Generative design for gene therapy: An *in vivo* validated method

Farhan Damani, David Brookes, Jeffrey Chan, Rishi Jajoo, Alexander Mijalis, Joyce Samson, Flaviu Vadan, Cameron Webster, Stephen Malina, Sam Sinai
Dyno Therapeutics
343 Arsenal Street, Suite 101
Watertown, MA, US
farhan.damani@dynotx.com, sam.sinai@dynotx.com

Abstract

Machine learning-assisted biological sequence design is a topic of intense interest due to its potential impact on healthcare and biotechnology. In recent years many new approaches have been proposed for sequence design through learning from data alone (rather than mechanistic or structural approaches). These black-box approaches roughly fall into two camps: (i) optimization against a learned oracle (ii) sampling designs from a generative model. While both approaches have demonstrated promise, real-world evidence of their effectiveness is limited, whether used alone or in combination. Here we develop a robust generative model named `VAEProp` and use it to optimize Adeno-associated virus (AAV) capsids, a fundamental gene therapy vector. We show that our method outperforms algorithmic baselines on this design task in the real world. Critically, we demonstrate that our approach is capable of generating vector designs with field-leading therapeutics potential through in-vitro and non-human primate validation experiments.

1 Introduction

Machine learning presents high potential in improving the safety and efficacy of gene therapy through the engineering of delivery vectors with improved properties. In recent years ML-guided sequence design has shown promise in designing proteins such as antibodies, enzymes, and Adeno-associated virus (AAV) capsids (Bryant et al., 2021; Ogden et al., 2019; Biswas et al., 2021; Madani et al., 2023). These methods draw from two philosophical paradigms (though increasingly a combination, as herein). Search-based methods use a learned sequence-to-function oracle to guide search directly in (discrete) sequence space. These methods benefit from explicitly maximizing the desired property but are susceptible to adversarial optimization. This can happen due to model misspecification or epistemic uncertainty affecting oracles, which poses risk when the validation step is expensive (Brookes et al., 2019b). Generative methods sidestep optimization by sampling sequences directly from a model. These methods may generate realistic proteins with an above-baseline functionality but are limited in their ability to optimize sequences with higher quality than those in the training set (Sinai et al., 2021; Riesselman et al., 2018). With notable

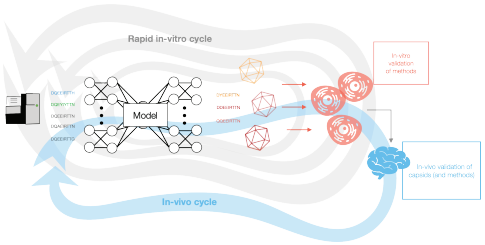

Figure 1: **An experimental ML-in-the-loop platform.** We validate sequence design methods through a rapid in-vitro prototyping of methods in cell culture, before final design occurs in-vivo.

NeurIPS 2023 Generative AI and Biology Workshop.

exceptions (Madani et al., 2023; Bryant et al., 2021), there has been limited validation of both types of methods in the real world, with most previous work using *in silico* proxies for evaluation. Here we introduce `VAEProp`, which achieves the benefit of both design paradigms by performing constrained search in the latent space of a supervised VAE. `VAEProp` controls the explore-exploit trade-off with an explicit parameter, enabling human designers to control their risk. We show that `VAEProp` outperforms both a generative and a search-based baseline at designing AAV capsid sequences for higher infectivity *in vitro* and that the risk parameter effectively trades off higher tail performance for lower average performance. We also show that our method can design sequences on a valuable *in vivo* task, highly relevant to clinical applications, by crossing the blood-brain barrier and efficiently transducing cells (expressing their genetic payload) in the primate brain.

Our approach demonstrates the power of a coupled experimental and in silico platform and illustrates our improved ability to engineer AAV capsids in NHPs and eventual translation to human medicines.

