SAEBER: Sparse AutoEncoders for Biological Entity Risk[1]

Michael Yu
MATS 8.2

With
Apart Research

Abstract

Protein design models like RFDiffusion3 and folding models like RoseTTAFold3 are powerful tools that could be misused to design virulent or toxic proteins. Existing biosecurity screens operate on sequence similarity or structural homology and offer little insight into why a protein is flagged as hazardous. We apply mechanistic interpretability techniques to RFDiffusion3 and RoseTTAFold3 for the first time, training Matryoshka BatchTopK Sparse Autoencoders (SAEs) on intermediate activations collected during the diffusion and folding processes. Using a length-matched dataset of 275 sequences drawn from SafeProtein and UniProt, we train logistic regression probes on both raw and SAE-encoded activations to classify designs as virulent or benign, and we score individual SAE features for hazard association using univariate AUROC with Benjamini-Hochberg FDR correction. Our best probe, trained on SAE features from block 12 of RFDiffusion3, achieves an AUROC of 0.817 ± 0.10 under homology-clustered splits, outperforming the corresponding raw-activation probe by $+0.054$ AUROC. We also identify individual SAE features that fire on virulent designs at up to $\sim$0.8 AUROC, with feature quality increasing with layer depth. While our classifier does not surpass the current SOTA (DTVF, 0.92 AUROC), it is the first method to provide structural, feature-level explanations for virulence predictions in a protein design model, opening the door to runtime monitoring, steering, and interpretable guardrails during generation.

[1] Research conducted at the AIxBio Hackathon, April 2026

1. Introduction

Protein folding and design models are powerful tools but can be misused, for instance to assist in the generation of virulent or toxic proteins. Previous works screen proteins based on sequence similarity and structure, or at the DNA level. Unfortunately, these methods are often not explainable.

In this work, we apply interpretability techniques to RFDiffusion3 (RFD3) and RoseTTAFold3 (RF3) to classify and interpret model behavior on virulent proteins. We build a framework to collect the intermediate calculations (activations) during the diffusion process and train classifiers on both raw and SAE activations to determine if the model is designing around virulent motifs. We also train Matryoshka Batchtopk Sparse AutoEncoders (SAEs) to attribute which structural features of proteins are associated with virulence.

Finally, we benchmark the efficacy of using SAE activations vs raw activations to classify proteins as harmful or benign, as well as against other methods.

Our contributions are the following:

1. A suite of SAEs trained on RFD3 and RF3.
2. A database of SAE features correlated with virulence in proteins.
3. Benchmarking raw vs SAE activations for harmful protein classification.

Results can be viewed at https://www.raft.bio/blog/saeber

2. Related Work

FoldSAE is the closest work, where the authors trained SAEs on RFDiffusion. However, they found only very simple features predicting secondary structure (alpha helices and beta sheets). Also, RFDiffusion3 shares no code with its predecessor and operates on individual atoms instead of only on tokens (amino acids). Our work extends FoldSAE to all atom diffusion models for the first time and uses it to predict virulence features.

InterProt and InterPLM are trained on protein language models such as ESM, but not diffusion based protein design models. Goodfire has trained SAEs on Evo 2, and done similar diffusion generation interpretability on MatterGen, a materials design model

The existing SOTA for virulence classification is DTVF (2024). They used a benchmark dataset containing 576 VFs and 576 non-VFs for independent testing, following DeepVF's method to construct the same independent test set. They achieved an AUROC of 0.92, which represents a significant 8.57% improvement over the latest VF-Pred model released in 2024. Across four metrics of accuracy, F1-Score, specificity, and AUROC, the DTVF model surpasses the most recent models, with 1% increase in accuracy and 3.89% enhancement in specificity compared to VF-Pred.

VirulentHunter (2025) also shows strong performance with AUC improvements of 47.76% and 68.37% over MP4 and DeepVF respectively, though it doesn't report absolute AUROC values for direct comparison with DTVF's 0.92.

