Do biological autoregressive models learn protein coevolution?

Xiaonan Liu
Department of Biostatistics
Harvard University
xiaonan_liu@g.harvard.edu

Zhidian Zhang
Department of Biology
Massachusetts Institute of Technology
zhidianz@mit.edu

Jeffrey A. Ruffolo
Profluent Bio
jruffolo@profluent.bio

Sergey Ovchinnikov
Department of Biology
Massachusetts Institute of Technology
so3@mit.edu

Abstract

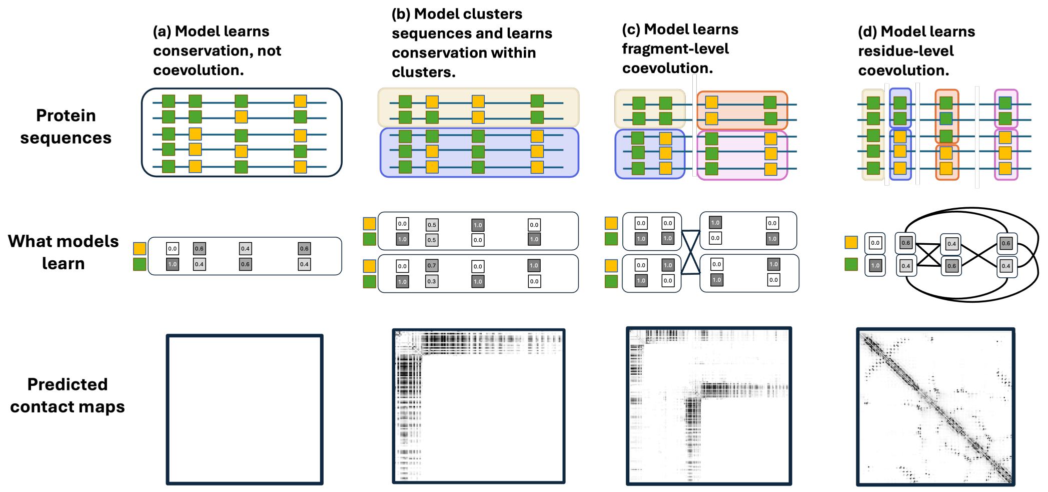
Autoregressive protein and genomic language models have shown remarkable success in designing proteins. However, it is unclear whether they, like masked protein language models, ESM2, have also learned protein coevolution. We applied the unsupervised Categorical Jacobian method to extract protein coevolutionary contacts across 105 proteins from ProGen3, Evo and Evo2. All autoregressive models demonstrated substantially lower contact recovery compared to the masked langauge model ESM2 (model with the best performance: ProGen3 P@L=0.077 vs ESM2=0.450). While scaling improved contact detection, even the largest model, ProGen3 46B, performed better than random in only 31 out of 105 proteins. We hypothesize that autoregressive models primarily learn dependencies between sequence fragments rather than residue-level coevolution, with fragments shorten as models scale. This limitation may explain why current autoregressive models often require generating millions of sequences followed by extensive post hoc filtering to obtain viable designs and why their generations have high sequence identity to the natural proteins, limiting their degree of novelty. Our findings provide insights into how autoregressive models generate successful designs and suggest directions for improving novel sequence generation.

1 Introduction

Language models have become increasingly popular in biological research. Two prominent examples are protein langauge models (plm) which learn from amino acid sequences and genomic language models (glm) which learn from nucleic acid sequences. Although these models enable a variety of predictive tasks, a major emerging application is generative design, with the aim to generate novel biological sequences with desired properties.