2 Methods

Offline Model Based Optimization. The objective in a data-driven design problem is to identify an optimal input $\boldsymbol{x}^*$ of some ground-truth function, $f(\boldsymbol{x})$, that encodes a scalar property of the input. This can be expressed as

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x})$$

, where $\mathcal{X}$ is a bounded set that we refer to as the "input space" of the problem. The input space may be a discrete or continuous space, and $f(\boldsymbol{x})$ is typically assumed to be a black box from which we can only make zeroth-order evaluations. We assume the availability of a static dataset, $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, consisting of elements of the input space paired with corresponding noise-corrupted evaluations of the ground truth function.

Latent Space Optimization. Because $\mathcal{X}$ is typically high-dimensional and discrete, we train a joint VAE-regression model similar to Gómez-Bombarelli et al. (2018) to map discrete data points $\boldsymbol{x}$ to low-dimensional continuous points $\boldsymbol{z}$, where the loss is a combination of the evidence lower bound (ELBO), for the observed data x and the mean-squared error (MSE) between the true output y and the predicted output from the regression model $f_\theta(\boldsymbol{z})$. The loss function for this model is given by:

$$\mathcal{L}(\boldsymbol{x}, y) = \text{ELBO}(\boldsymbol{x}) + \text{MSE}(y, f_\theta(\boldsymbol{z}))$$

, where the ELBO for the observed data $\boldsymbol{x}$ is:

$$\text{ELBO}(x) = \mathbb{E}_{z \sim q_\phi(z|x)} \left[\log p_\theta(x|z)\right] - \text{D}_{KL}\left(q_\phi(z|x) \| p(z)\right)$$

The term inside the expectation represents the log likelihood of observing $\boldsymbol{x}$ given the latent variable $\boldsymbol{z}$. The second term is the Kullback-Leibler divergence between the approximate posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and the prior $p(\boldsymbol{z})$. MSE is the mean squared error between the true label y and the predicted output from the regression model, $f_\theta(\boldsymbol{z})$:

$$\text{MSE}(y, f(z)) = \mathbb{E}_{z \sim q_\phi(z|x)} \left[(y_i - f(z))^2)\right]$$

The MSE term can be approximated with Monte Carlo sampling, but for simplicity, we use the posterior mean $\boldsymbol{z}_\mu$. The training objective is to minimize $\mathcal{L}$ with respect to the model parameters by adjusting both the parameters of the regression model and the variational approximation to the posterior.

After model training, the model can be used for optimization by computing gradients of $f(\boldsymbol{z})$ with respect to $\boldsymbol{z}$ holding the regression model parameters constant $\nabla_{\mathbf{z}|\theta} f(\mathbf{z})$. The key challenge is that this search procedure is likely to find points with high-error (Brookes et al., 2019a), which are regimes of the latent space that do not have data support. Below, we describe a simple, scalable approach to constraining latent space optimization to regimes with training data support.

Approximating the Aggregate Posterior. We constrain search to regions with training data support by using the aggregate posterior, which has been shown to be a more robust quantity for estimating the density over latent data points (Tomczak & Welling, 2018; Hoffman & Johnson, 2016). The aggregate posterior is an expectation of the approximate posterior, marginalized over the data:

$$q(\boldsymbol{z}) = \mathbb{E}_{p(\boldsymbol{x})} \left[q(\boldsymbol{z}|\boldsymbol{x})\right]$$

When there is a mismatch between the prior and the aggregate posterior, the performance of the VAE can suffer (Tomczak & Welling, 2018; Hoffman & Johnson, 2016). This mismatch can lead to situations where the decoder might not have encountered samples from certain regions of the prior distribution, especially those where the aggregate posterior allocates minimal probability density. Effectively, the support of the optimal decoder, with respect to the latent vector, tends to be confined within the support of the marginal posterior. This phenomenon is characterized as the "posterior holes" problem in Rezende & Viola (2018).

While previous work addressed this issue by minimizing the gap between the prior and the posterior (thereby lowering the gap between the prior and the aggregate posterior) through more powerful data-driven priors (Tomczak & Welling, 2018), we do not attempt to fix this mismatch but instead use the aggregate posterior directly to constrain search to regions with training data support given a fixed VAE.

