REVIEW

PROTEIN ENGINEERING

How artificial intelligence is reengineering protein engineering

Jennifer Listgarten[1,2,3]* and Hanlun Jiang[1]

Over the past decades, protein engineering has matured into a field of its own, driven by computational modeling and high-throughput wet lab experiments, with broad application in therapeutics, diagnostics, agriculture, and manufacturing. In recent years, artificial intelligence (AI) has further propelled protein engineering by enabling more efficient search through high-dimensional sequence space for proteins with desired properties. Notable AI-based advances encompass generative modeling of sequences, backbone structure, and atoms; tailoring general versions of such models to design proteins with specific properties; modeling for extraction of protein representations and scoring candidate protein sequences; and developing techniques for library design, including synthesis-aware approaches. Herein we discuss these advances, emphasizing a unifying view through a statistical interpretation of modern AI approaches.

Given that proteins are key drivers of all forms of life, it is no surprise that humans have long sought to tinker with them. Proteins animate our genomes by catalyzing and orchestrating nearly every biological process from energy metabolism to gene regulation, while also providing cellular structure. We tinker with naturally existing proteins so as to coerce them into new roles, and even create altogether new proteins for bespoke use cases. By engineering proteins, we can create new therapeutics (*1*) and vaccines (*2*); design plants with enhanced resilience to environmental stressors (*3*) and more efficient carbon fixation (*4*); and enable cost-effective biomanufacturing of antibiotics (*5*) and materials (*6*). Few scientific fields have the potential for a broader societal impact than protein engineering.

Whereas natural evolution has slowly crafted the proteins of life over billions of years (*7*), the task of protein engineering is to create proteins with specified properties on a vastly compressed timescale—in years, and with the help of artificial intelligence (AI), maybe in days. Toward this goal, two distinct pre-AI approaches—directed evolution (DE) (*8*–*16*) and computational protein design (CPD) (*17*–*26*)—were developed in parallel and recognized by Nobel Prizes in 2018 and 2024. In DE, the mechanisms of natural evolution are repurposed and turbocharged by iterative rounds of mutation and human-specified selection, guided by wet lab measurements. DE requires an initial protein with a hint of relevant functionality but does not require biophysical modeling. CPD, by contrast, is based entirely on approximate biophysical modeling informed by statistics from the Protein Data Bank (PDB) (*27*); it does not necessarily require a functional initial protein with its complete sequence specified. CPD makes in silico moves in protein sequence and structure space, guided by an approximate physics-based energy function that serves as a universal fitness landscape for all engineering problems (*28*, *29*). This energy function scores the favorability of any three-dimensional (3D) conformation for a protein sequence, enabling effective design of protein structures and even protein-protein interactions. However, it is too coarse to capture complex protein dynamics or quantum-mechanical effects essential for engineering enzyme catalysis (*30*).

Both of these approaches must contend with the daunting fundamental problem of searching through an unimaginably large space of proteins; each protein is a string of amino acids from a 20-letter alphabet, with a typical length in the hundreds (*31*). A small protein with only 100 amino acids can take on already $20^{100} \approx 10^{130}$ possible sequences, far exceeding the estimated number of atoms in the universe of about 10^{80} (*32*). Moreover, only a tiny fraction of these variants can fold and express.

At their core, both DE and CPD function by searching through protein space and scoring variants. To navigate protein space, DE scores with experimental measurements directly relevant to the desired protein properties, albeit at great cost, time, and labor; thus, DE is limited in throughput and confined to exploring sequences similar to that of the initial protein. Meanwhile, CPD can search far beyond known protein space and score quickly and inexpensively in comparison to DE; but it relies on a single energy function regardless of the protein property being engineered, which, as mentioned above, is likely inadequate for design tasks dependent on protein dynamics and/or the nuances of catalysis. These two approaches thus complement each other and are often used together, such as using DE to refine CPD candidates (*33*). AI has infused both of them, making their distinction increasingly blurred.

What is the hope that AI brings to the field of protein engineering?

At a conceptual level, the hope is to improve both searching and scoring. That is, to (i) more efficiently search through the enormous space of protein sequences—making large, smart jumps, rather than small, random perturbations, to find sequences unreachable by traditional methods—and (ii) more quickly and cost-effectively predict protein properties of interest (e.g., expression, stability, and activity) with useful levels of accuracy for the problem on hand, which in turn can inform more efficient search. Note that the terms "property prediction" and "fitness prediction" are often used interchangeably in protein engineering, despite their different but related meanings.

Herein, we walk through these topics, starting from the most intuitive view of a simple in silico search on an in silico fitness function, then bridging from there to generative models to perform AI-informed search, and later to conditional generative models. As we shall see, a given approach can be viewed through several of these lenses. Having set this conceptual groundwork, we then take a more in-depth tour of how AI is currently used in protein engineering for generating variants, but also for variant scoring, representation learning, and library design. Owing to space and citation constraints, we were unable to mention many other important topics of AI-based protein engineering, including model parameter scaling and agent-based engineering, as well as legal and ethical issues.

Navigating the search space with in silico property predictors

Intuitively, how might AI help us to move more efficiently through protein sequence space? Suppose we had some in silico estimate of a fitness function on sequences, s, for fitness, y (e.g., catalytic efficiency) given by $y = f(s)$. And suppose we seek a sequence with high fitness—that is, to find s such that $y = f(s) > \tau$ for a given threshold, τ (see Table 1 for summary of notation used). This fitness function could arise from a biophysical model or a machine learning–based one. The simplest search (i.e., optimization) algorithm would be to start with some sequence, then make random, uninformed mutations in sequence space, evaluate $f(s)$, select the fittest mutants, and repeat the procedure, much like in DE. Indeed, this is a so-called in silico evolutionary algorithm (EA) for optimization of a function. Notably, this method for proposing mutations is oblivious to $f(s)$, knowledge of which could help to make smarter moves. However, in a modern

[1]Department of Electrical Engineering and Computer Science, UC Berkeley, Berkeley, CA, USA. [2]Center for Computational Biology, University of California Berkeley, CA, USA. [3]The UC Berkeley–UCSF Graduate Program in Bioengineering, University of California, San Francisco, San Francisco, CA, USA. *Corresponding author. Email: jennl@berkeley.edu

Table 1. Summary of notation used.

Symbol	Meaning
$\boldsymbol{s}$	A protein sequence (string of amino acids)
$\boldsymbol{b}$	A protein backbone structure (3D coordinates of backbone atoms)
$\boldsymbol{x}$	A general object (sequence, structure, or atomic configuration)
$\boldsymbol{y}$	A protein property (e.g., stability, catalytic efficiency)
$\boldsymbol{Y}$	A set of desired property values (e.g., high stability, high activity), possibly for several properties
τ	A threshold value for a property y
$\boldsymbol{f(s)}$	A fitness function
$\boldsymbol{f_\theta(s)}$	A learned fitness or property predictor with parameters θ
$\boldsymbol{p(s)}$	A probability distribution over protein sequences
$\boldsymbol{p(b)}$	A probability distribution over protein backbone structures
$\boldsymbol{p(s, b)}$	Joint probability of sequence and structure
$\boldsymbol{p(y \mid s)}$	Probability of property y for a given sequence s
$\boldsymbol{p(s \mid y \in Y)}$	Probability of sequence s having a property value $y \in Y$
$\boldsymbol{s \sim p(s)}$	A sequence, s, is sampled (generated) from distribution $p(s)$
$\boldsymbol{s \sim p_\theta(s)}$	A sequence, s, is sampled (generated) from a generative model with parameters θ
$\boldsymbol{\nabla_x log\, p(x)}$	Gradient of log $p(x)$ with respect to x

AI-informed incarnation of EAs (*34*–*36*), mutated sequences are instead proposed by a generative model on sequences [i.e., a probability density model over sequence space such as a variational autoencoder (VAE) (*37*)], $s \sim p_{\phi^t}(s)$, whose parameters, ϕ^t, get updated upon each search iteration, t. At each iteration, we retrain the generative model on the sampled sequences weighted by their fitness score. Initially, mutations drawn from this model are oblivious to $f(s)$ as in DE and EAs. However, as iterations progress, the updated generative model gets to learn about (i) what sequences score better under $f(s)$, and consequently, (ii) how to more smartly move through the space, by, for example, making multiple mutations that act in a coordinated manner [i.e., according to epistasis in $f(s)$]. Thus, mutations drawn from this model become progressively fitter as the generative model in essence slowly sniffs out the fitness landscape geometry and uses it to its advantage. Upon termination, one obtains desired protein sequences by sampling from the generative model at the final iteration, T, that is, $s \sim p_{\phi^T}(s)$. It is interesting to note that in progressing from naïve search to AI-informed search, we have replaced explicitly searching for s with searching for the parameters of a generative model that captures the probability distribution of our desired s, which brings us into the language of probability, and opens up a world of generative models.

