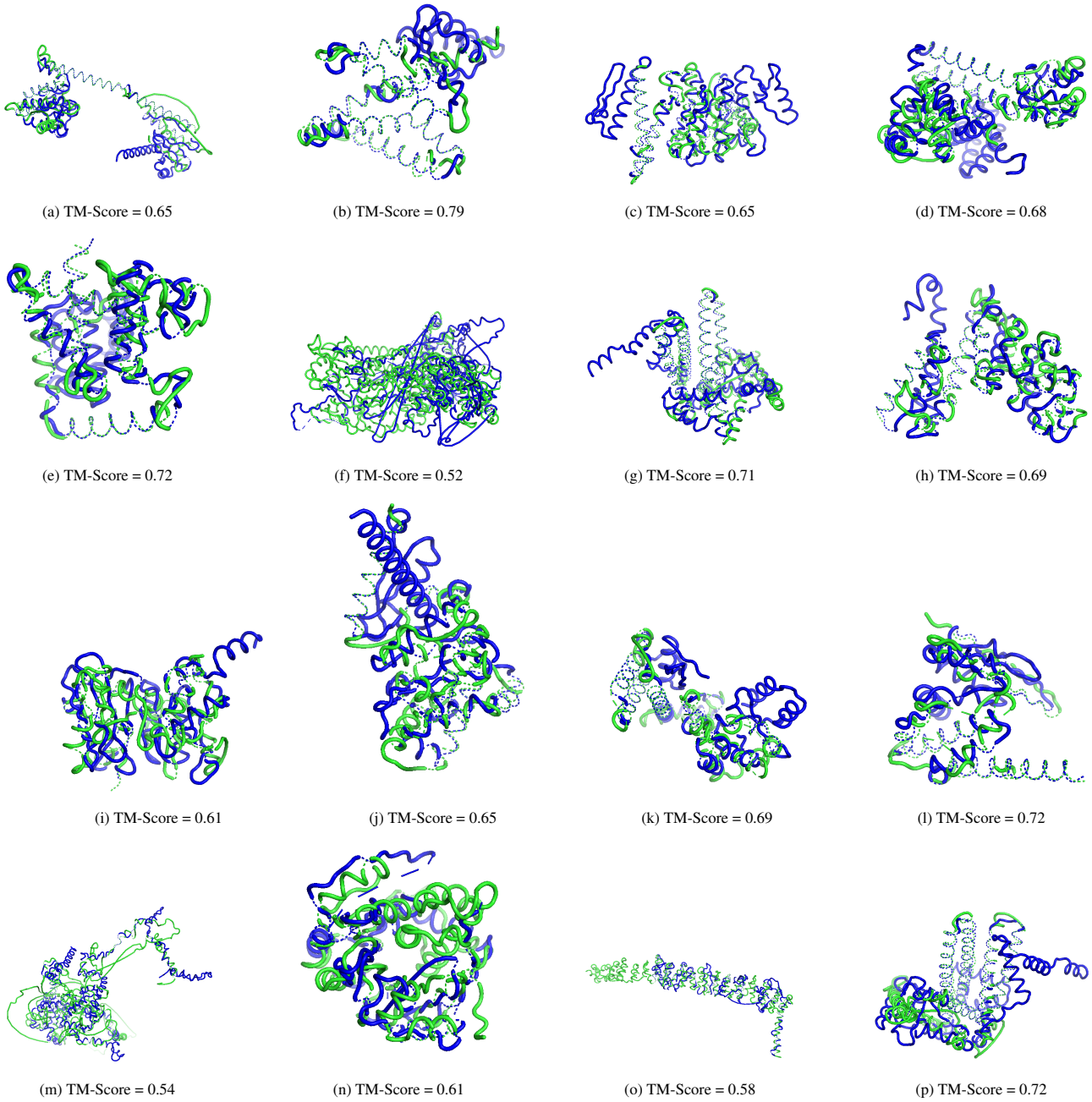
(f) ProFET embeddings

Fig. S13: Top-1 and Top-10 accuracies for ProteoGAN, *Non-Hierachical*, CVAE and a Random baseline, which consists in sampling real sequences from the training set. Distances between generated sequences and real sequences are calculated in the embedding spaces, either Spectrum, Unirep or ProFET.

O Structure predictions of ProteoGAN generated proteins

We obtained structure predictions for a small sample ($n = 50$) of generated sequences with trRosetta (Yang *et al.*, 2020) (max_iter= 200). Out of three predictions for each sequence we selected the best by choosing the one with minimal energy as predicted by the Rosetta all-atom energy function (Alford *et al.*, 2017). We then measure how much the designed proteins resemble natural folds by aligning (Deng *et al.*, 2019) their structures to the Protein Data Bank (PDB). The structural similarities are reported as TM-score Zhang and Skolnick (2004), where higher is better. Some exemplary structure alignments are shown in Figure S14, the average TM-score was 0.60 ± 0.09.

We note however that this prediction pipeline was not tested on generated sequences before and that it is not guaranteed to work as expected with natural proteins.



(a) TM-Score = 0.65 (b) TM-Score = 0.79 (c) TM-Score = 0.65 (d) TM-Score = 0.68

(e) TM-Score = 0.72 (f) TM-Score = 0.52 (g) TM-Score = 0.71 (h) TM-Score = 0.69

(i) TM-Score = 0.61 (j) TM-Score = 0.65 (k) TM-Score = 0.69 (l) TM-Score = 0.72

(m) TM-Score = 0.54 (n) TM-Score = 0.61 (o) TM-Score = 0.58 (p) TM-Score = 0.72

Fig. S14: Some exemplary ProteoGAN generated structures (blue) in comparison to their structurally closest homolog in PDB (green).