Conditional Generative Modeling for De Novo Hierarchical Multi-Label Functional Protein Design

1 Supplemental Information

1.1 Background: biological mechanisms underlying protein functions

Proteins are complex biological structures but can be simply represented as chains of amino-acids, a 20-character alphabet. While this is useful for modeling approaches, it hides the functional complexity of proteins. In fact, amino acids, with their diverse physico-chemical properties, fold and assemble into complex three dimensional constructs at a local and global level, giving rise to the overall protein structure. In turn, the structure of the protein is responsible for its function, where shape and electrochemical properties dictate the behavior and biological activity of the macromolecule. The link between sequence and function is therefore highly complex and still not understood by the research community. The design of proteins has consequently been so far mainly based on relatively uninformed trial-and-error processes with slight random alterations of the protein sequence and a subsequent assay of functionality [1] or entail computationally heavy simulations of molecular dynamics [23]. However, advances in machine learning have enabled the development of novel *in silico* design methods.

1.2 Methods

In this section, we first introduce three variants of MRR suited for hierarchically-structured labels. Then, we assess the proposed evaluation measures of cGANs by estimating empirical worst and best bounds for our experimental setting. Finally, we describe in details the state-of-the-art conditioning mechanisms and constructed variants that we used in this project.

1.2.1 Evaluation measures for conditional generative models in the case of hierarchical labels

In addition to the evaluation measures MMD and MRR described in the main document, we built three variants of MRR in order to better characterise the effectiveness of the conditional generation and its ability to handle hierarchical multi-label settings. We look at sub-measures of MRR where either all parent nodes (MRR_P), all child nodes (MRR_C), or both (MRR_B) are ignored in the ranking of a conditional MMD term. Removing parent or child terms attempts at understanding to which degree the conditioning mechanism leads to severe off-target generation of sequences of unrelated labels. Indeed, these three alternative measures do not penalize if the generated distribution of sequences for a given label is closer to either parents' or children's sequences than it is to its target's sequences, which is less severe than off-target generation of sequences of an unrelated label. Therefore, getting good MRR_X values would indicate that our model is able to conditionally generate

sequences up to closely related (parent's or child's) functions. Additionally, comparing MRR_X values between themselves and with MRR would give some insight in closely-related conditional generation performance. For example, a MRR_P (resp. MRR_C) value much larger than MRR could indicate that sequences that are generated with a target function are often closer to the natural sequences exhibiting the parent (resp. child) functional label, i.e. the model is too general, or too specific, respectively.

1.2.2 Assessment of the evaluation measures for cGANs

We assess the quality of the proposed evaluation measures by constructing "best case" and "worst case" scenarios, to understand what would represent a perfect success or failure mode of our model. We consider as "best case" the case where the generative model generates sequences that are observed. The "worst case" scenario differs depending on the evaluation measure.

Scenarios for MMD: MMD has theoretical bounds of $[0, \sqrt{2}]$ if the sequences in both sets are self-similar and totally dissimilar from each other. As we aim to compare real sequences to generated sequences that resemble real sequences, we therefore fix one set to be a collection of n natural protein sequences and the second set to be n other natural protein sequences modified with different percentages of random noise. In practice, the set of natural sequences is the test set used to report the results in the main document, and the random noise is injected in the form of single-point mutations to sequences of the second set. The results are reported in Table S1 and indicate that MMD is a proxy for the quality of the generated sequences. We observe that MMD increases with the amount of noise injected in the sequences of the second set. The generation of close to constant sequences is a plausible failure mode of the GAN and would lead to a very high MMD value (last row). The lengths of the sequences of the mutated set were conserved, however we also report MMD with respect to fully random sequences of maximum length. The MMD value between two sets of real sequences is around 0.0237, adding 1% of noise to the sequences in one of the set leads to an MMD value of 0.0240, 10% of noise to 0.0324 and 20% of noise to 0.0484. In comparison, in biology, proteins have been shown to be viable up to 30 - 60% of mutations in the amino-acids of their sequences [16, 19, 22]. We also report empirical p-values following Borgwardt et al. [2], under the null hypothesis that the two sets are from the same distribution. These were obtained by ranking the original MMD statistic in 1000 iterations of statistics where the aggregated sequences were randomly assigned to each of the two sets.

Scenarios for MRR: Since MRR is a conditional measure, we constructed the "worst case" sample as a set of natural protein sequences with randomized label assignments. This aims to simulate a generative model that produces well-formed sequences, but ignores the conditioning objective. One could also construct a scenario that simulates an antagonistic model that actively assigns wrong labels, instead of random ones. This will likely not occur in practice, though. Tables S7 and S8 show the MRR values for a real data sample and the same sample with randomized labels. The reference for MRR was again the test set and the evaluated sample an equally structured set ("Positive Control") where the label annotations were randomly shuffled among the sequences ("Negative Control"). The MRR evaluates a set of sequences with respect to the 50 selected labels. We also look at the sub-measures of MRR where either all parent terms (MRR_P), all child terms (MRR_C), or both (MRR_B) are ignored in the ranking of a term. This gives additional insights on how well the model works with respect to the up- and downstream labels in the GO DAG.

1.2.3 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 GANs' objective functions and variants used in this project.

Sample	MMD	p-value
Dataset Sample	0.0237	0.1499
Dataset Sample + 1% noise	0.0240	0.0370
Dataset Sample + 2% noise	0.0243	0.0050
Dataset Sample + 3% noise	0.0248	0
Dataset Sample + 5% noise	0.0262	0
Dataset Sample + 10% noise	0.0324	0
Dataset Sample + 20% noise	0.0484	0
Dataset Sample + 30% noise	0.0660	0
Dataset Sample + 50% noise	0.1009	0
Dataset Sample + 100% noise	0.1788	0
100% noise (maximum length)	0.3044	0
Constant (all leucine)	1.0258	0

Table S1: MMD values with different percentage of mutations, p-values

Loss function of conditional GANs: Our models are trained with the Wasserstein objective with gradient penalty from [9]. As a reminder, the WGAN-GP losses can be written as follows:

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

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 conditional setting cGAN with projection discriminator [18] (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 [9]. The discriminator and generator losses can be written as follows:

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

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 [18]: 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 [18] can therefore be written as:

$$D(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{A}(\boldsymbol{v}(\boldsymbol{y})^T \boldsymbol{\phi}_\theta(\boldsymbol{x}) + \psi_\gamma(\boldsymbol{\phi}_\theta(\boldsymbol{x}))) \tag{3}$$

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 [17], potentially thanks to the introduction of a form of regularization on the discriminator.