When the aforementioned in silico fitness function comes from a supervised machine learning model, $y = f_\theta(s)$, with parameters θ learned from data, it is not generally accurate in parts of sequence space dissimilar to the training data. Consequently, it is wise to rein in the search [i.e., control updating of $p_{\phi^t}(s)$] by making use of prior, auxiliary knowledge, such as some notion of how "protein-like" a sequence is, how stable a protein is, or which sequences the predictive model, $f_\theta(s)$, was trained on. If this auxiliary knowledge is encoded in a probability distribution, $p_0(s)$—for example, having trained $p_0(s)$ on a large swath of the protein space, such as in UniRef (*38*)—then one can modulate the AI-informed search to pay heed to the prior information encoded in $p_0(s)$. This approach, known as conditioning by adaptive sampling (CbAS), balances pushing the optimization of $p_{\phi^t}(s)$ to find high-scoring variants according to $f_\theta(s)$, with focusing on parts of the space favored by $p_0(s)$ (*35*, *39*). Again, we obtain desired protein sequences by sampling from the final generative model, $s \sim p_{\phi^T}(s)$.

When the in silico fitness function has a probabilistic interpretation—namely, $f_\theta(s) = p_\theta(y \in Y \mid s)$, as is common for many classes of supervised machine learning models—then CbAS becomes a statistical procedure for combining two models: (i) an unconditional generative model that encodes prior information, $p_0(s)$, and (ii) a property predictive model, $p_\theta(y \mid s)$. In fact, CbAS is executing Bayes' rule to obtain the conditional distribution, $p_{\phi^T}(s) = p(s \mid y \in Y)$, where $y \in Y$ means that the properties satisfy a desired condition(s), such as that an enzyme has catalytic efficiency for a particular reaction higher than some target value. This property-conditioned distribution can then be sampled for desired proteins.

Fundamentally, the goal of AI-based protein engineering is to precisely estimate and then sample from a property-conditional probability distribution over sequences—that is, we seek to sample sequences, s, from a distribution conditioned on any number of design criteria encapsulated in Y, formally, $s \sim p(s \mid y \in Y)$ (Fig. 1).

Three ways to obtain a conditional generative model

So far we have discussed how to use auxiliary knowledge encoded in a model to rein in an AI-informed search of an in silico fitness function, which was one narrative path to conditional generative models. It may also be useful to think of conditional generative modeling for protein engineering as starting with a general model of some kind [e.g., a "pan-protein" model trained on UniRef (*38*)] and then conditioning it on properties of interest for specific engineering goals. In the previous section, we briefly described one way to obtain a conditional generative model, namely, by way of CbAS. More generally, there are three primary strategies to obtain a conditional generative model, which we describe next.

In perhaps the most straightforward approach, we decide upfront which types of properties we want to condition on (e.g., enzyme commission number, secondary structure, binding to a particular ligand), and directly "bake in" these conditioning variables, $Y_1, \ldots, Y_K$ to the generative model during training—that is, we directly estimate $p_\theta(s \mid y_1 \in Y_1, \ldots, y_k \in Y_k)$. The downside of such an approach is that we must decide upfront which properties we want to condition on, and, moreover, we must have sufficient supervised data for those properties at the time of training this model. Consequently, whenever we want to include a new conditioning variable, or use newly available supervised data, we need to retrain a potentially large model.

When we do not "bake in" the conditioning from the outset, we are likely combining an existing unconditional model, $p_\phi(s)$, with a supervised property predictive model, $p_\theta(y \mid s)$, to obtain the desired conditional distribution, such as in CbAS. Any time we correctly combine these two models to obtain the desired protein engineering conditional distribution, $p(s \mid y)$, we are, one way or another, applying the 18th-century Bayes' rule, $p(s \mid y) = \frac{p(y \mid s)p(s)}{p(y)} = \frac{p(y \mid s)p(s)}{\sum_s p(y \mid s)p(s)}$. One advantage of this rule is that it gives us a "plug-and-play" ability to take expertly engineered, general, pan-protein models trained on massive amounts of data, and to tailor their usage to our needs by focusing them on properties of interest at any point in the future. However, one catch is that the denominator of Bayes' rule is generally computationally intractable owing to the summation over all possible protein sequences, s, thus generally requiring some kind of approximation such as variational inference, or unwieldy sampling (*40*). Moreover, variational inference for executing Bayes' rule often requires iteratively training a new generative model over sequences, such as $p_\phi^T(s)$ in CbAS, which can be time-consuming but still a useful approach for our toolkit. The related approach of direct preference optimization (DPO) also iteratively trains a new generative model, modulating the prior information with a preference ranking model in a manner nearly resembling Bayes' rule (*41*–*43*). Is it possible to side-step both the Bayes' rule denominator and having to train a new generative model, while still obtaining samples $s \sim p(s \mid y \in Y)$? It turns out, yes, as we discuss next.