Although our method does not achieve SOTA for classification, it is the first to give practitioners features explaining why the protein is virulent or not. Our method would also be useful for steering the protein design process and intervening or detecting when a virulent sequence is about to be generated.

Sparse AutoEncoders

A Sparse Autoencoder is an autoencoder that projects activations into a higher dimension and back to the original dimension to conduct unsupervised nonlinear principle component analysis, decomposing polysemantic activation vector positions into sparse, ideally monosemantic features. It is trained with both a reconstruction loss and sparsity penalty for regularization. Formally, it consists of an encoder matrix W_enc [m, n] and decoder matrix W_dec [n, m] with an activation function σ, where m>n. Let x∈ R^n be the input activation vector from RFDiffusion3, z∈ ^{m} be the sparse activation vector after projection through W_enc and activation function, and x_hat∈ R^n be the reconstructed vector.

Then the forward pass is
$x_{hat} = W_{dec}\sigma(W_{enc}x)$
And the reconstruction loss L_{recon} is
$L_{recon} = || x_{hat} - x ||^2_2$
The sparsity penalty is typically implemented as a regularization term:
$L_{sparsity} = \lambda||z||_1$

where λ is the sparsity coefficient that controls the trade-off between reconstruction quality and sparsity.

The total loss function becomes:
$L_{total} = L_{recon} + L_{sparsity} = ||\hat{x} - x||_2^2 + \lambda||z||_1$

Matryoshka BatchTopK SAEs are a variant that uses a top-k activation pattern rather than $L_{sparsity}$, where only the top k activations in each batch dimension are kept active, with the rest set to zero. This approach provides more direct control over sparsity levels and can prevent issues such as feature absorption. They offer a balance of ease of training and near SOTA capabilities (as opposed to JumpReLU SAEs, which are extremely difficult to train and offer limited incremental benefits).

3. Methods

We use the foundry repository from RoseTTACommons to run RFDiffusion and modify engine.py to optionally collect activations using user specified hooks.
https://github.com/RosettaCommons/foundry/blob/production/models/rf3/README.md
https://github.com/RosettaCommons/foundry/tree/production/models/rfd3
All experiments can be run on a single L40 GPU or equivalent with at least 40GB VRAM.

We used SafeProtein (virulent only) + UniProt with filters NOT KW-0800 NOT KW-0843 NOT taxonomy_id:10239 AND length:100 TO 300 to construct positive and negative length-matched datasets for virulence. After filtering to sequences under 300 residues long (due to compute constraints), we have n=275 sequences.

We use the pdbs of the associated sequences as input motifs to RFDiffusion3, and set partial_t=5 to noise the motifs by 5 angstroms, denoising it back to a structure highly similar to the original motif. This simulates the activation distribution the model would output when denoising toxin-virus resembling proteins from scratch. For RF3, we simply fold the sequences. We then wrote an ActivationBuffer to stream activations from hooks to disk in activations.h5.

We collect from layers at roughly 25%, 50%, and 75% of the diffusion transformer in RFD3 and RF3, according to the literature on language model SAEs. We then train Matryoshka BatchTopK SAEs.

We now have 2 activation caches saved to disk

- RF3 activations (2 hooks)
    - diffusion_module.diffusion_transformer.blocks.12.conditioned_transition_block
    - diffusion_module.diffusion_transformer.blocks.16.conditioned_transition_block
- RFD3 activations (3 hooks)
    - diffusion_module.diffusion_transformer.blocks.6.transition_block
    - diffusion_module.diffusion_transformer.blocks.8.transition_block
    - diffusion_module.diffusion_transformer.blocks.12.transition_block

And 3 variables to sweep across

- Hook point (2 for RF3, 3 for RFDiffusion)
- Raw or SAE activations
- Homologous clustering or not