In this project, we also 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:

$$D(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{A}(\sum_{i=1}^{k} (\boldsymbol{v}_i(\boldsymbol{y})^T \boldsymbol{l}^i_{p_i} \circ \cdots \boldsymbol{l}^i_1(\boldsymbol{x}) + g_i(\boldsymbol{x}))) \tag{4}$$

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).

Case of the cGAN model with auxiliary classifier [20]: 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 2). In addition to notations introduced for Equation 2, 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:

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

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 equiped with auxiliary classifier or with multiple projections at several layers (see Equation 4)) 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 shows that the combination of both mechanisms helps to obtain a better performing conditioning mechanism, as measured by MRR.

1.3 Experimental setup

In this section, we describe in detail the dataset and preprocessing steps, the baselines and the hyperparameter searches performed.

1.3.1 Data

Sequence data is acquired from the UniProt Knowledgebase (UniProtKB, Consortium [4]). The database contains more than 180 million protein sequences with rich annotations such as structure and function. Nonetheless, most of these entries are only automatically annotated. To ensure high quality data for our model, we filter for sequences that are manually curated and have experimental evidence of some form. There are also some specialized proteins that have very long sequences, we only keep the sequences whose length is not exceeding 2048 amino acids, which covers ca. 98.5% of the data points. Five manually selected label combinations (named A-E) were held out from the training data, resulting in 149,390 sequences total. The cut-off at 2048 amino-acids is multiple times longer than other approaches in this field, which is between 30- to 500-long [5, 6, 8, 22], and allows for a more complete model of the known sequence space.

Functional labels are 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.

Train, validation and test splits were created to preferably represent all labels uniformly in the test and evaluation sets. We use a 80-10-10 split for evaluation for hyperparameter optimization and the results detailed in the supplement. For the validation and test set, we randomly sample sequences until there is at least 1.300 (300 in the optimization) sequences per class. The selections 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.

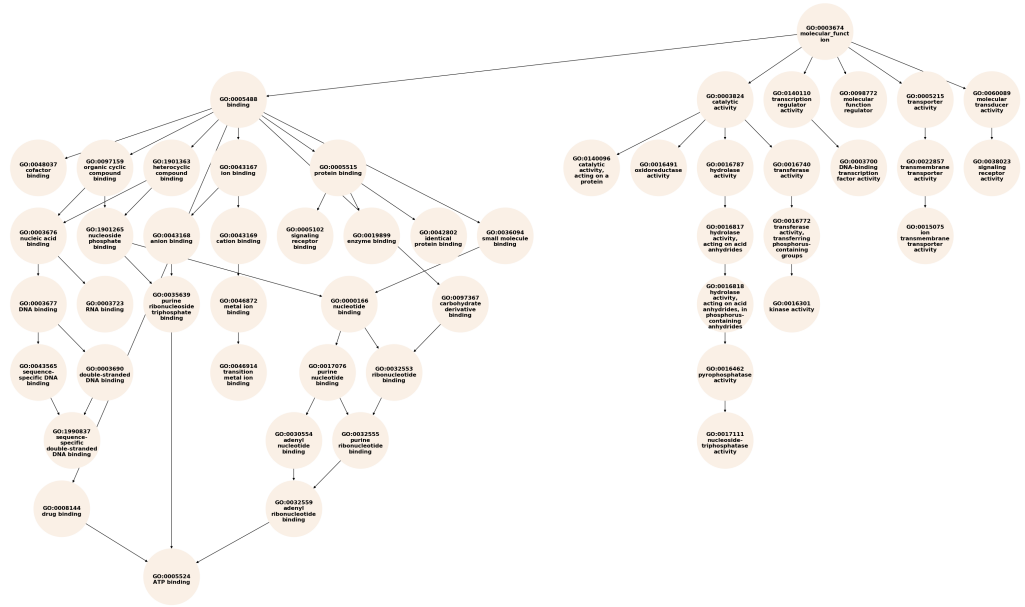


Figure S1: GO DAG of the 50 labels selected for this project.

1.3.2 Model Architecture

1.3.3 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 four baselines that we gathered from the literature.

CVAE [8] uses a conditional VAE (CVAE) in order to generate either metalloproteins with desired metallo-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. Here also, we introduced a KL-balancing term to stabilize training. As for PepCVAE, the model presents a loss scheduling scheme and therefore we could not use the BOHB optimization. However, 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. [8] for more information on the model.

The HMM baselines were implemented based on HMMER. For *OpC-HMM*, all sequences in the training dataset containing a specific label combination were aggregated, for each label set 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-intense 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

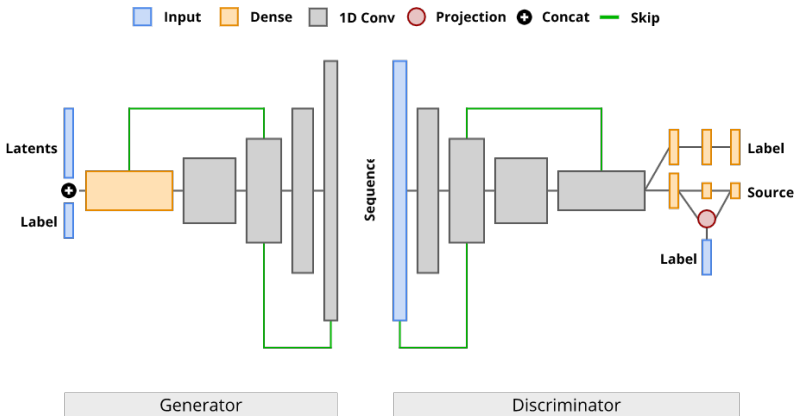


Figure S2: A schematic overview of the general model structure that we screened during hyperparameter search. Architectural features such as layer number and skip connections can vary. We note that the architecture of our final model differs from the one depicted here.

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 training data length distribution.

Initially, we also wanted to use as baseline PepCVAE [5]. It uses a VAE framework with a single layer gated recurrent unit (GRU) RNN for both the encoder and the decoder, in a semi-supervised setting. We implemented the model but could only trained it on short sequence of 32 amino-acids as it didnot scale further, the RNN component of the model was unfortunately highly resource-consuming. We therefore removed this baseline from the main document, but the training scheme is still described here for completeness. Conditioning was performed by concatenating the encoded target labels to the latent code. In the paper, the sequences of interest are antimicrobial peptides, with a maximum length of thirty amino-acids. The conditioning on label was binary, i.e. antimicrobial activity or not. We modified PepCVAE by introducing a multiplying factor in front the the KL term of the ELBO loss as suggested by [Higgins et al.], to increase the stability of the model. We optimise the model with a Bayesian Optimization hyperparameter search, for which we tried $1,000$ combinations of hyperparameters. We do not use BOHB because the early stopping would interfere with the different training phases of the model. The hyperparameters and their value ranges, as well as the final model configuration can be found in Table S2. For a fair assessment and to be able to compare to this model, we also ran our model ProteoGAN on sequences of 32 amino-acids for this experiment only. The results are only present in the supplementary material Figure S3 and Table S4 because obtained on too short sequences. We refer the reader to Das et al. [5], Hu et al. [11] and Bowman et al. [3] for more information on the model.