Among the various training paradigms, autoregressive (AR) modeling has been widely adopted for generative purposes due to its efficiency and flexibility. Unlike masked language models, which predict randomly masked tokens, AR models predict each token sequentially based on the preceding context, enabling generation of variable-length sequences through sequential forward passes. AR models have demonstrated practical success in protein design. For example, ProtGPT2 Ferruz et al. [2022] is capable of generating protein sequences that are distantly related to natural sequences, ProGen models Madani et al. [2023]Ruffolo et al. [2024]Bhatnagar et al. [2025] produced diverse proteins that were expressed in vitro, Evo Nguyen et al. [2024] generated CRISPR-Cas9 complexes, Evo2 Brixi et al. [2025]King et al. [2025] generated viable bacteriophage genomes. Previous work has shown that masked plm, ESM2 Lin et al. [2023], though trained using single protein sequences, stores statistics of coevolving residues Zhang et al. [2024]. In contrast, preliminary analyses of Evo Nguyen et al. [2024], using categorical jacobian, suggested that the model may not capture protein coevolution, despite its success in generating protein designs. This paradox raises the question: do these AR models learn protein coevolution, or what patterns enabled them to generate valid designs?

Here we outline four hypotheses describing patterns these models may have learned (Figure 1). (a) At the simplest level, models may learn only the conservation profile of a protein family, without capturing any residue-residue interactions, leading to a blank predicted contact map. (b) A model may learn to cluster sequences and capture conservation patterns within each cluster. In this case, first few positions could determine the cluster assignment and perturbing at those positions would influence the downstream regions, creating stripe patterns (horizontal or vertical lines) in contact maps. Interestingly, models that learn to cluster sequences and sample from the site-independent probabilities of each cluster may still produce sequences that appear to capture pairwise dependencies, for example if the covarying residues are completely conserved in each cluster, as illustrated in figure 1b. (c) A more advanced model may capture fragment-level coevolution, capturing dependencies between contiguous fragments of the sequences, hence leading to multiple stripes in contact maps. (d) Finally, models may learn residue-level coevolution where the predicted contact map should resemble the experimental structural contacts. Guided by these hypotheses, we investigated the latest models, ProGen3, Evo, and Evo2 to determine what patterns AR models learn.



(a) Model learns conservation, not coevolution.
(b) Model clusters sequences and learns conservation within clusters.
(c) Model learns fragment-level coevolution.
(d) Model learns residue-level coevolution.
Protein sequences
What models learn
Predicted contact maps

Figure 1: Hypotheses describing progressively richer patterns AR models may have learned. Green and yellow boxes indicate protein residues. Numbers indicate conservation probabilities of residues. Predicted contact maps show what contact maps extracted from AR models would look like.

2 Methods

2.1 Categorical Jacobian

To assess whether language models capture coevolutionary signals, we used Categorical Jacobian (CatJac) Zhang et al. [2024], an unsupervised approach that allows direct comparison across models regardless of architectures. CatJac, denoted as $\mathbf{J}$ is computed as follows: For a protein sequence of length L, we mutate each residue to all 20 possible amino acids, and measure how each of these mutations perturbs the model output (i.e. logits) at every position. The logits, which represent the predicted amino acid probabilities, have shape $L \times 20$. Consequently, $\mathbf{J}$ has a shape $L \times 20 \times L \times 20$, with each entry describing how a mutation at one position impacts the predicted distribution at another. After normalization and average product correction (APC), $\mathbf{J}$ is reduced to a predicted contact map of shape $L \times L$ (see Methods).