We train 20 logistic regression probes (binary cross entropy with L2 weight decay) on per-design feature vectors, using n=220 out of the full 275 designs, reserving 55 for the test set. We also generate a homology clustered version of the splits with mmseqs2 easy-cluster at 30% sequence

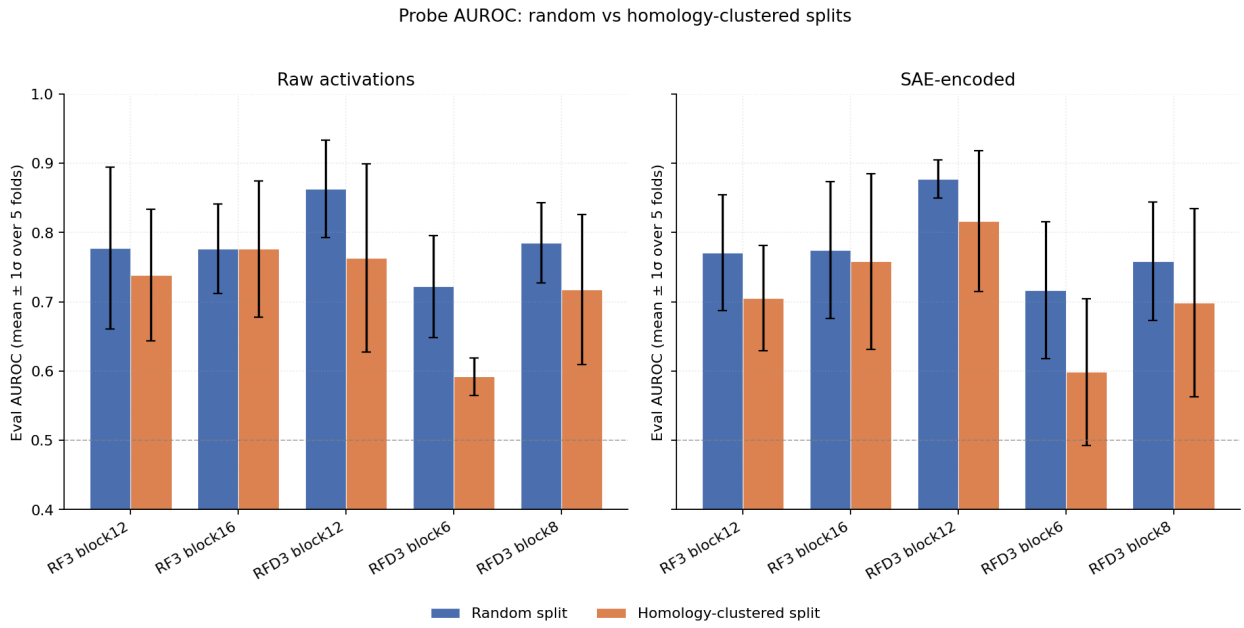
identity. This is to check the model is not simply remembering proteins from similar evolutionary lineages. We also do a simple baseline regression on sequence length to check the classifier is not relying on a trivial signal.

Independent of the probe sweep, we score every SAE feature against the full labels CSV using univariate AUROC. For the top-200 features by |AUROC - 0.5| we compute Mann-Whitney U p-values, then apply Benjamini-Hochberg FDR (False Discovery Rate) correction across all non-constant features to get q-values. BH correction matters because the dictionary has 12288 features and most carry no class information; without it the raw p-values would overstate significance. We treat features with $q < 0.05$ as discoveries.

For the top hazard-firing features we render PyMOL scripts that highlight residues where the feature fires most strongly. We read the CIF, parse it with biotite, and map token index to (chain, residue) by chain order. Token count equals residue count for our full-protein partial-diffusion designs (no ligands or nucleic acids).