Table S2: PepCVAE hyperparameters subject to BO optimization.

Name		Values	Final Value
Learning rate		[1e-5,1e-2]	4e-3
Pretrain iterations		[1,5000]	3181
Latent dimension		[10,1000]†	101
Word dropout keep rate		[0,1]	0.43
Classifier loss balancing	λ_C	[1e-3,100]†	9.9e-3
Latent loss balancing	λ_Z	[1e-3,100]†	1.9e-1
KL balancing	β	[1e-3,100]†	1.5e-2

† Values were sampled on a logarithmic scale.

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.

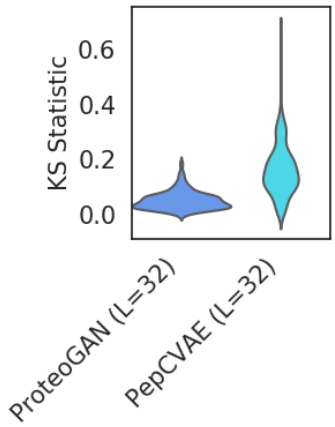


Figure S3: Sequence feature analysis of the models trained on truncated data (Sequence length = 32). KS statistics over $\sim$ 500 ProFET features, lower is better.

Table S4: Evaluation of ProteoGAN and PepCVAE on truncated sequences with length 32. Shown are mean and standard deviation of five different data splits. The arrows indicate that lower (↓) or higher (↑) is better.

Model	MMD↓	Gaussian MMD↓	MRR↑
PepCVAE (L=32)	0.122 ± 0.019	0.077 ± 0.012	0.139 ± 0.022
ProteoGAN (L=32)	**0.033 ± 0.002**	**0.022 ± 0.001**	**0.321 ± 0.029**

1.3.4 Hyperparameter search

Description of the BOHB: For ProteoGAN, we conducted hyperparameter searches with the Bayesian Optimization and HyperBand (BOHB) algorithm. The Hyperband [15] algorithm uses successive halving [13] 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) [7], the optimization strategy used in this project.

Hyperparameter optimization with BOHB: We conducted two BOHB optimizations. For both we evaluated $1,000$ models. All networks were trained with the Adam optimizer [14] with $\beta_1 = 0$ and $\beta_2 = 0.9$ (following [9]). 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 [15]) 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, in the validation set, the ratio of metrics MMD/MRR which 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 S5. The parameters of the best models selected by the two BOHB optimizations are presented Table S6.

Quantification of hyperparameter importance: After the optimization, we analyze hyperparameter importance with the approach presented in [12]. 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 S11 (resp. Figure S12). 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

[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

Table S5: 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 S6: 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.

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 practice when designing conditional generative adversarial networks and to further improve the performance of our model. The second fANOVA clarified the importance of the remaining hyperparameters, such as use of chemophysical features and label embeddings among others (Figure S12).

We also show marginal predictions for hyperparameters of the first optimization in Figure S13, and for the second optimization in Figure S9 and Figure S10.

Obtainment of the final model: The 27 best selected models of the second hyperparameter search were then trained for a prolonged duration of 100 epochs, where the conditioning mechanism and an associated weighing factor became most important, according to the last fANONA study (Figure S10). We evaluated twice per epoch and selected the weights of the final model at the checkpoint that showed the best (smallest) ratio MMD/MRR in the validation set. The final model, ProteoGAN, is a convolutional conditional generative adversarial network, with two conditioning mechanisms: an auxiliary classifier and projections. The dimensions of the convolutional layers are following the pyramidal architecture of DCGAN [21], i.e. with increasing output length and decreasing filter depth for the generator, and vice versa for the discriminator. The other hyperparameters are presented Table S6.

1.4 Meta-Evaluation of metrics

1.4.1 Sequence: ProFET features and kernel embeddings

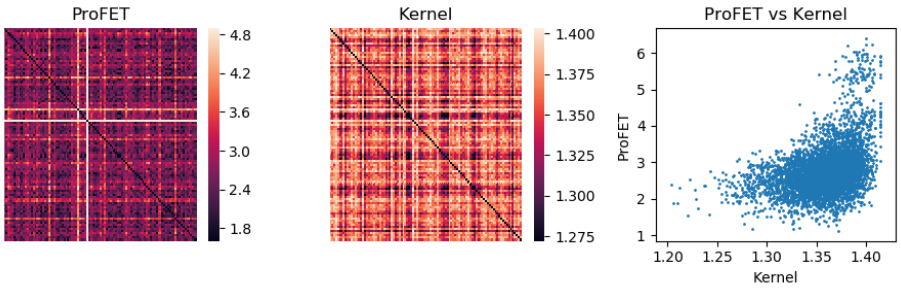


Figure S4: Pairwise distances of a random sample (n=200 sequences) of the dataset. Left: ProFET embedding, Middle: Kernel embedding, Right: Correlation between the pairwise distances of both embeddings (Self-similarity datapoints excluded for better visualization). The correlation (Spearman's rho) across 10 replicates with n=1000 sequences was 0.35 ± 0.01.

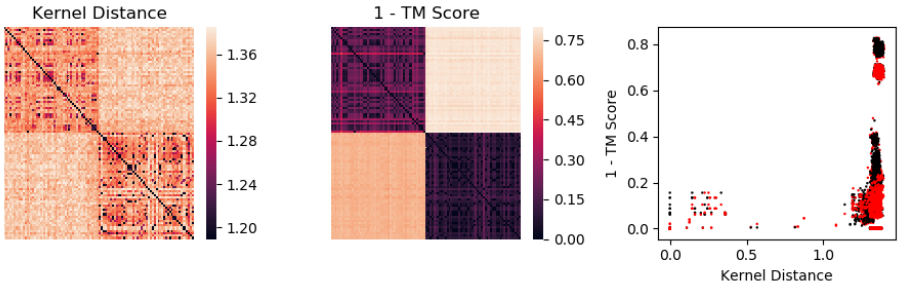
1.4.2 Structure: TM-Score versus kernel embeddings