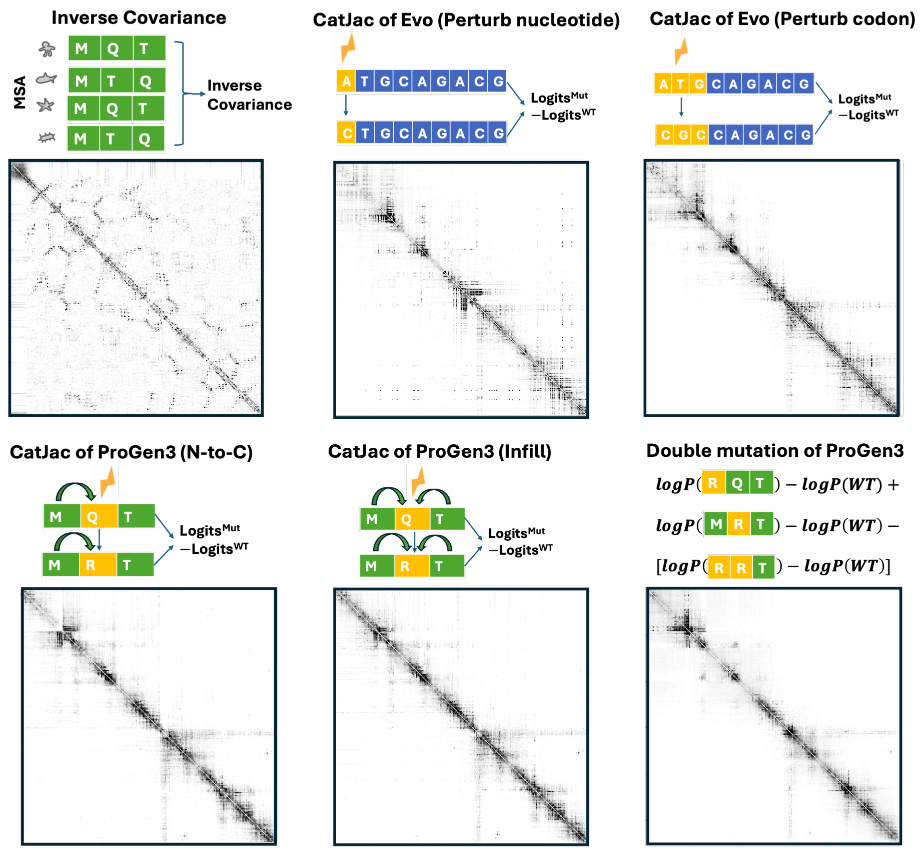


Figure 2: Illustration of CatJac methods and results for Evo (7B) and ProGen3 (3B), using protein 5CXX as an example. Green denotes protein sequences, blue denotes DNA sequences, and yellow indicates the perturbed site. The inverse covariance from the multiple sequence alignment (MSA) is shown as a reference for residue-level coevolutionary information. Mut: mutation, WT: wild-type.

2.2 CatJac for Evo

CatJac was originally developed for masked plm trained on amino acid sequences, whereas Evo is a glm that tokenizes at the nucleotide level. Since no established approach exists for extracting protein contact directly from nucleotide-based models, we adapted the CatJac framework for glm using two strategies: perturb-nucleotide and perturb-codon. In the perturb-nucleotide approach, each position of the DNA sequence is mutated to all four nucleotides $\{A, C, G, T\}$, and the resulting changes in model outputs are used to infer residue-residue interaction (see Methods). However, since protein contacts are usually obtained from perturbation at the amino acid level in plm, we developed a perturb-codon strategy such that every three positions were perturbed to all possible codons to allow for extracting residue-level coevolution more effectively (see Methods, Figure 2).

2.3 CatJac for ProGen3

For the AR plm, we first computed the standard CatJac for sequences in N-to-C direction (NC CatJac). While CatJac was originally designed for masked language models that use bidirectional context, AR models only use preceding context. This might create a potential bias because later positions see more contexts than earlier positions, causing CatJac more prone to detect coevolutionary signals towards the C-terminal. To mitigate this positional bias, we implemented two additional methods: 1) tokenize sequence using infilling to mimic masked language models (infill CatJac) (see Methods), and 2) the double mutation approach Trinquier et al. [2021], designed specifically for AR models where it compares the effect of double mutations with the sum of the effects of the single mutations, when introduced separately into the wild-type sequence (see Methods, Figure 2).