4. Results

Figure 1. Random vs Homologous Splits, Raw activations vs SAE encoded activations



As shown in Figure 1, the best performing classifier, controlling for family memorization with homologous clustering, was trained on RFD3 Block 12 SAE, with AUROC of 0.817 ± 0.10. This increases to 0.877 ± 0.03 without clustering, showing memorization of family folds in RFD3. Notably, performance on RFD3 block 6 decreases sharply to nearly random for homology clustering, suggesting this block strongly learns family representations. Also note that this is lower than the current SOTA of DTVF at 0.92.

RFD3 memorizes more than RF3. In Figure 2, RFD3 has an average decrease of around 0.10 AUROC when controlling with clustering, while RF3 AUROC barely changes. However, overall they perform equally within the error bars.

Figure 2: Table of Random vs Homologous Clustering AUROC

cell	random	cluster	Δ
rfd3 block6 identity	0.722	0.592	**−0.130**
rfd3 block6 sae_encode	0.717	0.599	**−0.118**
rfd3 block8 identity	0.785	0.718	−0.067
rfd3 block12 identity	0.863	0.763	**−0.100**
rfd3 block12 sae_encode	0.877	0.817	−0.060
rf3 block12 identity	0.777	0.738	−0.039
rf3 block16 identity	0.776	0.776	0.000
rf3 block16 sae_encode	0.774	0.758	−0.016

Figure 3. SAE vs Raw Activations

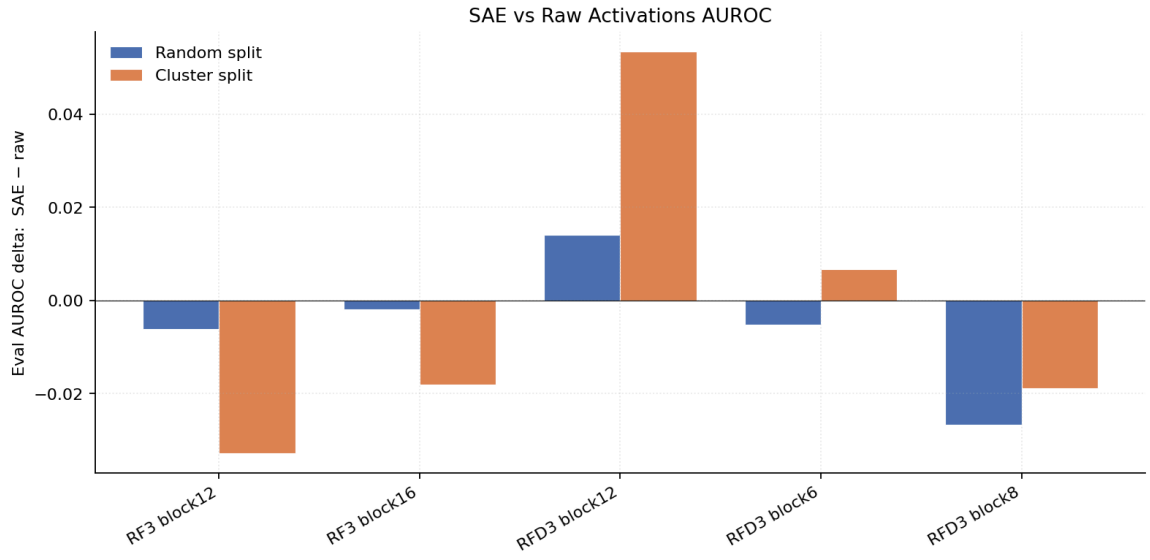
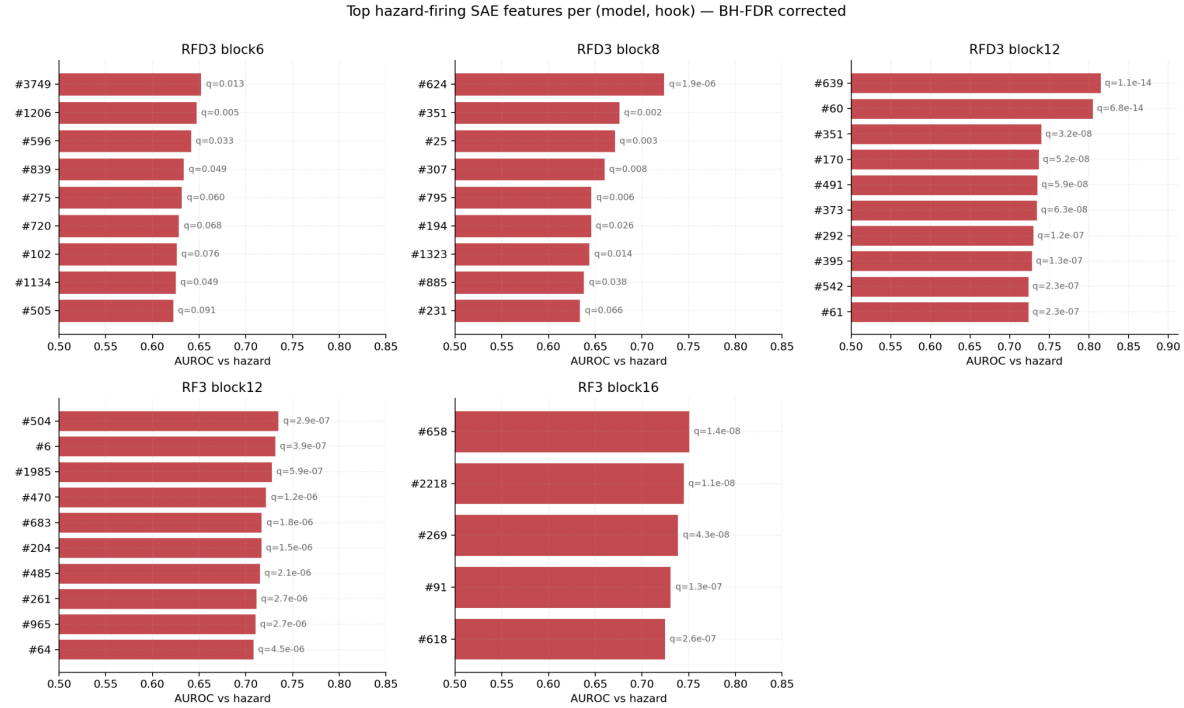