1.5 Complement to the results in the main document

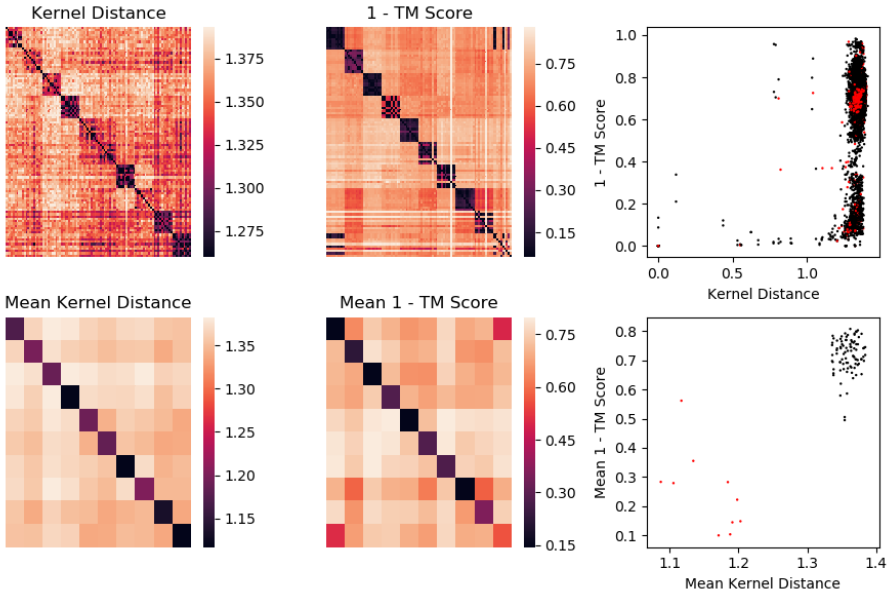
1.5.1 Results for MMD and MRR variants

Tables S7 and S8 show the results for the MMD and MRR variants for ProteoGAN and the baselines. The results confirm the conclusion drawn in the main document. The results are robust to the choice of kernel and of embedding (kmer counts or kmer indicators). The comparison between MRR

optimization did not allow to estimate long term effects seen in the selected models that were trained for 100 epochs.



(a) Pairwise distances for a random sample of N=100 sequences split evenly between P=2 randomly chosen protein families (50 sequences per family). Left: Kernel sequence embedding, Middle: 1 - TM-Score (TM-Score assesses similarity based on 3D structures), Right: Correlation between pairwise distances calculated with both measures. Distances between members of the same family (diagonal) are in red.



(b) Pairwise distances for a random sample of N=100 sequences split evenly between P=10 randomly chosen protein families (10 sequences per family). Left: Kernel sequence embedding, Middle: 1 - TM-Score (TM-Score assesses similarity based on 3D structures), Right: Correlation between pairwise distances calculated with both measures. Distances between members of the same family (diagonal) are in red. The first row shows individual proteins, the second row mean values per family. For individual proteins, the mean correlation (Spearman's rho) across 10 replicates was 0.17 ± 0.04 and the AUPR for data which was binarized at TM-Score= 0.5 was 0.53 ± 0.05. Per family, the mean correlation (Spearman's rho) across 10 replicates was 0.29 ± 0.07 and the AUPR for data which was binarized at TM-Score= 0.5 was 0.90 ± 0.07.

Figure S5: Correlation between TM-score, a 3D structure similarity measure, and the sequence-based kernel embedding.

Table S7: Evaluation of ProteoGAN and various baselines with our proposed measures (MMD (kmer counts), MRR (kmer counts), MMD_P (kmer counts) and MRR_C (kmer counts)) on the testset. An arrow indicates that lower (↓) or higher (↑) is better. Best results in bold, second best italicized. Given are mean values and standard deviation over five different data splits. Due to the computational effort, the OpL-GAN model was only trained on one split. The Positive Control uses as generated sequences a sample of real sequences and simulates a perfect model, the Negative Control is a sample that simulates the worst possible model for each metric (constant sequence for MMD, randomized labels for MRR).

Model	MMD (counts)↓	MRR (counts)↑	MRR_P (counts)↑	MRR_C (counts)↑
Pos. Control	0.011 ± 0.000	0.889 ± 0.017	0.923 ± 0.025	0.930 ± 0.016
Neg. Control	1.024 ± 0.000	0.092 ± 0.002	0.113 ± 0.008	0.093 ± 0.002
ProteoGAN	*0.049 ± 0.001*	**0.531 ± 0.024**	**0.627 ± 0.025**	**0.586 ± 0.040**
Pred.-Guided	**0.037 ± 0.001**	0.122 ± 0.001	0.161 ± 0.017	0.125 ± 0.006
Non-Hier.	0.342 ± 0.106	0.297 ± 0.033	0.315 ± 0.038	0.383 ± 0.034
CVAE	0.243 ± 0.070	*0.335 ± 0.038*	*0.447 ± 0.063*	*0.398 ± 0.041*
OpC-HMM	0.160 ± 0.002	0.125 ± 0.012	0.136 ± 0.015	0.169 ± 0.014
OpC-n-gram	0.069 ± 0.014	0.263 ± 0.014	0.296 ± 0.026	0.289 ± 0.016
OpL-GAN	**0.038**	**0.532**	**0.611**	**0.561**
OpL-HMM	0.186 ± 0.002	0.123 ± 0.010	0.136 ± 0.010	0.164 ± 0.013
OpL-n-gram	0.097 ± 0.000	0.090 ± 0.000	0.099 ± 0.000	0.092 ± 0.000
ProteoGAN100	0.047	0.438	0.470	0.472
ProteoGAN200	0.166	0.130	0.116	0.142

variants suggests that proteins often resemble proteins in the target class. When this is not the case, proteins are often similar to their parent class, which makes sense as the class is then more general. The small difference between the MRR_B value of our model and of the positive control indicates that the model rarely generates sequences that resemble proteins in an unrelated class. Additionally, compared to the controls, the MRR_C are relatively low compared to MRR, which suggests that the model does not tend to create more specific child labels.

1.5.2 Losses and real-time evaluation of the final model

The loss function of the final model presented in the main document, combining projection and auxiliary classifier, is shown Figure S6. 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. 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.

Table S8: Evaluation of ProteoGAN and various baselines with our proposed measures (MRR_B (kmer counts), MRR_P (indicator) and MRR_C (indicator)) on the testset. An arrow indicates that lower (↓) or higher (↑) is better. Best results in bold, second best italicized. Given are mean values and standard deviation over five different data splits. Due to the computational effort, the OpL-GAN model was only trained on one split. The Positive Control uses as generated sequences a sample of real sequences and simulates a perfect model, the Negative Control is a sample that simulates the worst possible model for each metric (constant sequence for MMD, randomized labels for MRR).