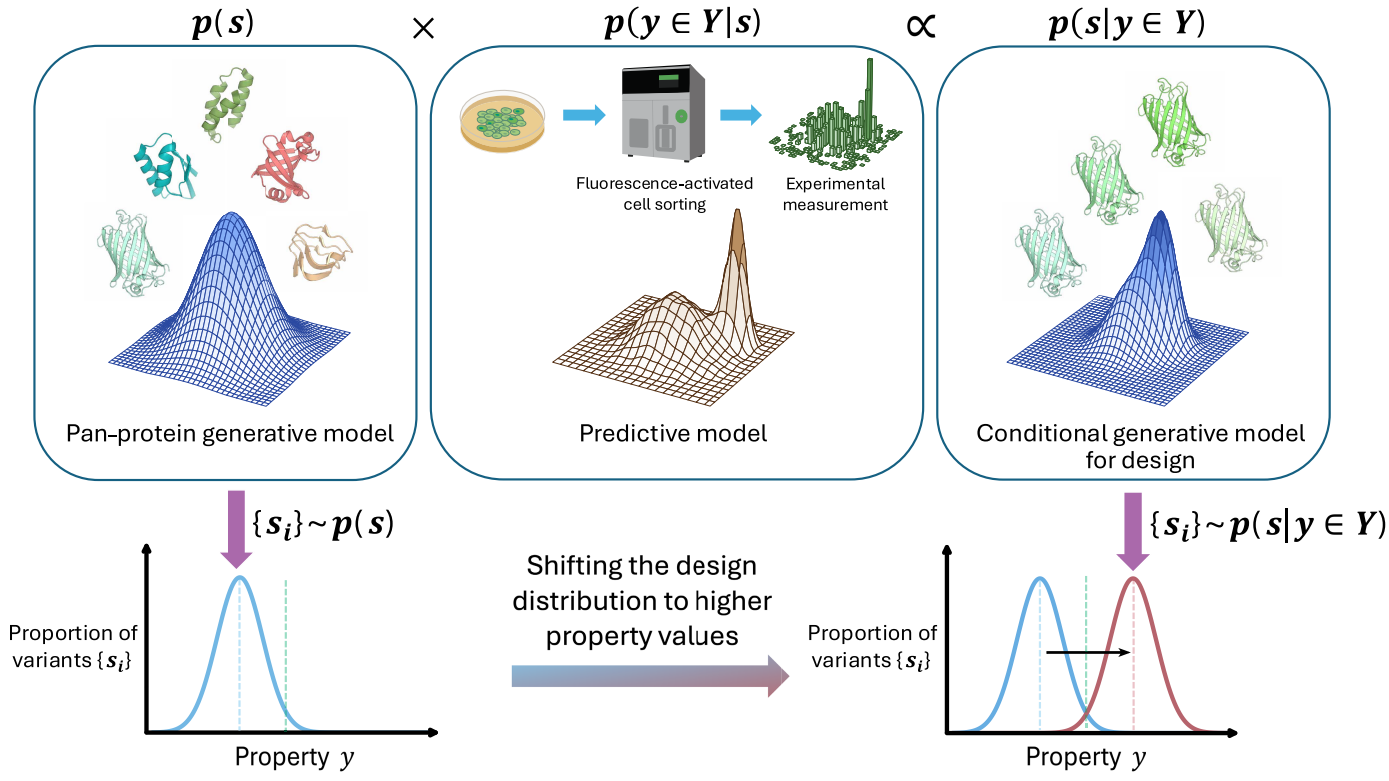

Fig. 1. AI-based protein engineering viewed through a statistical lens. Designing sequences is tantamount to sampling sequences, s, from the desired conditional distribution, $p(s \mid y \in Y)$. The set $\{s_i\}$ denotes a set of sequences, indexed by i, which were sampled repeatedly from a generative model of sequences. The set Y, for example, could be those fluorescence values, y, higher than some threshold, τ, denoted by the green dashed line. One can obtain the design distribution by directly estimating the desired conditional distribution from suitable data; or, one might adapt an existing, more general pan-protein model by modulating it with, say, experimental data. The latter strategy is depicted in the figure, namely, background knowledge encoded as a pretrained generative model, $p(s)$, is combined with experimentally acquired data from which a predictive model, $p(y \in Y \mid s)$, is built. The combination is performed with Bayes' rule, $p(s \mid y \in Y) \propto p(s)p(y \in Y \mid s)$, where $\propto$ denotes proportionality. Icons of Petri dish and flow cytometer were adapted and used under Creative Commons license (CC0) from BioIcons.com.

In a third strategy, which we refer to as "on-the-fly" conditioning, we do not need to train a new generative model. Rather, we modulate, or "guide" the sampling process of an unconditional model with information from the property predictive model, so as to effectively sample from the desired conditional distribution, $s \sim p(s \mid y \in Y)$, without explicitly training or representing this distribution. However, to do so, we must incur a relatively costly, iterative sampling algorithm. A common class of generative models where this on-the-fly strategy is used is diffusion and related model classes [i.e., flow matching, score-matching, and stochastic interpolants (*44*–*49*)]. At their core, these models seek to estimate the gradient of the log density of x, that is, $\nabla_x \log p(x)$ (*50*), rather than taking the classical strategy of estimating the density itself, $p(x)$, for some general object, x. Consequently, sampling $x \sim p(x)$ in such models requires iteratively following the gradient estimate from the model (technically, to numerically integrate a differential equation). However, what we get in return for this effort is that, when combining these models with property predictors through Bayes' rule, taking the gradient gets rid of the computationally intractable denominator (because it does not depend on x). Specifically, the iterative sampling algorithm in time, t, used for these models, blends information from two sources: $x^{t+1} = x^t + \nabla_x \log p(x^t) + \nabla_x \log p(y \in Y | x^t)$, where the first gradient is from the already-trained unconditional model, and the second—the modulation of the unconditional gradient—is "guidance" provided by a property predictive model (*48*, *49*, *51*).

One hiccup in the aforementioned discussion of on-the-fly "guidance" (i.e., conditioning) is that these concepts and intuitions apply only when x lives in a continuous state space or can be well-approximated as such; 3D coordinates of atoms naturally live in such a space, whereas biological sequences do not. This is because gradients with respect to x fundamentally capture how minutely perturbing an independent variable changes a dependent quantity, but we cannot quantitatively perturb an amino acid identity—we can only mutate it or leave it alone. One can consider "relaxing" discrete variables into a continuous state space or learning a continuous latent representation of discrete state spaces so as to be able to use these more standard gradient-based generative models (*52*). However, diffusion and flow matching models can be correctly developed anew for sequences (*53*–*55*), including on-the-fly guidance and conditioning methods (*56*, *57*)—these rely on estimating a "rate" matrix [rather than $\nabla_x \log p(x)$], which determines the probability of choosing a particular amino acid at a particular position as the sequence is iteratively generated. Notably, recent developments have shown that the most commonly used versions of these diffusion and flow matching models on sequences (namely, those with masking noise processes) are in fact equivalent to both masked language models (MLMs) and autoregressive models, thereby enabling simplification of the training and generation algorithms (*56*).

Beyond guidance for on-the-fly conditioning, an alternative strategy is to make use of classical Markov chain Monte Carlo (*58*); such an approach is applicable for models whose likelihood can be efficiently estimated (up to a constant), and consequently not practically useful for diffusion and related models where estimating the likelihood is computationally expensive.

All three conditioning strategies have limitations determined by the amount, quality, and suitability of training data; choice of model parameterization; and how the model was trained. It is also worth noting

that the commonly used approach of updating the parameters of an unconditional model by further training it with the same loss function on a curated set of sequences (e.g., a set of sequences with high catalytic activity for the reaction of interest)—known as "fine-tuning"—does not correspond to any well-defined statistical operation of combining the unconditional model with the predictive distribution for the property of interest. Although such an approach can sometimes be useful in practice, proper statistical conditioning provides a formal framework with which to understand how one is combining information from different sources. This formal framework better supports understanding when these combinations should be helpful and also enables principled development of modeling capabilities. Moreover, some practical limitations of this type of fine-tuning have recently been reported, suggesting they may struggle to extrapolate to previously unseen property values. (*56*). Note that in a distinct but related line of work known as "active learning," one can leverage classical AI approaches to determine which variants to measure next in the laboratory so as to improve the accuracy of $f(s)$ (*59*).

A tour of generative models in action for protein engineering

Having described important conceptual aspects of conditional generative modeling, we now take a more detailed tour of how these approaches are being used for protein engineering. One possible bifurcation for such a discussion is between so-called de novo design approaches wherein one attempts to create proteins without starting from an entirely known protein (*25*, *26*, *60*, *61*), and redesign approaches wherein one starts with an entirely known protein (*12*, *14*). However, many of the AI tools developed for one of these are often used for the other—thus, one might argue that this distinction is becoming increasingly blurred. Not to mention that if de novo AI tools are trained on many examples of existing proteins, can we still call them de novo? For these reasons, we do not strongly anchor our exposition on these categories.

From family-specific to pan-protein sequence generative models

One way to generate new protein sequences for a function of interest is to sample variants based on statistics of a functional wild-type protein and its homologs. In a recent example (*62*), a chorismate mutase wild-type sequence was chosen to curate a set of sequences hoped to be evolutionarily related to it. From these sequences, a multiple sequence alignment (MSA) was constructed and used to train a Potts model (*63*–*65*), an unconditional density model, $p_{\theta_{MSA}}(s)$ typically trained on just one protein family defined by an MSA. Implicitly, the model was conditioned on the wild type used to curate the MSA. After training, the model was used to sample protein variants, which were then expressed in *Escherichia coli*, and their enzymatic activity measured. A distribution of activities similar to that of the proteins in the MSA was found. Such an approach by itself does not directly give much control in tailoring the distribution toward certain desired properties: One can only adjust the "temperature" parameter, which affects the entropy of the distribution. It also relies on natural homologs in a database, and, like directed evolution, relies on a good starting point (the wild type). Nevertheless, it shows that evolutionarily related samples curated from a database provide a solid foundation on which to create functional diversity of sequences for a target family. Furthermore, such methods could provide alternative starting points for directed evolution.

Potts models and their richer-capacity relatives such as VAEs (*66*) are trained anew for each individual protein family. But given the vast number of sequences across all known families in the UniRef (*38*) database, we might want to build a single, large pan-protein model that contains all these family sequences at once so that we need train a model only once. Doing so enables information sharing across protein families. MSA Transformer (*67*) and its cousin the Neural Potts model (*68*) did just that—these are density models over all MSAs created from sequences in UniRef50, $p_\theta(\{s_i\}_{MSA})$. The utility of these models was shown only for contact and structure prediction (rather than for sampling), a task for which AlphaFold2 (*69*) soon dominated, but the spirit of this pan-protein sequence model persists in an MSA-free manner, in the form of the more general "language" models (*70*–*78*). Of these, ESM2 and other MLMs—which have been used to generate sequences—are not, in the strictest sense, considered generative models (*79*). These models are not amenable to correct statistical sampling: MLMs can only be correctly sampled when their training masking rates span 0 to 100% (*56*), whereas, for example, ESM2 has a fixed 15% masking rate (*76*). By comparison, its relatives, such as ESM3 (*80*) and models that do span the full masking rate range, can be sampled from correctly, as can autoregressive models (*81*, *82*) and diffusion and related models. ESM3 was trained on three primary modalities: sequence; discretized ("tokenized") structure at the all-atom level; and free-form, text-based function annotation. Meanwhile, numerous other multimodal models were also developed (*54*, *83*, *84*), following on the lead of ProstT5 and SaProt who introduced models over both sequence and discretized structure (*79*, *85*). All the aforementioned uses of discretized structure followed on ideas introduced in FoldSeek (*86*).