Figure 3 shows the performance of classifiers trained on SAE activations vs raw activations. SAE activations underperform in most blocks, possibly due to the sparse activation vector having lower effective dimensionality than the dense, raw activations. Surprisingly, for block12 of RFD3 cluster split SAE activations outperform raw activations by +0.054 AUROC (0.817 vs 0.763). We hypothesize this is due to block 12 untangling complex polysemantic features that provide signal to the logistic regression probe, and this polysemanticity increases at later layers. Further work could investigate other deeper layers of RFD3 for interesting features.

Figure 4. Top SAE Features Firing



We see the appearance of certain features firing above chance on positive class designs, up to ~0.8 AUROC. Also, the AUROC increases with layer depth, especially for RFD3, showing deeper layers have learned more meaningful features representing virulent proteins. Unfortunately I did not have time to label the features.

You can visualize them here, and let me know if there are labels you would add.

5. Discussion and Limitations

Limitations

- The number of samples is very small due to time and compute constraints, only 220 designs in the train dataset and 55 in the test dataset, lowering statistical significance and perhaps causing underfitting

- The layer selection was based on LLM transformer SAE literature, future work should do ablations and find hookpoints in more principled ways. For example, try to noise and denoise 100 virulent structures, then systematically knock out layers and see which one results in the biggest pLDDT (predicted Local Distance Difference Test) decrease.
- Scope limitations. Due to limited GPU compute, we filtered to sequences <300 amino acids long, filtering out longer proteins from SafeProtein.
- It would be interesting to breakdown the per category AUROC classification (/ etc.) but SafeProtein doesn't ship category labels, perhaps we could generate with external PFAM/InterPro
- SAE feature classification/interpretation is extremely time consuming and requires lots of expert labor to look at highlighted PyMOL rendered PNGs and generate labels. Could perhaps use VLMs (vision language models) or crowdsource
- The SAE feature scoring could be improved
    - i.e. Per-variant probe weight, per-fold attribution, diff of means visualization