Model	MRR_B (counts)↑	MRR_P (indicator)↑	MRR_C (indicator)↑
Pos. Control	0.967 ± 0.013	0.924 ± 0.025	0.930 ± 0.016
Neg. Control	0.115 ± 0.009	0.113 ± 0.009	0.093 ± 0.002
ProteoGAN	**0.683 ± 0.039**	**0.637 ± 0.008**	**0.590 ± 0.027**
Pred.-Guided	0.167 ± 0.016	0.124 ± 0.009	0.112 ± 0.005
Non-Hier.	0.404 ± 0.035	0.288 ± 0.080	0.352 ± 0.091
CVAE	*0.494 ± 0.064*	*0.447 ± 0.031*	*0.412 ± 0.033*
OpC-HMM	0.181 ± 0.016	0.114 ± 0.010	0.151 ± 0.008
OpC-n-gram	0.425 ± 0.026	0.294 ± 0.028	0.285 ± 0.025
OpL-GAN	**0.643**	**0.021**	**0.543**
OpL-HMM	0.179 ± 0.016	0.112 ± 0.005	0.146 ± 0.004
OpL-n-gram	0.101 ± 0.000	0.090 ± 0.000	0.091 ± 0.000
ProteoGAN100	0.571	0.564	0.501
ProteoGAN200	0.183	0.153	0.125

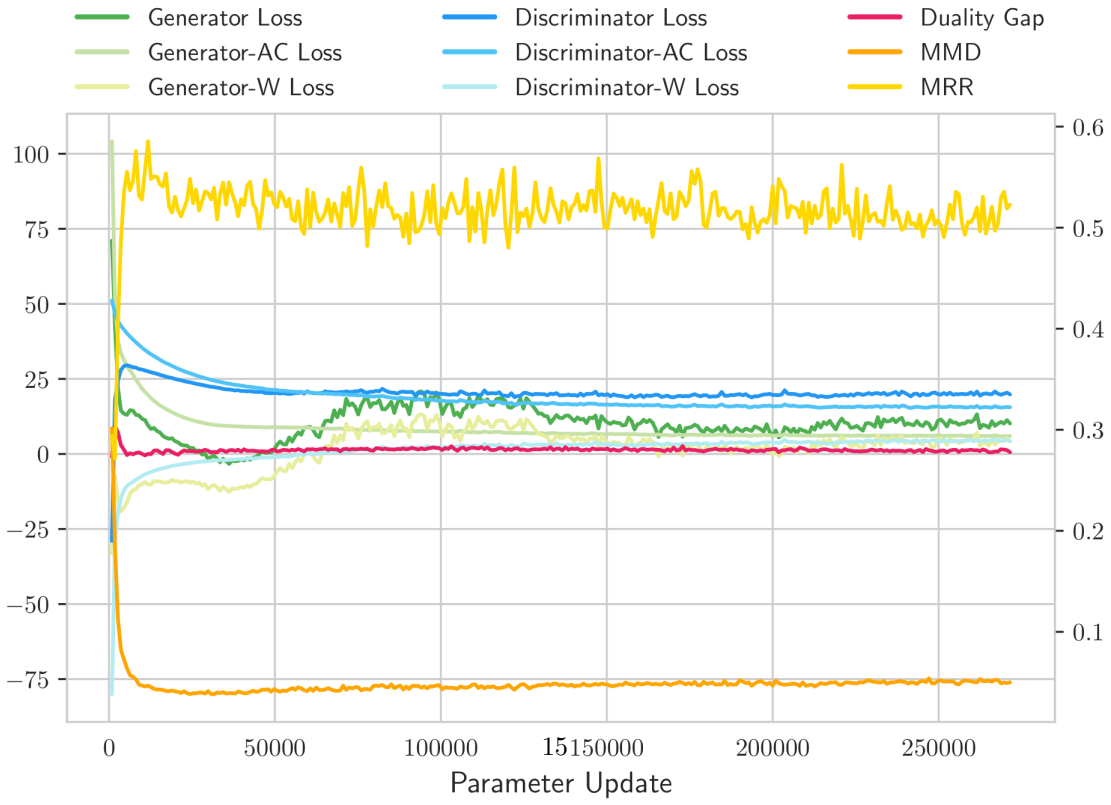


Figure S6: Losses and evaluations at training time. W = Wasserstein, AC = Auxiliary Classifier

Table S9: Top-1 and Top-10 accuracy (%) for the 5 OOD sets, A to E, and for the three models (ProteoGAN, CVAE and *Non-Hierarchical*). We add a random baseline for comparison, where generated sequences are in fact taken at random from the dataset. The performance is variable between OOD sets and it seems that no model clearly stands out. The standard deviations are not shown because negligible. Best results are in bold.

	ProteoGAN		CVAE		*Non-Hier.*		Random	
	Top1	Top10	Top1	Top10	Top1	Top10	Top1	Top10
A	**64**	**97**	43	87	25	95	4	41
B	18	79	**23**	**80**	2	47	2	34
C	**30**	**89**	19	72	0	1	4	38
D	5	**41**	**7**	**41**	0	5	1	30
E	41	**97**	**42**	**97**	2	24	6	45

1.5.3 Out-of-distribution (OOD) performance on held-out GO label combinations

We investigate whether ProteoGAN and baselines that allow for conditioning on multiple labels (CVAE and *Non-Hierarchical*) are able to generate sequences that present a combination of GO labels unseen during training. To do so, we held out from the training set 5 different sets (called A-E, which will be referred to as 'OOD set(s)') of sequences that present unseen combinations of 2, 3 or 4 GO labels, while sequences with some but not all labels of the held-out GO label combinations were included at training time. The OOD sets A to C combine 2 labels, D combines 4 labels and E combines 3 labels. We compare the generated sequences to real sequences in the test set that either belong to a) sequences in the OOD set, b) sequences that have one of the labels present in the held-out GO label combination (but not all), or c) sequences that have any other combination of labels, excluding labels that belong to the combination of interest. We report Top-X accuracy for $X \in \{1, 10\}$ for each OOD set and each model in Table S9. Top-X accuracy is estimated as follows: among the X nearest neighbours (NN) of each generated sequence, as estimated by euclidean distance in k-mer count embedding space (k=3), a sequence is counted as accurate is any of the NN belongs to the OOD set. The results are averaged over 1000 sequences in each of the generated sets. The OOD sets contain approximately 1000 sequences each and the number of real sequences that are not in the OOD sets is approximately 15000.

It seems that the OOD sets A, C and E are the easiest to generate when held-out while B and D are harder. It is possible that D is the hardest OOD set due to its larger number of components (4 GO labels). The model *Non-Hierachical* performs surprisingly worse than random for the OOD sets C, D and E. In comparison, ProteoGAN and CVAE are often much better than random. This could be due to the fact that accounting for the hierarchical organisation is important.