Backbone generative models

A typical AI workflow inherited from the traditional CPD for de novo protein engineering is to first sample backbone structures from a backbone generative model, $b \sim p(b)$, and then to provide each structure, b, in turn, as input to a backbone-conditioned sequence generative model (an "inverse folding" model) to sample protein sequences, $s \sim p(s \mid b)$. One can appeal to the chain rule of probability to justify doing so—namely, this two-step procedure samples from the joint distribution, $s, b \sim p(s, b)$ by using the equivalent factorized version, $s, b \sim p(s \mid b)p(b)$. We now discuss backbone generative models before moving to inverse folding.

A number of early backbone generative models were developed, some trained for specific tasks such as antibodies (*87*), one a diffusion model (*88*), but none trained broadly on PDB or validated experimentally (*88*–*90*). The first experimentally validated backbone generative models, Chroma and RFdiffusion (*91*, *92*)—both diffusion models trained on all of PDB—unlocked usage by the protein engineering community at large. Diffusion models start sampling from a noise distribution of backbone 3D coordinates for the protein—coordinates that do not initially correspond to any folded protein—and then slowly "diffuse" (denoise) these toward coordinates that correspond to a structured backbone. The gradual morphing toward structure is achieved by following an increasingly less noisy estimate of the gradient, $\nabla_b \log p_\theta^t(b)$, over the model's diffusion time, t, such that $\nabla_b \log p_\theta^{t=t_{\text{final}}}(b)$ is the gradient of the probability density of the training data. [Note: Technically $\nabla_x \log p(x)$ is referred to as the score, and the models often as "score-based" models, but because herein we employ the separate meaning of "scoring sequences," we instead call these models "gradient-based" for clarity.] The key component for training diffusion models is in learning the parameters of a "denoising" model, which learns to take as input a noisy sample of b from partway in the diffusion process, to predict a noise-free version of b. Learning the denoising model is intimately tied to estimating $\nabla_b \log p_\theta^t(b)$ (*93*).

These backbone generative models are not particularly useful if we cannot condition on some desired properties, such as a small molecule, protein, DNA, or RNA binding partners. In RFdiffusion, conditioning on a known motif in the protein such as an active site or binding interface ("motif scaffolding") was "baked in," meaning it was set up at the time of training the model; by contrast, ligand pocket generation is performed with on-the-fly conditioning. Chroma uses only on-the-fly conditioning. Both backbone generative models can condition on symmetry, geometry, binding targets, or functional motifs (including epitope binding motifs and enzyme active sites) (*91*, *92*, *94*–*99*). Chroma can additionally condition on natural language annotations. It is important

to note that, as with all generative model sampling, samples can only be as adherent to the specified conditional property as the supervised data allow. With the guidance approach, this means as good as the predictive model used to guide the sampling process. In the "baked-in" case, this means as good as the overall conditional generative model was able to extract information from supervised data at the time it was trained.

Evaluation of backbone generative models remains challenging. One key metric used is "designability" (*100*), wherein a generated backbone is given to an inverse folding model to design an amino acid sequence, which in turn is given to a structure prediction model such as AlphaFold2 (*69*), from which confidence metrics are used as a proxy for how likely the backbone is to have amino acids that would truly fold up into the generated structure. However, such an evaluation is affected by biases in the inverse folding model and the structure predictor, both of which are likely to be biased toward naturally occurring proteins whose manifold we may seek to explore beyond (*101*).

Inverse folding models

As mentioned earlier, backbone generative models are typically paired with an inverse folding model, $p(s \mid b)$, to obtain protein sequences that are compatible with the generated backbones. Initially developed before backbone generative models to mitigate classical energy-based side-chain rotamer packing issues, the first model (*102*) formed the basis for subsequent such models (*103*, *104*). These models use a graph-based representation of the 3D backbone structure as input that includes geometric information in a manner adherent to properties of physical 3D space, such as rotation invariance.

Inverse folding models did not become widely used until later when trained on much larger data sets compared to the ~20,000 CATH (*105*) protein chains of the first models. Contemporaneous models ProteinMPNN (*103*) and ESM-IF1 (*104*) were trained on all ~200,000 sequence-structure pairs in the PDB, and ESM-IF1 additionally used all sequence-structure pairs in AlphaFold DB (*106*) for a total of roughly 12 million pairs. Both are autoregressive models, although ProteinMPNN is an "any-order" autoregressive model, enabling sampling any part of the sequence while fixing the rest to some predetermined amino acids. In later work, ProteinMPNN was specialized to soluble proteins (*107*), thermostable proteins (*108*), and ligand-binding proteins by training on additional, task-specific data (*109*). Some backbone generative models have a special twist in that they provide a distribution over not only amino acids at each position, but also over side-chain torsional angles, χ, conditioned on the amino acid sequence, $p(\chi \mid s)$, yielding all-atoms and their geometry (*88*, *92*).

There are two main conceptual weaknesses in most inverse folding models, both having to do with the fact that they are typically trained with a single structure paired with a single sequence, whereas this relationship is actually many-to-many (*110*, *111*). First, sequences sampled from the current inverse folding models may be even more likely to fold into a structure different from the conditioned structure (*112*). It would make sense to pair each sequence with an ensemble of likely structures, possibly weighted by their relative probability, because proteins have multiple conformational states—in an ideal world, we would condition on that sequence's Boltzmann distribution. However, these multistate data are not available at scale. In principle, one could leverage models for predicting conformational ensembles from sequence to help mitigate this issue (*113*), but these are still in early stages (*114*). Second, because each structure can be potentially encoded by multiple sequences, we may be losing diversity of sequences sampled from the current inverse folding models. Therefore, it may make sense to train these models with multiple sequences that all have a high probability of folding into one structure. A step in this direction is to replace, at training time, each sequence with an MSA containing that sequence and its homologs—this can encourage the model to be aware of multiple sequences mapping to one structure (*115*). However, these sequences may not truly have the same structure.

The performance of an inverse folding model cannot be accurately estimated in silico, nor at scale in the laboratory, making comparative evaluation challenging. The primary in silico metric used to compare models is the "sequence recovery" rate—how often a naturally existing protein has its sequence recovered when the model is given its backbone structure (*116*). A recovery rate of 50% (e.g., 52.4% by ProteinMPNN) is generally considered better than 30% (e.g., 32.9% by Rosetta) (*103*). However, as this rate continues to increase, at some point it is too high such that the model has memorized a mapping from structure to sequence, rather than encoded the broader distribution of sequences that tend to form the specified backbone. Even if we knew the ideal recovery rate, this metric, in any case, does not reflect anything about the conformational landscape—our true interest. Another evaluation metric is the "refolding" score of a sampled sequence, which refers to how well we recover the conditioned structure when we give the inverse-folded sequence to a structure predictor such as AlphaFold (*18*). Such a metric may be biased by the structure predictor's flaws. Altogether, because of these inadequate metrics, we cannot truly gauge which model is the best; consequently, we cannot tell from these evaluation metrics whether the extra predicted structural data from AlphaFold proved useful, or even detrimental, for ESM-IF.

Additionally, many proteins of interest have important disordered regions, such as the loops of antibodies that confer binding. Inherently, current inverse folding models cannot provide much utility for such cases without knowing the binding partner and bound conformation of the two partners, including the disordered regions (*101*, *117*).

Although inverse-folding models have proven to be a key component of de novo protein engineering, they can also be used independently of backbone generative models to generate new sequences for a protein family, conditioning on the wild-type structure (*118*, *119*).