Future Work

- Extend to more datasets such as Virulence Factor Database (VFDB), NCBI Viral, or even new threat categories like toxicity prediction (ToxinPred3) and immune response (VaxiJen Series)
- Optimize SAE trainer parameters like dict_size, k, and group_fractions, and benchmark different SAE types
- Rerun experiments with more structures and bigger datasets, more layers
- Explore steering and guardrails during protein design
    - Prevention and runtime monitoring: if we notice RFDiffusion3 is producing latents indicating it's designing a potentially viral or toxic motif, we can terminate harmful diffusing trajectories.
    - Steering: We can steer away from viral/toxic directions in latent space to generate safer proteins
- Use SAE activations like sparse embeddings to screen large datasets of sequences, as demonstrated by data centric interpretability work like Jiang et al and HypotheSAEs
- Extend the SAE feature database to build something like Neuronpedia but for biology models
- Cleanup the codebase and make documentation

6. Conclusion

We present **SAEBER**, the first application of sparse autoencoders to all-atom protein design and folding models. Our results demonstrate that interpretable, hazard-associated features can be

extracted from the internal representations of RFD3 and RF3, and that probes trained on these features achieve competitive (though not state-of-the-art) virulence classification, with the strongest signal emerging from deeper layers of the diffusion transformer. Beyond classification, the SAE feature dictionary provides a base for explanation, steering, and runtime monitoring of harmful design trajectories which are capabilities that pure sequence- or structure-similarity screening methods don't provide.

While the small dataset, narrow length range, and ad-hoc layer selection limit the strength of our quantitative claims, we find that virulence-related features exist and this qualitative finding is robust across hookpoints and clustering schemes. We see SAEBER as an early but encouraging step toward interpretability-based biosecurity for generative protein models, and we hope future work will scale the approach to larger datasets, more threat categories, and active interventions during the design process itself.

Code and Data

Include links if applicable. If your project doesn't involve code (e.g., policy analysis) or if there are info-hazard considerations, note that here.

- ***Code repository**: [Under Construction]*
  - *https://github.com/michaelwaves/saeber*
  -
- ***Data/Datasets**:*
  - *Activations*
    - *https://huggingface.co/datasets/michaelwaves/saeber-rfd3-safeprotein-activations*
    - *https://huggingface.co/datasets/michaelwaves/saeber-rf3-safeprotein-activations*
  - *Probes*
    - *https://huggingface.co/datasets/michaelwaves/saeber-virulence-probes-clustered*
    - *https://huggingface.co/datasets/michaelwaves/saeber-virulence-probes-random*
    -
- *Models:*
  - *https://huggingface.co/michaelwaves/saeber-rf3-safeprotein-sae*
  - *https://huggingface.co/michaelwaves/saeber-rfd3-safeprotein-sae*
  - *https://huggingface.co/michaelwaves/saeber-rf3-toxinpred3-sae*
  -
- ***Website**: https://www.raft.bio/blog/saeber*

References

@inproceedings{
fan2025safeprotein,
title={SafeProtein: Red-Teaming Framework and Benchmark for Protein Foundation Models},
author={Jigang Fan and Zhenghong Zhou and ZAIXI ZHANG and Ruofan Jin and Le Cong and Mengdi Wang},
booktitle={NeurIPS 2025 Workshop on Biosecurity Safeguards for Generative AI},
year={2025},
url={https://openreview.net/forum?id=rAlReAI7bx}
}