1.5.4 Additional evaluations

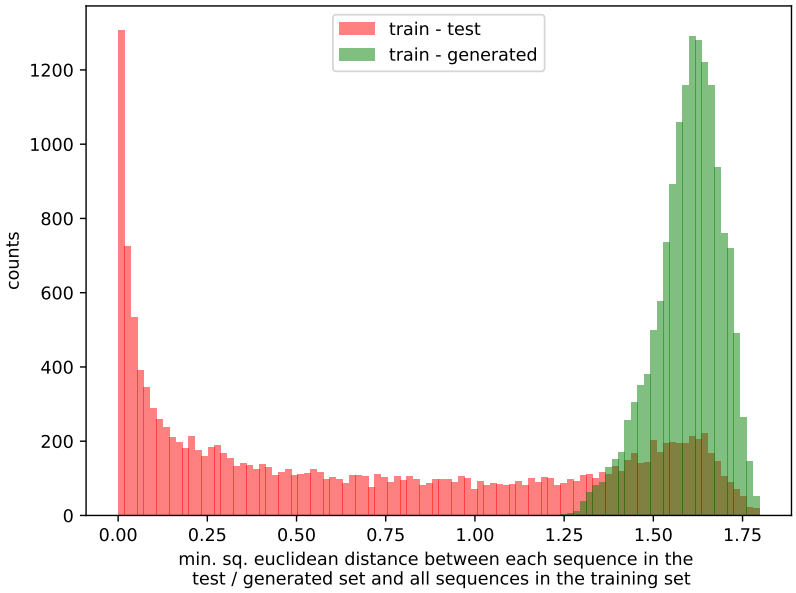
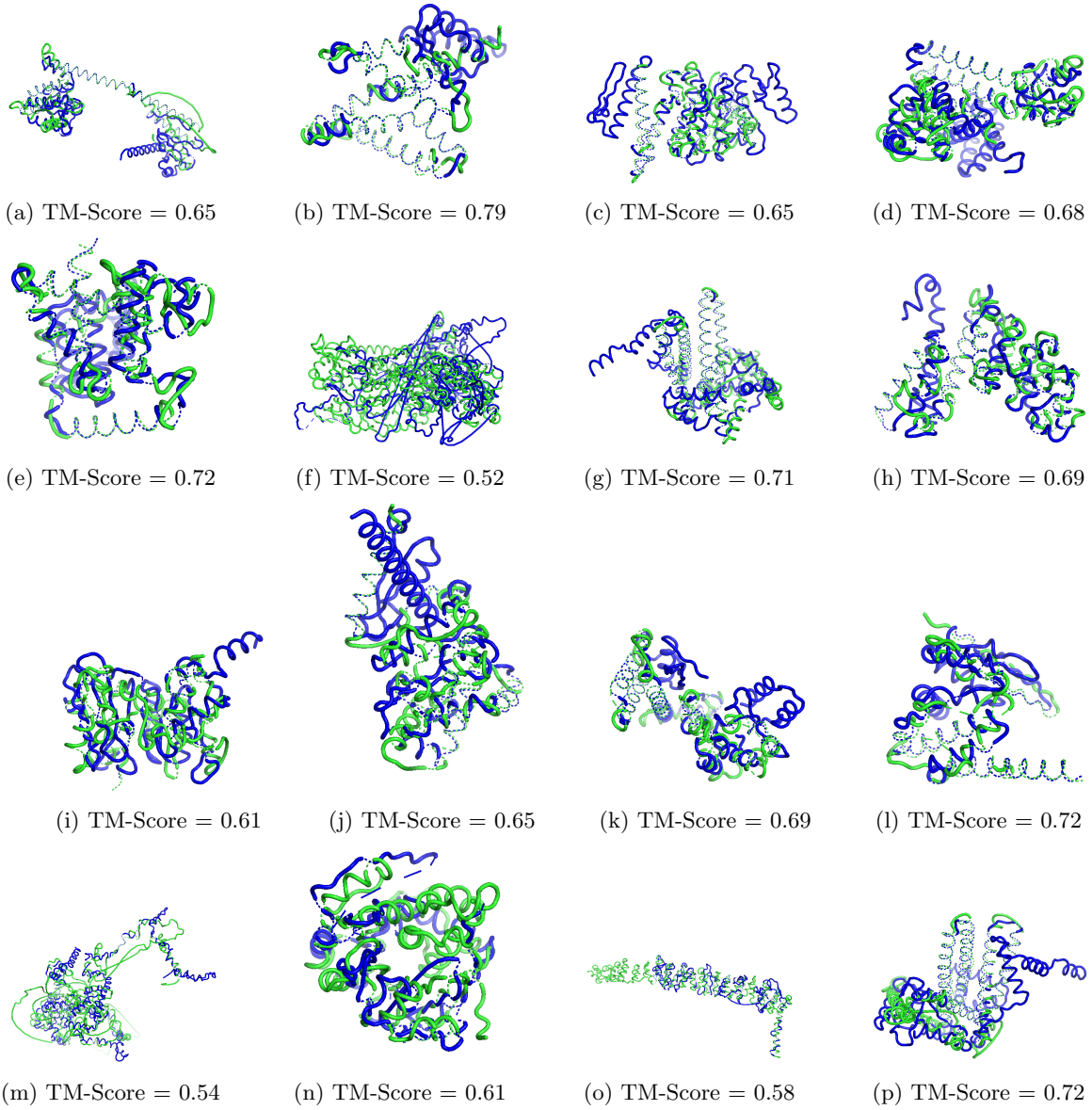


Figure S7: Distributions of pairwise distances in kernel feature space between a testset and the training set (red) and a generated set of ProteoGAN and the training set (green). It can be seen that the generated sequences are not closer to the training set than the testset (which would indicate overfitting). Further the generated sequences are about as far, but not further, away from the training set than the testset.



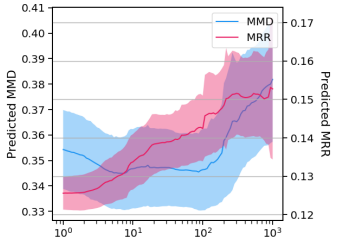
(a) TM-Score = 0.65 (b) TM-Score = 0.79 (c) TM-Score = 0.65 (d) TM-Score = 0.68

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

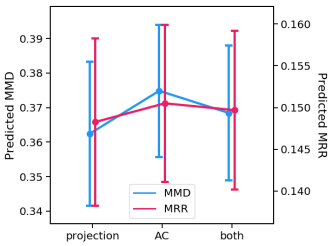
(i) TM-Score = 0.61 (j) TM-Score = 0.65 (k) TM-Score = 0.69 (l) TM-Score = 0.72

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

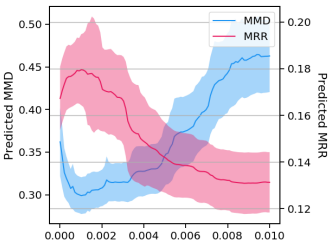
Figure S8: Some exemplary ProteoGAN generated structures (blue) in comparison to their structurally closest homolog in PDB (green).