While ideally we would estimate $q(\boldsymbol{z})$ exactly, this is computationally expensive as it requires a sum over all training points:

$$q_\lambda(\boldsymbol{z}) = \frac{1}{N}\sum_{i=1}^{N} q_\phi(\boldsymbol{z}|\boldsymbol{x}_i)$$

Therefore, we approximate the aggregate posterior $q(\boldsymbol{z})$ by embedding our data points using the encoder e, which gives you the posterior means $\{\boldsymbol{z}_{\mu,i}\}_{i=1}^{N}$ for the training data, and then we fit a density model to this batch of data points. Because $\boldsymbol{z}_\mu$ is low-dimensional and continuous, we use a Gaussian Mixture Model (GMM) to approximate $q(\boldsymbol{z})$ as $\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{z}|\mu_k, \Sigma_k)$ where π_k is the mixing coefficient for the kth component. In the next section we describe how the aggregate posterior can be used as a tool for constraining search.

Constraining Search. To avoid latent space optimization from veering off into regions with high prediction error, we use the aggregate posterior to constrain search to reliable regions. Previous work has used spherical boundary constraint (Gómez-Bombarelli et al., 2018) and decoder uncertainty (Notin et al., 2021) to constrain this search. Our approach benefits from simplicity (Scikit-learn's (Pedregosa et al., 2011) GMM tool) and can be estimated quickly as it does not require any additional forward passes through the VAE decoder.

Similar to Notin et al. (2021), we pre-specify an uncertainty threshold T as a stopping criterion for optimization. We use the 90th, 95th, and 99th percentile negative log-likelihood scores of the training data for low, medium, and high-risk optimization buckets.

Algorithm 1 Constrained Latent Space Optimization Using The Aggregate Posterior

Input: Log-likelihood threshold T, total gradient updates N, gradient scaling factor α
Compute posterior means, $\boldsymbol{z}_{\mu,i}$, for training data $\{\mathbf{x}_i\}_{i=1}^{N}$ using encoder e
Approximate the aggregate posterior, $q(\mathbf{z})$, by fitting a Gaussian Mixture Model (GMM) to $\{\mathbf{z}_{\mu,i}\}_{i=1}^{N}$
Initialize optimization: encode the initial sequence $\boldsymbol{x}$ to $\boldsymbol{z}$ using encoder e
for $i = 1$ to N **do**
    if $\log q(\mathbf{z}) \leq T$ **then**
        Update $\mathbf{z}$ with gradient ascent: $\mathbf{z} \leftarrow \mathbf{z} + \alpha \nabla_{\mathbf{z}} f(\mathbf{z})$
    end if
end for

3 Results

We consider the problem of designing AAV variants that maximize transduction (i.e. how well the viruses can deliver genetic material to specific tissues or cell types) – a major bottleneck in translating gene therapies from research to the clinic. A necessary precursor to successfully delivering genetic material is the proper folding of the viral capsid and encapsulation of the virus's genetic material, processes that we collectively refer to as "packaging". Both packaging and transduction can be quantitatively measured experimentally using standard sequencing-based techniques (see Appendix for further detail).

3.1 Experiment Setup

We designed variants of wild-type AAV9 with mutations in variable loop regions of VP3 using three design methods, `VAEProp`, AdaLead (Sinai et al., 2020) (as our regression-based search baseline), and a VAE (Kingma & Welling, 2013) (the generative model baseline without the added optimization component). We use an initial training dataset containing AAV sequence variants associated with packaging and transduction measurements. Further details of the methods can be found in the Appendix.