@InProceedings{pmlr-v267-bussmann25a,
  title = {Learning Multi-Level Features with Matryoshka Sparse Autoencoders},
  author = {Bussmann, Bart and Nabeshima, Noa and Karvonen, Adam and Nanda, Neel},
  booktitle = {Proceedings of the 42nd International Conference on Machine Learning},
  pages = {6077--6101},
  year = {2025},
  editor = {Singh, Aarti and Fazel, Maryam and Hsu, Daniel and Lacoste-Julien, Simon and Berkenkamp, Felix and Maharaj, Tegan and Wagstaff, Kiri and Zhu, Jerry},
  volume = {267},
  series = {Proceedings of Machine Learning Research},
  month = {13--19 Jul},
  publisher = {PMLR},
  pdf = {https://raw.githubusercontent.com/mlresearch/v267/main/assets/bussmann25a/bussmann25a.pdf},
  url = {https://proceedings.mlr.press/v267/bussmann25a.html},
  abstract = {Sparse autoencoders (SAEs) have emerged as a powerful tool for interpreting neural networks by extracting the concepts represented in their activations. However, choosing the size of the SAE dictionary (i.e. number of learned concepts) creates a tension: as dictionary size increases to capture more relevant concepts, sparsity incentivizes features to be split or absorbed into more specific features, leaving high-level features missing or warped. We introduce Matryoshka SAEs, a novel variant that addresses these issues by simultaneously training multiple nested dictionaries of increasing size, forcing the smaller dictionaries to independently reconstruct the inputs without using the larger dictionaries. This organizes features hierarchically - the smaller dictionaries learn general concepts, while the larger dictionaries learn more specific concepts, without incentive to absorb the high-level features. We train Matryoshka SAEs on Gemma-2-2B and TinyStories and find superior performance on sparse probing and targeted

concept erasure tasks, more disentangled concept representations, and reduced feature absorption. While there is a minor tradeoff with reconstruction performance, we believe Matryoshka SAEs are a superior alternative for practical tasks, as they enable training arbitrarily large SAEs while retaining interpretable features at different levels of abstraction.}
}

@article{
doi:10.1126/science.adu8578,
author = {Bruce J. Wittmann and Tessa Alexanian and Craig Bartling and Jacob Beal and Adam Clore and James Diggans and Kevin Flyangolts and Bryan T. Gemler and Tom Mitchell and Steven T. Murphy and Nicole E. Wheeler and Eric Horvitz },
title = {Strengthening nucleic acid biosecurity screening against generative protein design tools},
journal = {Science},
volume = {390},
number = {6768},
pages = {82-87},
year = {2025},
doi = {10.1126/science.adu8578},
URL = {https://www.science.org/doi/abs/10.1126/science.adu8578},
eprint = {https://www.science.org/doi/pdf/10.1126/science.adu8578},
abstract = {Advances in artificial intelligence (AI)–assisted protein engineering are enabling breakthroughs in the life sciences but also introduce new biosecurity challenges. Synthesis of nucleic acids is a choke point in AI-assisted protein engineering pipelines. Thus, an important focus for efforts to enhance biosecurity given AI-enabled capabilities is bolstering methods used by nucleic acid synthesis providers to screen orders. We evaluated the ability of open-source AI-powered protein design software to create variants of proteins of concern that could evade detection by the biosecurity screening tools used by nucleic acid synthesis providers, identifying a vulnerability where AI-redesigned sequences could not be detected reliably by current tools. In response, we developed and deployed patches, greatly improving detection rates of synthetic homologs more likely to retain wild type–like function. One way in which companies and governments can limit potentially harmful biological research is by restricting the commercial synthesis of DNA that encodes particular proteins or is from select pathogens. However, screening methods are not necessarily designed to detect engineered sequences. Wittmann et al. worked with four commercial DNA synthesis companies to stress test and develop patches for screening methods to greatly improve their ability to identify sequences that should be restricted. Their results demonstrate that proteins of concern can be flagged by updated software even if their sequences have been altered using protein design methods. —Michael A Funk}}

LLM Usage Statement

We used Claude Code to implement most results. All results and claims were verified by me and Claude.