Joint generation of sequence and structure

The two-step de novo sampling strategy just discussed, $s, b \sim p(s \mid b)p(b)$, is statistically exact only if we can perfectly estimate each of the factored distributions. Because probability estimation is never exact, directly estimating $p(s, b)$ might prove beneficial over estimation of the two factors independently, although the relative benefits of such a strategy remain unclear. One way to estimate this joint distribution is with a so-called "all-atom" modeling approach, wherein all atoms and their 3D spatial positions are generated at once [note some factored approaches also model all-atom configurations (*88*, *92*)]. One major practical motivation for joint modeling—with, say, a diffusion model—is to natively enable conditioning on certain configurations of atoms, namely, (i) atom configurations of functional motifs, such as active sites not defined by entire amino acids, and (ii) atoms arising from non–protein binding targets, such as RNA, DNA, and small molecules. Although it is possible to achieve such conditioning by instead "baking in" atomistic conditioning, the all-atom approach within diffusion-type models offers an elegant and perhaps more useful direction. However, such joint modeling is technically challenging; for example, how many atoms should be generated for each protein if its amino acid sequence is not yet known? Strategies are emerging to tackle this issue (*120*–*124*). One can sidestep simultaneous generation at the all-atom level by instead "codesigning" both the backbone and amino acids, although this strategy may be less amenable to general conditioning, such as on atom motifs, because not all atoms are explicitly modeled (*54*, *125*).

Generative models for scoring and representation learning

In some cases, "generative" models are not developed to generate new samples. Instead, they may be used to "score" sequences as good or bad under that model, according to the likelihood or approximations thereof (*63*, *66*, *126*). When using a generative model to score

sequences, we are asking how much the sequence statistically "looks like" the training data, filtered through the lens of the chosen model. This task is sometimes called "zero-shot" prediction because "zero" labeled data are explicitly used. For example, we may score sequences with an inverse folding model, a sequence-only family-based model, or a pan-protein model that may be MSA and/or structure-aware (*62*, *67*, *103*). Notably, when benchmarking models for zero-shot scoring on variants with multiple mutations (relative to a wild type) (*127*), top-performing models are not typically those trained only on single sequences. Rather, they employ structure information, are MSA-aware, or do both (*81*, *128*, *129*). A zero-shot score may reflect substantially on stability (*130*) but may also contain functional information. A separate use case for generative models is to extract representations (e.g., a layer of a neural network used to model the density) to then be used for further downstream machine learning, typically supervised, such as to predict protein structure or properties (*73*, *76*, *78*, *131–133*). Notably, existing benchmark data sets (for both zero-shot and supervised learning) leave much to be desired owing to the limited number of mutations from wild type—the exact part of design space that we hope AI can take us beyond. Zero-shot models can be used in tandem with library design, discussed in the next section.

The role of library design and its relation to synthesis costs

Even the most sophisticated machine learning models described above, trained on large corpuses of data, are not so accurate that we can simply take a single output and be done—we need to design sets of proteins as a library. Moreover, we are likely to need to do so for the foreseeable future. Depending on the complexity of the challenge, the amount of information already known and infused into the modeling, and the proportion of the protein sequence being designed, we may need to generate anywhere from tens to millions of candidate sequences to be tested in the laboratory to have some hope of finding a single protein with specified properties, let alone several lead candidates. This raises two additional challenges: jointly designing batches of protein variants, and accounting for the cost of gene synthesis.

The less accurate a protein-design method, the more sequences we will have to test in the lab. At some low enough level of accuracy, it may be beneficial to replace designing specific sequences that we exactly synthesize, with instead designing for a stochastic synthesis process wherein we learn parameters of a sequence distribution that correspond to knobs that control the synthesis (*134*, *135*). For example, if one is willing only to specify the probability of A, C, T, and G at each position in a fixed length sequence, then, for the same synthesis cost, one might generate several orders of magnitude more sequences (*134*). This of course comes at the expense of less control over each individual sequence. Even as full synthesis costs decline, this scenario is likely to remain relevant for many years to come.

A related but distinct problem from that of library design is that of AI-based iterative design of experiments, of which a particular instantiation is machine learning–guided directed evolution (*35*, *58*, *59*, *132*, *134*, *136*, *137*). The problem setup here is to figure out how to make the most effective use of an experimental budget of time, labor, and money, over several iterative rounds of experiment, as with active learning mentioned earlier (*59*).

What's easy, what's hard?

Where do we stand in our progress on various protein-engineering problems, such as designing protein binders, enzyme engineering, and so forth? Pre-AI, hit rates for libraries of computationally designed protein binders against target therapeutic proteins were typically less than 0.05% (*138*). Now, with the help of generative models, along with the ever-growing PDB, hit rates have increased in some cases by orders of magnitude, allowing routine characterization in microplates instead of labor-intensive high-throughput screens (*139–141*). However, generative models alone are not sufficient—the field currently depends heavily on post hoc filtering of generated proteins by way of calls to AI-based structure predictors as well as more traditional biophysics-based criteria (*141*). Designing proteins to target any biomolecule, such as DNA, RNA, and small molecules, rather than just a protein, remains more challenging, owing to a paucity of structural data on these molecules in complex with proteins (*94*, *142*, *143*).

Most successfully designed protein binders are small, globular proteins composed primarily of helices and sheets, with minimal loop content, whereas natural proteins, such as antibodies, frequently rely on loops for molecular recognition. Although antibody-specific generative models are showing promise (*140*, *144–146*), there are no general-purpose models to robustly design flexible loops and intrinsically disordered regions.

Perhaps the hardest problem is that of enzyme design, which requires precise atomic knowledge, including configuration of the active site. For a simple and well-studied reaction, an expert chemist can posit an idealized atomic active site configuration—a "theozyme"—by intensive quantum mechanics calculations (currently not AI-based) (*147*). However, for more complex reactions, we can only extract the active site of a well-characterized enzyme, then condition on it to generate a full enzyme. Such a strategy might enable miniaturization, increasing stability, or monomerization (*97*, *148*), but it is not sufficient to design catalytic activity for new reactions. Moreover, it is likely to yield lower catalytic activity compared to the original enzyme, thus requiring further optimization by DE (*148*).

In an alternative AI-based enzyme design strategy, one can directly generate enzyme sequences without modeling active-site structures by conditioning on functional annotations such as Enzyme Commission numbers (*56*, *149*). Such an approach is unlikely to generate functional enzymes for previously unknown reactions.

Outlook and discussion

Much of current AI-based protein design relies on protein structure predictors, either explicitly within a generative loop (*141*) or by way of post hoc filtering. Yet structure prediction models are anchored on the natural protein universe and may be insufficient to broadly judge useful, designed sequences. As the field of structure prediction progresses (*150–153*), it is likely that much of this progress will be translated to protein engineering. Newer directions such as methods for infusing AlphaFold with cryo–electron microscopy data or molecular dynamics data have recently emerged (*154–156*). Moreover, such trends remind us that AI depends critically on costly, invaluable, and ideally public databases whose contents require laborious, expensive wet lab experiments (*157*).

A ubiquitously nagging question is to what extent can AI models generalize to "new" parts of protein space, with quotes to indicate that defining the word new in this context is itself a difficult problem (*137*). As this generalization issue will likely always exist, one might consider rationally blending biophysics-based models—which should be equally good throughout protein space—with AI models, which dominate in performance only sufficiently close to the training data, to get the best of both worlds (*158*).

One conundrum with development of AI-based protein engineering is how to track progress—most papers do not carry out a systematic, realistic comparison of methods because doing so is too expensive, requiring making and measuring proteins in a wet lab. More generally, generative models are extremely difficult to evaluate. In contrast, classical tasks in AI, such as prediction, can be more readily evaluated in silico. For example, the CASP competition (*159*), critical to the progress of structure prediction, did not require synthesis and measurement of proteins after competition submissions. Ideally, the protein engineering community would craft more robust in silico benchmark problems (*160*), using more useful metrics, but doing so for many design-related problems is itself a difficult challenge. Competitions that include wet lab validation are also likely to prove useful (*161*, *162*). Nevertheless, more baselines should be encouraged for publication of new methods claiming to provide a win.