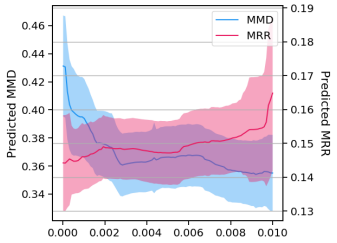
(a) Auxiliary Classifier weighing factor



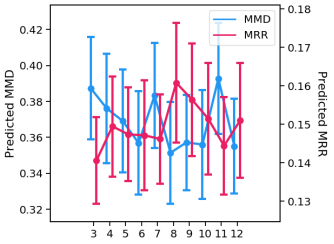
(b) Conditioning mechanism



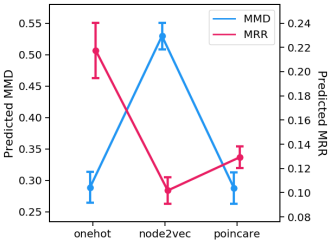
(c) Discriminator learning rate



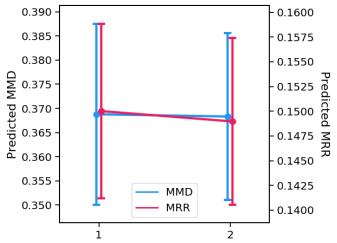
(d) Generator learning rate



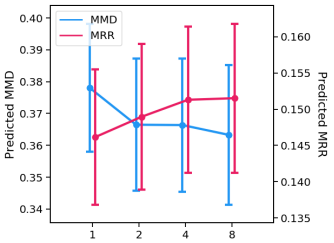
(e) Kernel size



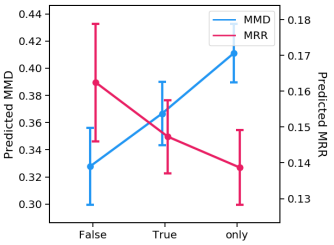
(f) Label embedding



(g) Projections

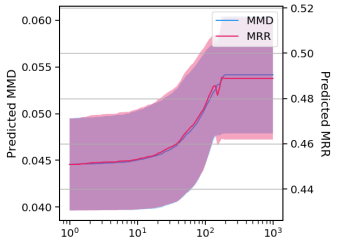


(h) Strides

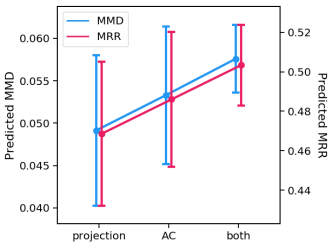


(i) Use chemophysical features

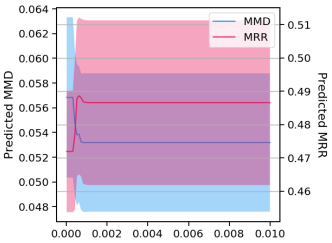
Figure S9: Marginal predictions of hyperparameters based on optimization data in the second optimization. Predictions were obtained training on MMD and MRR. Note that for MMD, lower is better.



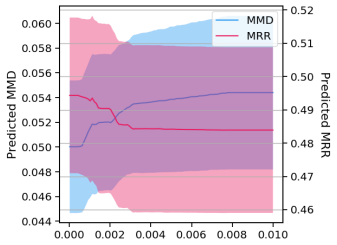
(a) Auxiliary Classifier weighing factor



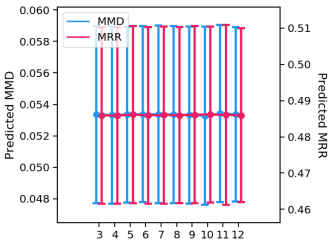
(b) Conditioning mechanism



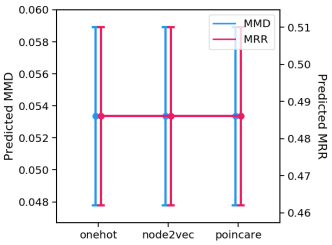
(c) Discriminator learning rate



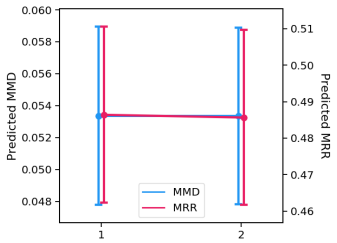
(d) Generator learning rate



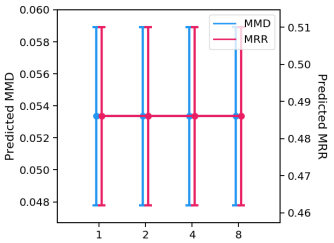
(e) Kernel size



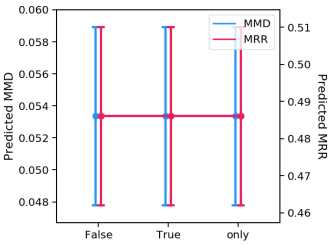
(f) Label embedding



(g) Projections



(h) Strides



(i) Use chemophysical features

Figure S10: 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.

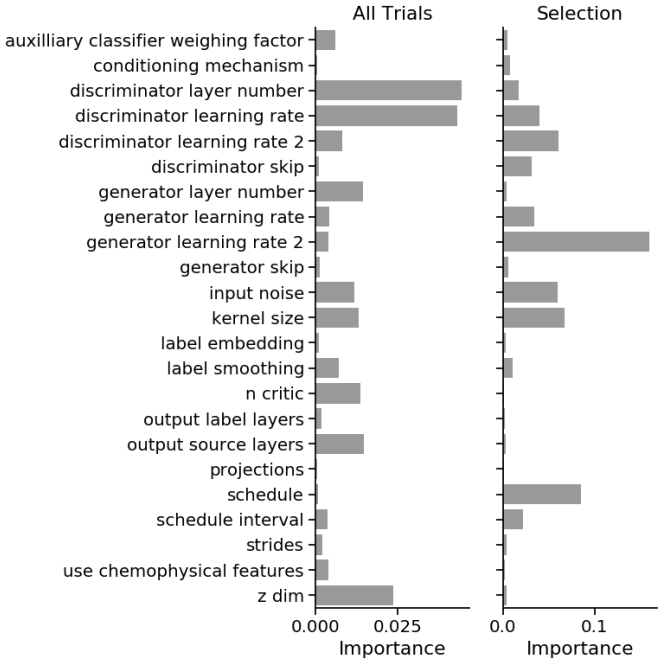


Figure S11: Hyperparameter importance for the first BOHB optimization. Shown are all hyperparameters subject to optimization for all models (left), and a manual selection of models that was trained for 100 epochs (right).