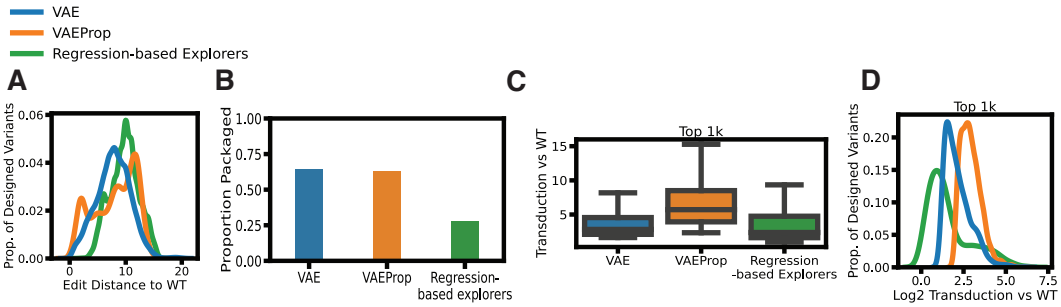

Figure 2: `VAEProp` **achieves high packaging and high transduction.** We apply our design method to optimize in-vitro transduction in cell culture by designing variants of wild-type (WT) AAV9. We compare our method to two baselines (each method received a budget of 8K variants) and report both packaging and transduction measurements. (A) Distribution of edit distances to WT. (B) Bar plot of proportion of designed variants that package (defined as rate of packaging at least $1/32$ of WT). Boxplot (C) and histogram (D) of in-vitro transduction measurements for best 1000 performing designed variants.

3.2 *In vitro* designed AAV capsids

We apply `VAEProp` to the design of AAV sequences in cell culture (HEK293T cells). Our goal was to evaluate the efficacy of our method as well as understand the impact of the number of steps of optimization on design quality. `VAEProp` outperforms the generative and search-based baseline at designing capsids that are both good at packaging and achieve high transduction. `VAEProp` successfully designs capsids at edit distances to wild-type of up to 16 (Figure 2A), of which almost half package. In comparison, only around a quarter of variants designed by the genetic algorithm successfully package. Furthermore, `VAEProp`'s best-performing capsids transduce cells 16-fold better than AAV9 and between 2-4 fold better than the competing baselines (Figure 2C-D), with `VAEProp`'s median sequence transduction exceeding both the VAE and AdaLead's upper quartile sequence transduction (Figure 2D).

As an additional advantage `VAEProp`'s risk parameter enables precise control of the risk-reward trade-off. We generated sequences with `VAEProp` with three different risk settings (low, medium, high). Risk settings correspond to three values for the log-likelihood threshold T. These values were estimated by computing the 90th, 95th, and 99th percentile negative log-likelihood scores of the training data under the $q(\boldsymbol{z})$. This constrains search to points that are *at least* as likely as the least likely sequences in the training data. We compare our results to sampling from a VAE. Relative to the VAE, `VAEProp`'s designed sequences' latent points show a density gradient that shifts away from the training data's region of highest density and towards the region of higher transduction as we increase risk (Figure 3A). As expected, higher risk variants have higher average sequence edit distance and maximum transduction, but at the cost of lower packaging fraction (Figure 3C) and predictability by models.

3.3 *In vivo* designed blood-brain barrier crossing capsid

We apply `VAEProp` to the design of capsids in a NHP study, the gold standard for evaluating clinical translatability. Our design goal is to maximize transduction in the central nervous system (CNS). This task is especially challenging because the blood-brain barrier mediates what molecules can enter the