REFERENCES AND NOTES

1. S. B. Ebrahimi, D. Samanta, *Nat. Commun.* **14**, 2411 (2023).
2. T. M. Caradonna, A. G. Schmidt, *NPJ Vaccines* **6**, 154 (2021).
3. R. M. Rivero, R. Mittler, E. Blumwald, S. I. Zandalinas, *Plant J.* **109**, 373–389 (2022).
4. Z. Zhao, A. R. Fernie, Y. Zhang, *Curr. Opin. Plant Biol.* **85**, 102699 (2025).
5. C. Li, R. Zhang, J. Wang, L. M. Wilson, Y. Yan, *Trends Biotechnol.* **38**, 729–744 (2020).
6. A. Miserez, J. Yu, P. Mohammadi, *Chem. Rev.* **123**, 2049–2111 (2023).
7. E. R. R. Moody *et al.*, *Nat. Ecol. Evol.* **8**, 1654–1666 (2024).
8. K. Chen, F. H. Arnold, *Proc. Natl. Acad. Sci. U.S.A.* **90**, 5618–5622 (1993).
9. H. Zhao, L. Giver, Z. Shao, J. A. Affholter, F. H. Arnold, *Nat. Biotechnol.* **16**, 258–261 (1998).
10. G. P. Smith, *Science* **228**, 1315–1317 (1985).
11. J. McCafferty, A. D. Griffiths, G. Winter, D. J. Chiswell, *Nature* **348**, 552–554 (1990).
12. F. H. Arnold, *Angew. Chem. Int. Ed.* **57**, 4143–4148 (2018).
13. C. Zeymer, D. Hilvert, *Annu. Rev. Biochem.* **87**, 131–157 (2018).
14. Y. Wang *et al.*, *Chem. Rev.* **121**, 12384–12444 (2021).
15. S. B. J. Kan, R. D. Lewis, K. Chen, F. H. Arnold, *Science* **354**, 1048–1051 (2016).
16. P. S. Coelho, E. M. Brustad, A. Kannan, F. H. Arnold, *Science* **339**, 307–310 (2013).
17. B. I. Dahiyat, S. L. Mayo, *Science* **278**, 82–87 (1997).
18. B. Kuhlman *et al.*, *Science* **302**, 1364–1368 (2003).
19. L. Jiang *et al.*, *Science* **319**, 1387–1391 (2008).
20. C. E. Tinberg *et al.*, *Nature* **501**, 212–216 (2013).
21. P. Lu *et al.*, *Science* **359**, 1042–1046 (2018).
22. D. A. Silva *et al.*, *Nature* **565**, 186–191 (2019).
23. X. Pan *et al.*, *Science* **369**, 1132–1136 (2020).
24. N. F. Polizzi, W. F. DeGrado, *Science* **369**, 1227–1233 (2020).
25. T. Kortemme, *Cell* **187**, 526–544 (2024).
26. P.-S. Huang, S. E. Boyken, D. Baker, *Nature* **537**, 320–327 (2016).
27. S. K. Burley *et al.*, *Nucleic Acids Res.* **51** (D1), D488–D508 (2023).
28. R. F. Alford *et al.*, *J. Chem. Theory Comput.* **13**, 3031–3048 (2017).
29. J. K. Leman *et al.*, *Nat. Methods* **17**, 665–680 (2020).
30. D. Baker, *Protein Sci.* **28**, 678–683 (2019).
31. Y. Nevers, N. M. Glover, C. Dessimoz, O. Lecompte, *Genome Biol.* **24**, 135 (2023).
32. M. M. Vopson, *AIP Adv.* **11**, 105317 (2021).
33. D. Röthlisberger *et al.*, *Nature* **453**, 190–195 (2008).
34. D. Brookes, A. Busia, C. Fannjiang, K. Murphy, J. Listgarten, in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference* (Association for Computing Machinery, Inc, 2020), pp. 189–190.
35. D. H. Brookes, H. Park, J. Listgarten, "Conditioning by adaptive sampling for robust design" in *Proceedings of the 36th International Conference on Machine Learning* (2019), vol. 97, pp. 773–782.
36. J. C. Bowden, S. Levine, J. Listgarten, "Leveraging Discrete Function Decomposability for Scientific Design" in *International Conference on Learning Representations* (2026).
37. D. P. Kingma, M. Welling, "Auto-Encoding Variational Bayes" in *International Conference on Learning Representations* (2014).
38. B. E. Suzek, H. Huang, P. McGarvey, R. Mazumder, C. H. Wu, *Bioinformatics* **23**, 1282–1288 (2007).
39. C. Fannjiang, J. Listgarten, "Autofocused oracles for model-based design" in *Proceedings of the 34th International Conference on Neural Information Processing Systems* (2020), pp. 12945–12956.
40. K. P. Murphy, *Probabilistic Machine Learning: Advanced Topics* (MIT Press, 2023).
41. R. Rafailov *et al.*, *Adv. Neural Inf. Process. Syst.* **36,** 53728–53741 (2024).
42. T. Widatalla, R. Rafailov, B. Hie, bioRxiv [Preprint] (2024). https://doi.org/10.1101/2024.05.20.595026
43. H. C. Gasser, D. A. Oyarzún, J. A. Alfaro, A. Rajan, *Protein Eng. Des. Sel.* **38**, gzaf003 (2025).
44. M. S. Albergo, E. Vanden-Eijnden, "Building Normalizing Flows with Stochastic Interpolants" in *International Conference on Learning Representations* (2023).
45. Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, M. Le, Flow Matching for Generative Modeling" in *International Conference on Learning Representations* (2023).
46. X. Liu, C. Gong, Q. Liu, "Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow" in *International Conference on Learning Representations* (2023).
47. J. Ho, A. Jain, P. Abbeel, Denoising Diffusion Probabilistic Models. *Adv. Neural. Info. Process. Syst.* **33**, 6840–6851 (2020).
48. Y. S. Song *et al.*, Score-Based Generative Modeling through Stochastic Differential Equationsin *International Conference on Learning Representations* (2021).
49. J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, S. Ganguli, S. Edu, "Deep Unsupervised Learning using Nonequilibrium Thermodynamics" in *International Conference on Machine Learning* (2015), pp. 2256–2265.
50. A. Hyvärinen, *J. Mach. Learn. Res.* **6**, 695–709 (2005).
51. P. Dhariwal, A. Q. Nichol, "Diffusion Models Beat GANs on Image Synthesis" in *Proceedings of the 35th International Conference on Neural Information Processing Systems* (2021), pp. 8780–8794.
52. H. Stark, B. Jing, C. Wang, G. Corso, B. Berger, R. Barzilay, T. Jaakkola, "Dirichlet Flow Matching with Applications to DNA Sequence Design" in *International Conference on Machine Learning* (ML Research Press, 2024), vol. 235, pp. 46495–46513.
53. A. Campbell *et al.*, "A continuous time framework for discrete denoising models" in *Proceedings of the 36th International Conference on Neural Information Processing Systems* (2022), pp. 28266–28279.
54. A. Campbell, J. Yim, R. Barzilay, T. Rainforth, T. Jaakkola, "Generative Flows on Discrete State-Spaces: Enabling Multimodal Flows with Applications to Protein Co-Design" in *International Conference on Machine Learning* (ML Research Press, 2024), vol. 235, pp. 5453–5512.
55. S. Alamdari *et al.*, bioRxiv 2023.09.11.556673 [Preprint] (2024). https://doi.org/10.1101/2023.09.11.556673
56. J. Xiong, I. Gaur, M. Lukarska, H. Nisonoff, L. M. Oltrogge, D. F. Savage, J. Listgarten, ProteinGuide: On-the-fly property guidance for protein sequence generative models. arXiv:2505.04823 [cs.LG] (2025).
57. H. Nisonoff, J. Xiong, S. Allenspach, J. Listgarten, Unlocking Guidance for Discrete State-Space Diffusion and Flow Models. *International Conference on Representation Learning* 2025., 36052–36106 (2025).
58. P. Emami, A. Perreault, J. Law, D. Biagioni, P. St. John, *Mach. Learn. Sci. Technol.* **4**, 025014 (2023).
59. P. A. Romero, A. Krause, F. H. Arnold, *Proc. Natl. Acad. Sci. U.S.A.* **110**, E193–E201 (2013).
60. I. V. Korendovych, W. F. DeGrado, *Q. Rev. Biophys.* **53**, e3 (2020).
61. S. P. Ho, W. F. DeGrado, *J. Am. Chem. Soc.* **109**, 6751–6758 (1987).
62. W. Russ *et al.*, Evolution-based design of chorismate mutase enzymes. bioRxiv 2020.04.01.020487 [Preprint] https://doi.org/10.1101/2020.04.01.020487 (2020).
63. T. A. Hopf *et al.*, *Nat. Biotechnol.* **35**, 128–135 (2017).
64. F. Morcos *et al.*, *Proc. Natl. Acad. Sci. U.S.A.* **108**, E1293–E1301 (2011).
65. M. Figliuzzi, P. Barrat-Charlaix, M. Weigt, *Mol. Biol. Evol.* **35**, 1018–1027 (2018).
66. A. J. Riesselman, J. B. Ingraham, D. S. Marks, *Nat. Methods* **15**, 816–822 (2018).
67. R. M. Rao *et al.*, "MSA Transformer" in *Proceedings of the 38th International Conference on Machine Learning*, M. Meila, T. Zhang, Eds., vol. 139 of *Proceedings of Machine Learning Research* (PMLR, 2021).
68. T. Sercu *et al.*, bioRxiv 2021.04.08.439084 [Preprint] (2021). https://doi.org/10.1101/2021.04.08.439084
69. J. Jumper *et al.*, *Nature* **596**, 583–589 (2021).
70. A. Madani *et al.*, *Nat. Biotechnol.* **41**, 1099–1106 (2023).
71. A. Elnaggar *et al.*, *IEEE Trans. Pattern Anal. Mach. Intell.* **44**, 7112–7127 (2022).
72. A. Rives *et al.*, *Proc. Natl. Acad. Sci. U.S.A.* **118**, e2016239118 (2021).
73. R. Rao *et al.*, *Adv. Neural Inf. Process. Syst.* **32**, 9689–9701 (2019).
74. N. Ferruz, S. Schmidt, B. Höcker, *Nat. Commun.* **13**, 4348 (2022).
75. N. Brandes, D. Ofer, Y. Peleg, N. Rappoport, M. Linial, *Bioinformatics* **38**, 2102–2110 (2022).
76. Z. Lin *et al.*, *Science* **379**, 1123–1130 (2023).
77. K. K. Yang, N. Fusi, A. X. Lu, *Cell Syst.* **15**, 286–294.e2 (2024).
78. E. C. Alley, G. Khimulya, S. Biswas, M. AlQuraishi, G. M. Church, *Nat. Methods* **16**, 1315–1322 (2019).
79. J. Su *et al.*, SaProt: Protein Language Modeling with Structure-aware Vocabulary. *Twelfth International Conference on Learning Representations* (2024).
80. T. Hayes *et al.*, *Science* **387**, 850–858 (2025).
81. T. Truong, T. Bepler, *Adv. Neural Inf. Process. Syst.* **36**, 77379–77415 (2023).
82. E. Nijkamp, J. A. Ruffolo, E. N. Weinstein, N. Naik, A. Madani, *Cell Syst.* **14**, 968–978.e3 (2023).
83. B. Jing *et al.*, bioRxiv 2025.09.03.672144 [Preprint] (2025); https://doi.org/10.1101/2025.09.03.672144.
84. X. Wang, Z. Zheng, F. YE, D. Xue, S. Huang, Q. Gu, "DPLM-2: A Multimodal Diffusion Protein Language Model" in *The Thirteenth International Conference on Learning Representations* (2025).
85. M. Heinzinger *et al.*, *NAR Genom. Bioinform.* **6**, lqae150 (2024).
86. M. van Kempen *et al.*, *Nat. Biotechnol.* **42**, 243–246 (2024).
87. R. R. Eguchi, C. A. Choe, P. S. Huang, *PLOS Comput. Biol.* **18**, e1010271 (2022).
88. N. Anand, T. Achim, Protein Structure and Sequence Generation with Equivariant Denoising Diffusion Probabilistic Models. arXiv:2205.15019 [q-bio.QM] (2022).
89. N. Anand, P.-S. Huang, "Generative modeling for protein structures" in *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (2018), pp. 7505–7516.
90. B. L. Trippe *et al.*, "Diffusion Probabilistic Modeling of Protein Backbones in 3D for the motif-scaffolding problem" in *International Conference on Learning Representations* (2023).
91. J. L. Watson *et al.*, *Nature* **620**, 1089–1100 (2023).
92. J. B. Ingraham *et al.*, *Nature* **623**, 1070–1078 (2023).
93. P. Vincent, *Neural Comput.* **23**, 1661–1674 (2011).
94. L. An *et al.*, *Science* **385**, 276–282 (2024).
95. M. Glögl *et al.*, *Science* **386**, 1154–1161 (2024).
96. S. Vázquez Torres *et al.*, *Nature* **639**, 225–231 (2025).