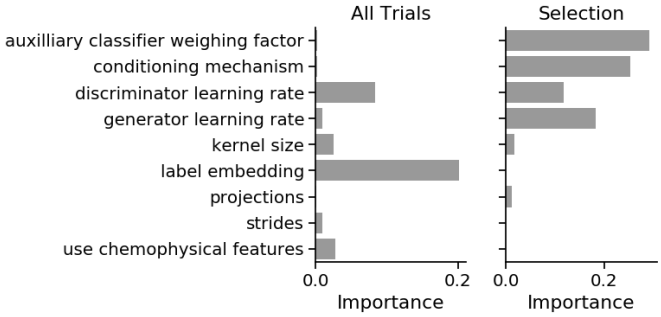
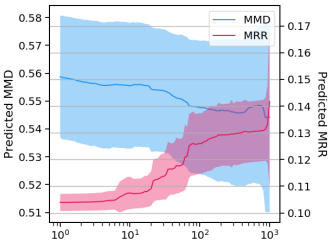
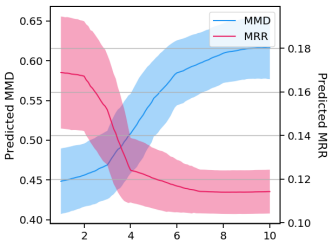


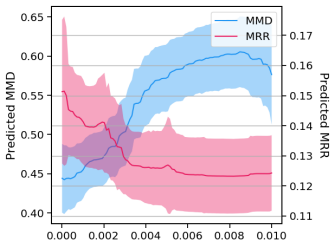
Figure S12: Hyperparameter importance for the second BOHB optimization. The bars show individual importance of each hyperparameter in terms of the variance they explain. We conducted the analysis for all trials of the optimization (left) and for the selected models that were trained for prolonged time (100 epochs) (right). The total variance explained by the main effects was 36% and 88%, respectively.



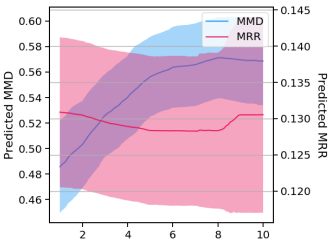
(a) Auxiliary Classifier weighing factor



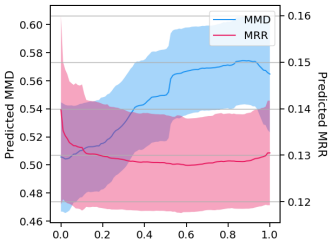
(b) Discriminator layer number



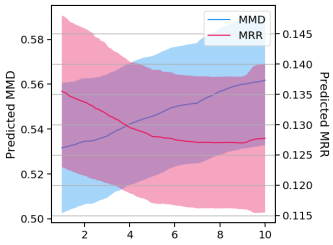
(c) Discriminator learning rate



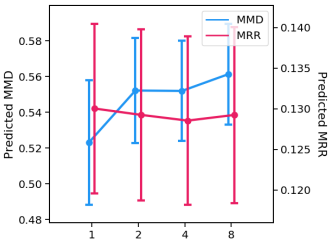
(d) Generator layer number



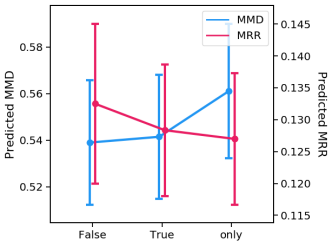
(e) Input noise



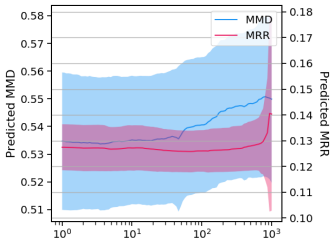
(f) Training ratio



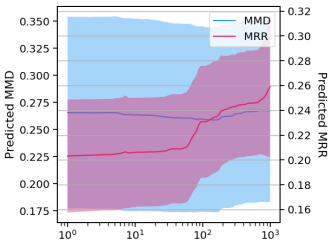
(g) Strides



(h) Use chemophysical features



(i) Latent dimensionality



(j) Auxiliary Classifier weighing factor (Selected)

Figure S13: 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.

References

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

[2] Borgwardt, K. M., Gretton, A., Rasch, M. J., Kriegel, H.-P., Schölkopf, B., and Smola, A. J. (2006). Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57.

[3] Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. (2016). Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*.

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

[5] Das, P., Wadhawan, K., Chang, O., Sercu, T., Santos, C. D., Riemer, M., Chenthamarakshan, V., Padhi, I., and Mojsilovic, A. (2018). Pepcvae: Semi-supervised targeted design of antimicrobial peptide sequences. *arXiv preprint arXiv:1810.07743*.

[6] Davidsen, K., Olson, B. J., DeWitt III, W. S., Feng, J., Harkins, E., Bradley, P., and Matsen IV, F. A. (2019). Deep generative models for t cell receptor protein sequences. *Elife*, 8:e46935.

[7] Falkner, S., Klein, A., and Hutter, F. (2018). Bohb: Robust and efficient hyperparameter optimization at scale. In *ICML*.

[8] Greener, J. G., Moffat, L., and Jones, D. T. (2018). Design of metalloproteins and novel protein folds using variational autoencoders. *Scientific reports*, 8(1):1–12.

[9] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777.

[Higgins et al.] Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. *International Conference on Learning Representations*.

[11] Hu, Z., Yang, Z., Liang, X., Salakhutdinov, R., and Xing, E. P. (2017). Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596.

[12] Hutter, F., Hoos, H., and Leyton-Brown, K. (2014). An efficient approach for assessing hyperparameter importance. In *International conference on machine learning*, pages 754–762. PMLR.

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

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

[15] Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816.

[16] Markiewicz, P., Kleina, L. G., Cruz, C., Ehret, S., and Miller, J. H. (1994). Genetic studies of the Lac repressor. XIV. Analysis of 4000 altered Escherichia coli Lac repressors reveals essential and non-essential residues, as well as" spacers" which do not require a specific sequence. *Journal of molecular biology*, 240(5):421–433.

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

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

[19] Ng, P. C. and Henikoff, S. (2001). Predicting deleterious amino acid substitutions. *Genome research*, 11(5):863–874.

[20] Odena, A., Olah, C., and Shlens, J. (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.

[21] Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434.

[22] Repecka, D., Jauniskis, V., Karpus, L., Rembeza, E., Zrimec, J., Poviloniene, S., Rokaitis, I., Laurynenas, A., Abuajwa, W., Savolainen, O., et al. (2019). Expanding functional protein sequence space using generative adversarial networks. *bioRxiv*, page 789719.

[23] Samish, I. (2017). *Computational protein design*. Springer.