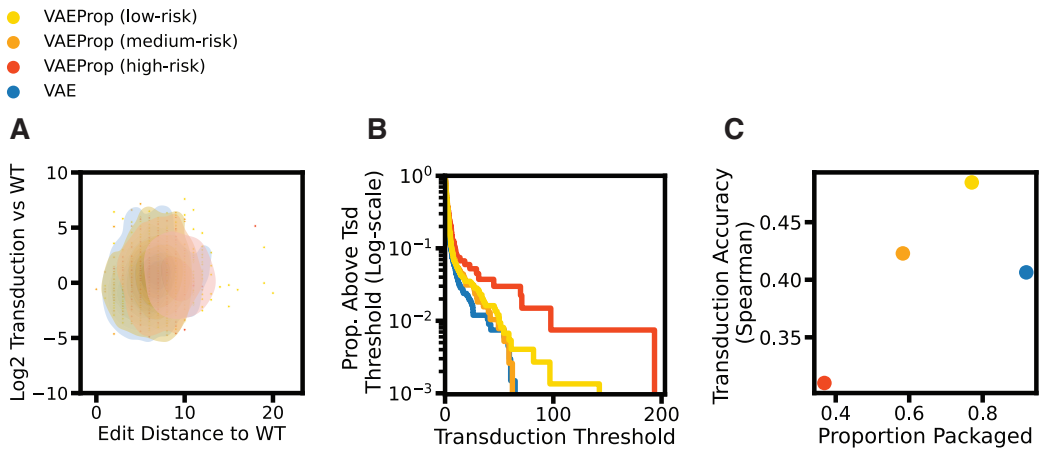

Figure 3: `VAEProp` **can balance risk-reward tradeoff**. We evaluate how the risk parameter translates to different design outcomes. Each method received a budget of 2K variants. (A) Kernel Density Estimate (KDE) plot of in-vitro transduction versus edit distance for VAE and the three risk settings for `VAEProp`. (B) Proportion of designed variants (log-scale) with a transduction measurement above a threshold (x-axis). (C) Regression model accuracy for predicting transduction versus proportion of designed variants that package colored by risk parameter.

brain in the first place, which in practice results in a sparse-reward problem (i.e., most designs have no transduction). We apply our method to two tasks: (1) a high-throughput design (N=5K) where the backbone capsid sequence was our previous best variant of AAV9. The new design was in a region that did not overlap with the positions previously mutated. (2) A low-shot design (N=5) where the backbone capsid sequence was the wild-type AAV9 sequence. Remarkably, in both settings, we manage to design variants that improve over our training set by multiple-folds. The best designs outperform AAV9 (the current standard in clinic) by more than 100-fold in transducing the primate brain.

4 Conclusion

We present `VAEProp`, a method that combines generative modeling and regression-based search. We show that it can design high performing and diverse sequences in the context of AAV design. We demonstrate that `VAEProp` outperforms baseline methods at generating functional and high performing sequences *in vitro* while providing the user more control over the risk-reward trade-off. Notably, we have also validated `VAEProp` in highly relevant *in vivo* setting in non-human primates and shown that it is capable of generating variants the can be appealing for clinical applications.

References

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

Brookes, D. H., Park, H., and Listgarten, J. Conditioning by adaptive sampling for robust design. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 773–782. PMLR, 09–15 Jun 2019a.

Brookes, D. H., Park, H., and Listgarten, J. Conditioning by adaptive sampling for robust design. *arXiv preprint arXiv:1901.10060*, 2019b.

Bryant, D. H., Bashir, A., Sinai, S., Jain, N. K., Ogden, P. J., Riley, P. F., Church, G. M., Colwell, L. J., and Kelsic, E. D. Deep diversification of an AAV capsid protein by machine learning. *Nature Biotechnology*, 39(6):691–696, 2021.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Central Science*, 4(2):268–276, 2018.

Hoffman, M. D. and Johnson, M. J. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *NIPS Workshop on Advances in Approximate Bayesian Inference*, 2016.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Madani, A., Krause, B., Greene, E. R., Subramanian, S., Mohr, B. P., Holton, J. M., Olmos Jr, J. L., Xiong, C., Sun, Z. Z., Socher, R., et al. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, pp. 1–8, 2023.

Notin, P., Hernández-Lobato, J. M., and Gal, Y. Improving black-box optimization in VAE latent space using decoder uncertainty. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=F7LYy9FnK2x`.

Ogden, P. J., Kelsic, E. D., Sinai, S., and Church, G. M. Comprehensive AAV capsid fitness landscape reveals a viral gene and enables machine-guided design. *Science*, 366(6469):1139–1143, 2019.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Rezende, D. J. and Viola, F. Taming vaes. *arXiv preprint arXiv:1810.00597*, 2018.

Riesselman, A. J., Ingraham, J. B., and Marks, D. S. Deep generative models of genetic variation capture the effects of mutations. *Nature methods*, 15(10):816–822, 2018.

Sinai, S., Wang, R., Whatley, A., Slocum, S., Locane, E., and Kelsic, E. D. Adalead: A simple and robust adaptive greedy search algorithm for sequence design. *arXiv preprint arXiv:2010.02141*, 2020.