97. A. Lauko *et al.*, *Science* **388**, eadu2454 (2025).
98. K. Wu *et al.*, *Science* **389**, eadr8063 (2025).
99. S. Wang *et al.*, *Nat. Mater.* **24**, 1644–1652 (2025).
100. H. Li, R. Helling, C. Tang, N. Wingreen, *Science* **273**, 666–669 (1996).
101. T. Lu, M. Liu, Y. Chen, J. Kim, P. S. Huang, *Cell Syst.* **16**, 101347 (2025).
102. J. Ingraham, V. K. Garg, R. Barzilay, T. Jaakkola, "Generative models for graph-based protein design" in *Proceedings of the 33rd International Conference on Neural Information Processing Systems* (2019), pp. 15820–15831.
103. J. Dauparas *et al.*, *Science* **378**, 49–56 (2022).
104. C. Hsu, R. Verkuil, J. Liu, Z. Lin, B. Hie, T. Sercu, A. Lerer, A. Rives, "Learning inverse folding from millions of predicted structures" in *Proceedings of the 39th International Conference on Machine Learning*, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, S. Sabato, Eds. (PMLR, 2022), vol. 162 of *Proceedings of Machine Learning Research*, pp. 8946–8970.
105. F. M. G. Pearl *et al.*, *Nucleic Acids Res.* **31**, 452–455 (2003).
106. J. Fleming *et al.*, *J. Mol. Biol.* **437**, 168967 (2025).
107. C. A. Goverde *et al.*, *Nature* **631**, 449–458 (2024).
108. H. Dieckhaus, M. Brocidiacono, N. Z. Randolph, B. Kuhlman, *Proc. Natl. Acad. Sci. U.S.A.* **121**, e2314853121 (2024).
109. J. Dauparas *et al.*, *Nat. Methods* **22**, 717–723 (2025).
110. A. M. Lesk, C. Chothia, *J. Mol. Biol.* **136**, 225–270 (1980).
111. P. N. Bryan, J. Orban, *Curr. Opin. Struct. Biol.* **20**, 482–488 (2010).
112. C. Norn *et al.*, *Proc. Natl. Acad. Sci. U.S.A.* **118**, e2017228118 (2021).
113. S. Lewis *et al.*, *Science* **389**, eadv9817 (2025).
114. R. W. Shuai, T. Lu, S. Bhatti, P. Kouba, P.-S. Huang, bioRxiv, 2025.09.30.679633 (2025).
115. L. Alessandro Silva, B. Meynard-Piganeau, C. Lucibello, C. Feinauer, *J. Stat. Mech.* **2025**, 084003 (2025).
116. B. Kuhlman, D. Baker, *Proc. Natl. Acad. Sci. U.S.A.* **97**, 10383–10388 (2000).
117. Y. Li *et al.*, *PLOS ONE* **20**, e0324566 (2025).
118. K. H. Sumida *et al.*, *J. Am. Chem. Soc.* **146**, 2054–2061 (2024).
119. H. Fei *et al.*, *Cell* **188**, 4674–4692.e19 (2025).
120. J. K. V. Butcher *et al.*, bioRxiv, 2025.09.18.676967 (2025).
121. T. Geffner *et al.*, "La-Proteina: Atomistic Protein Generation via Partially Latent Flow Matching" in *International Conference on Learning Representations* (2026).
122. Y. Chen, T. Lu, C. Zhao, H. K. Wayment-Steele, P. Huang, bioRxiv 2025.10.03.680398 [Preprint] (2025). https://doi.org/10.1101/2025.10.03.680398
123. W. Ahern *et al.*, bioRxiv, 2025.04.09.648075 [Preprint] (2025). https://doi.org/10.1101/2025.04.09.648075
124. A. J. Li, T. Kortemme, bioRxiv, 2025.10.18.683228 (2025).
125. S. L. Lisanza *et al.*, *Nat. Biotechnol.* **43**, 1288–1298 (2025).
126. E. N. Weinstein, A. N. Amin, J. Frazer, D. S. Marks, "Non-identifiability and the Blessings of Misspecification in Models of Molecular Fitness" in *Proceedings of the 36th International Conference on Neural Information Processing Systems* (2022), pp. 5484–5497.
127. P. Notin *et al.*, "ProteinGym: Large-Scale Benchmarks for Protein Design and Fitness Prediction" in *Proceedings of the 37th International Conference on Neural Information Processing Systems* (2023), pp. 64331–64379.
128. R. Weitzman *et al.*, "Protriever: End-to-End Differentiable Protein Homology Search for Fitness Prediction" in *Proceedings of the 42nd International Conference on Machine Learning*, vol. 267, pp. 66397–66418.
129. P. Li, X. Cheng, L. Song, E. Xing, bioRxiv **267**, 2024.12.02.626519 (2025).
130. J. Frellsen *et al.*, Zero-shot protein stability prediction by inverse folding models: A free energy interpretation. arXiv:2506.05596 [cs.LG] (2025).
131. C. Hsu, H. Nisonoff, C. Fannjiang, J. Listgarten, *Nat. Biotechnol.* **40**, 1114–1122 (2022).
132. B. J. Wittmann, Y. Yue, F. H. Arnold, *Cell Syst.* **12**, 1026–1045.e7 (2021).
133. K. Jiang *et al.*, *Science* **387**, eadr6006 (2025).
134. D. Zhu *et al.*, *Sci. Adv.* **10**, eadj3786 (2024).
135. E. N. Weinstein, A. N. Amin, W. S. Grathwohl, D. Kassler, J. Disset, D. Marks, "Optimal Design of Stochastic DNA Synthesis Protocols based on Generative Sequence Models " in *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, G. Camps-Valls, F. J. R. Ruiz, I. Valera, Eds. (PMLR, 2022), vol. 151 of *Proceedings of Machine Learning Research*, pp. 7450–7482.
136. J. Yang *et al.*, *Nat. Commun.* **16**, 714 (2025).
137. C. Fannjiang, J. Listgarten, *Cold Spring Harb. Perspect. Biol.* **16**, a041469 (2024).
138. L. Cao *et al.*, *Nature* **605**, 551–560 (2022).
139. V. Zambaldi *et al.*, Deepmind, De novo design of high-affinity protein binders with AlphaProteo. arXiv:2409.08022 [q-bio.BM] (2024).
140. Chai Discovery Team, J. Boitreaud, bioRxiv 2025.07.05.663018 [Preprint] (2025); https://doi.org/10.1101/2025.07.05.663018.
141. M. Pacesa *et al.*, *Nature* **646**, 483–492 (2025).
142. C. J. Glasscock *et al.*, *Nat. Struct. Mol. Biol.* **32**, 2252–2261 (2025).
143. D. M. Ichikawa *et al.*, *Nat. Biotechnol.* **41**, 1117–1129 (2023).
144. N. R. Bennett *et al.*, bioRxiv **14**, 2024.03.14.585103 (2025).
145. L. S. Mille-Fragoso *et al.*, bioRxiv 2025.09.19.677421 [Preprint] (2025); https://doi.org/10.1101/2025.09.19.677421.
146. E. Swanson, M. Nichols, S. Ravichandran, P. Ogden, bioRxiv 2025.09.26.678877 [Preprint] (2025); https://doi.org/10.1101/2025.09.26.678877.
147. D. J. Tantillo, J. Chen, K. N. Houk, *Curr. Opin. Chem. Biol.* **2**, 743–750 (1998).
148. A. H. W. Yeh *et al.*, *Nature* **614**, 774–780 (2023).
149. G. Munsamy *et al.*, bioRxiv 2024.05.03.592223 [Preprint] (2024); https://doi.org/10.1101/2024.05.03.592223.
150. G. Ahdritz *et al.*, *Nat. Methods* **21**, 1514–1524 (2024).
151. S. Passaro *et al.*, bioRxiv 2025.06.14.659707 [Preprint] (2025); https://doi.org/10.1101/2025.06.14.659707.
152. J. Abramson *et al.*, *Nature* **630**, 493–500 (2024).
153. N. Corley *et al.*, bioRxiv 2025.08.14.670328 [Preprint] (2025); https://doi.org/10.1101/2025.08.14.670328.
154. S. Bhakat, S. Vats, A. Mardt, E. M. Strauch, bioRxiv 2025.09.12.675912 [Preprint] (2025); https://doi.org/10.1101/2025.09.12.675912.
155. R. Raghu, A. Levy, G. Wetzstein, E. D. Zhong, Multiscale guidance of AlphaFold3 with heterogeneous cryo-EM data. arXiv:2506-4400 [cs.LG] (2025).
156. T. C. Terwilliger *et al.*, *Nat. Methods* **19**, 1376–1382 (2022).
157. J. Listgarten, *Nat. Biotechnol.* **42**, 371–373 (2024).
158. H. Nisonoff, Y. Wang, J. Listgarten, *ACS Synth. Biol.* **12**, 3242–3251 (2023).
159. J. Moult, J. T. Pedersen, R. Judson, K. Fidelis, *Proteins* **23**, ii–v (1995).
160. Z. Zheng *et al.*, MotifBench: A standardized protein design benchmark for motif-scaffolding problems. arXiv:2502.12479 [cs.LG] (2025).
161. C. Armer *et al.*, *Proteins* **93**, 2005–2014 (2025).
162. L. Fu *et al.*, *ACS Synth. Biol.* **13**, 3782–3787 (2024).

ACKNOWLEDGMENTS

Many researchers have made important contributions to AI-driven protein engineering—we apologize to those whose work we could not cite owing to space and citation limitations. Thank you to H. Nisonoff and B. (Junhao) Xiong for discussion. Thank you to H. Nisonoff, J. Lubin, C. Hsu, C. Fannjiang, and D. Savage for feedback on the manuscript. Thank you to M. Lukarska, J. Bowden, and F. Pardo-Avila for suggestions on figure preparation, and C. Hsu for motivating our figure through her thesis figure. Icons were adapted and used under Creative Commons license (CC0) from BioIcons.com: "Stem cell colony" by M. Tisch and "Flow cytometer cell sorter" by D. Croote. **Funding:** J.L. and H.J. were funded in part by the US Department of Energy, Office of Science, Office of Biological and Environmental Research, Lawrence Livermore National Laboratory BioSecure SFA within the Secure Biosystems Design program (SCW1710). This work was supported in part through the Chan Zuckerberg Initiative through the CZ Biohub Investigator Program. **Author contributions:** Conceptualization: H.J., J.L. Writing – original draft: H.J., J.L. Writing – review and editing: H.J., J.L. Visualization: H.J., J.L. Funding acquisition: J.L. Investigation: J.L. Supervision: J.L. Project administration: J.L. **Competing interests:** J.L. is on the scientific advisory board of Fable Therapeutics and Deep Apple Therapeutics and was previously on the scientific advisory board of Inscripta.

Submitted 27 October 2025; accepted 24 February 2026

10.1126/science.aec8444

How artificial intelligence is reengineering protein engineering

Jennifer Listgarten and Hanlun Jiang

Science **392** (6794), . DOI: 10.1126/science.aec8444

Editor's summary

Proteins, with their varied structure and chemistry, are the prime actors of biology and have long been targets for in vitro and in silico engineering. Generative protein models and other artificial intelligence (AI) tools are now being integrated into experimental workflows. Listgarten and Jiang reviewed advances in AI methods and discuss how statistical principles are being used to transform protein engineering through conditional generative modeling. Beyond the sizable advances so far, current challenges include designing functional enzymes, disordered proteins, and binders of all kinds, problems for which we currently lack sufficient training data. —Michael A. Funk