Sinai, S., Jain, N., Church, G. M., and Kelsic, E. D. Generative aav capsid diversification by latent interpolation. *bioRxiv*, pp. 2021–04, 2021.

Tomczak, J. and Welling, M. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pp. 1214–1223. PMLR, 2018.

A AAV experiment details

Data generation Our data contain AAV variants associated with transduction and packaging measurements. The sequences are variants of the AAV9 wild-type sequence, modified in a 63 amino-acid region containing the VR-IV and VR-V loop of the VP3 capsid protein.

Sequences were assayed for packaging and transduction leveraging a well-established sequencing-based method, as outlined in Ogden et al. (2019). Initially, a "library" of plasmid sequences, which encode the desired protein variants, is constructed. Subsequently, these plasmids are exposed to specific experimental conditions that allow them to transform into proteins and assemble viral capsids, thus forming a sample of viruses that contain the relevant genetic material. This sample is then introduced into cells, enabling the viruses to enter these cells and transfer their genetic material into the cell nuclei through a process known as transduction. Next, the genetic material that has successfully entered the cell nuclei is gathered. At each experimental stage, a small genetic material sample is sequenced using Next-Generation Sequencing methods. This facilitates the estimation of the prevalence of each sequence variant at different stages: within the plasmid library, the virus sample, and the successfully transduced genetic material. These measurements are provided as sequencing counts, i.e., the frequency with which a specific variant appears in the sequencing data. Denote n_i^{plasmid}, n_i^{virus} and n_i^{tsd} as the sequencing counts of the i^{th} variant in the plasmid, viral and transduced samples respectively. The packaging capaability of variant i is quantified as the log rate:

$$y_i^{\text{pkg}} = \log \frac{n_i^{\text{virus}}}{n_i^{\text{plasmid}}} \tag{1}$$

Similarly, the transduction ability of variant i is quantified as

$$y_i^{\text{tsd}} = \log \frac{n_i^{\text{tsd}}}{n_i^{\text{virus}}}. \tag{2}$$

Design strategy We run three sequence design methods given a fixed training dataset of $(\boldsymbol{x}_i, y_i^{\text{pkg}}, y_i^{\text{tsd}})$ triplets, where $\boldsymbol{x}_i$ is the sequence of variant i. The input space for the design is the space of all amino acid sequences. Details of the surrogate models and search method used are discussed below.

Surrogate models We trained two surrogate models: $\hat{f}_{\text{pkg}}(\boldsymbol{x})$ to predict packaging from sequence and $\hat{f}_{\text{tsd}}(\boldsymbol{x})$ to predict transduction. These packaging and transduction surrogate models were trained using the input-label pairs $(\boldsymbol{x}_i, y_i^{\text{pkg}})$ and $(\boldsymbol{x}_i, y_i^{\text{tsd}})$, respectively. Both models had a Convolutional Neural Network (CNN) architecture with following hyperparameters:

- Number of Convolutional Blocks: 2
- Number of Channels: $[32, 32]$
- Pooling Scales: $[0, 0]$
- Number of dense layers: 1
- Dense layer size: 32
- activation type: LeakyReLU

Both models were trained by minimizing an MSE loss using the Adam optimization algorithm until convergence using early stopping on a random-holdout validation set (10% of samples) with a patience set to 10 epochs.

Baselines. For our search-based baseline, we used AdaLead (Sinai et al., 2020), a variant of a genetic algorithm that has been shown to work well for biological sequence design applications, guided by a CNN oracle trained on the full training dataset as the reward function. We refer readers to Sinai et al. (2020) for a detailed treatment of the AdaLead algorithm.

For our generative baseline, we used a VAE with an encoder-decoder architecture parameterized with CNNs. We trained the VAE on a filtered set of variants with a transduction measurement greater than zero (measurements are on a log2 scale and normalized to wild-type), which biases sampling

towards variants with higher transduction. We sample variants using a common 2-step heuristic: (a) embed training data using the encoder e (b) sample from a multivariate Gaussian distribution fit to the embedding points. Designed variants are pre-filtered according to predicted packaging measurements using $\hat{f}_{\text{pkg}}(\boldsymbol{x})$.