2.4 Evaluation of contact recovery

We obtained 651 monomer structures from the Generative Regularized ModeLs of proteINs (GREMLIN) database Kamisetty et al. [2013]. Then we extracted 169 proteins with available DNA coding sequences from the GenBank and kept 105 proteins that have low structural similarity (TMalign score < 0.5). To evaluate the accuracy of predicted protein contacts, we computed P@L for each protein, defined as the proportion of top L predicted contacts (with residues ≥ 6 positions apart) that overlap with the experimental contacts obtained from the Protein Data Bank (PDB): $P@L = \frac{1}{L}\sum_{(i,j)\in P_L} \mathbb{1}[(i,j) \in T]$ where P_L denotes the indices of the top L predicted contacts values among pairs satisfying $j - i \geq 6$, T denotes the indices of experimental contacts. To determine whether P@L was significantly greater than 0, we performed permutation tests by randomly shuffling each row of experimental contacts for 9,999 permutations. We further adjusted for multiple testing using a Bonferroni correction, resulting in a significance threshold of 0.05/105.

3 Results

3.1 AR models learn limited residue-level coevolution

We first assessed different CatJac variants for Evo and ProGen3 to identify the most efficient method for large-scale analysis. For ProGen3, we computed NC CatJac, infill CatJac and double mutation methods on 10 randomly selected proteins due to computational constraints. NC, infill CatJac, and double mutation all performed similarly. Hence, we selected the computationally efficient NC CatJac for analysis in all 105 proteins. For Evo, we computed perturb nucleotide and codon methods on the same proteins; neither revealed contacts. We used the more efficient perturb nucleotide method for the full analysis. Figure 2 shows the results for a representative protein 5CXX and the predicted contact maps of all 10 proteins are shown in Figures 7 and 8.

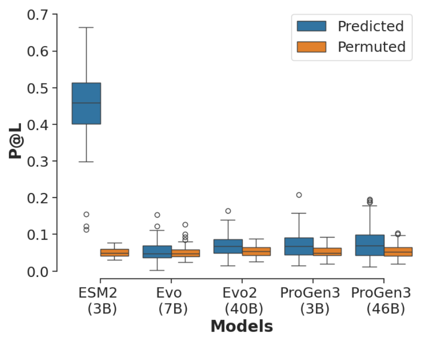


Figure 3: P@L values of predicted contacts (blue) and permuted contacts (orange) across 105 proteins.

Then to systematically assess whether language models capture residue-level coevolution, we analyzed predicted contacts across 105 proteins using ESM2(3B parameters), ProGen3 (3B and 46B), Evo (7B), and Evo2 (40B) models where ESM2 was used as the representative masked plm with good protein contact recovery. Figure 3 shows P@L computed using predicted contacts versus experimental contacts (blue) and versus permuted experimental contacts as a control (orange). ESM2 achieved the highest P@L with a mean of 0.450 and showed statistically significant contact detection across all proteins. In contrast, all AR models showed substantially low P@L, approaching the control, indicating poor contact recovery. Among the AR models, plms performed marginally better than glms (Figure 5). Performance ranking from the lowest to highest was: Evo (mean P@L=0.053, number of proteins with significant contact detection, n_sig=10), Evo2 (mean P@L=0.070, n_sig=21), ProGen3 (3B) (mean P@L=0.073, n_sig=24), and ProGen3 (46B) (mean P@L=0.077, n_sig=31).

3.2 AR models learn fragment-level protein coevolution

Across all AR models, the predicted contact maps exhibited distinct striped patterns indicating that perturbing a single position influenced the residue distribution throughout the contiguous fragment instead of particular single residues. This pattern suggests that AR models primarily capture fragment-level dependencies rather than residue-level coevolution. Model scaling improved contact recovery modestly (Figure 5). With the CatJac method, ProGen3 (46B) shows consistently less contact recovery than the much smaller masked language model ESM2 (3B). Notably, in some proteins, striped patterns diminished as model scales, revealing residue-level coevolutionary signals. To illustrate scaling effects, we present protein 1PQ4 as an example (Figure 4). As ProGen3 scales from 112M to 762M parameters, the predicted contact maps begin to separate domains of the protein, and

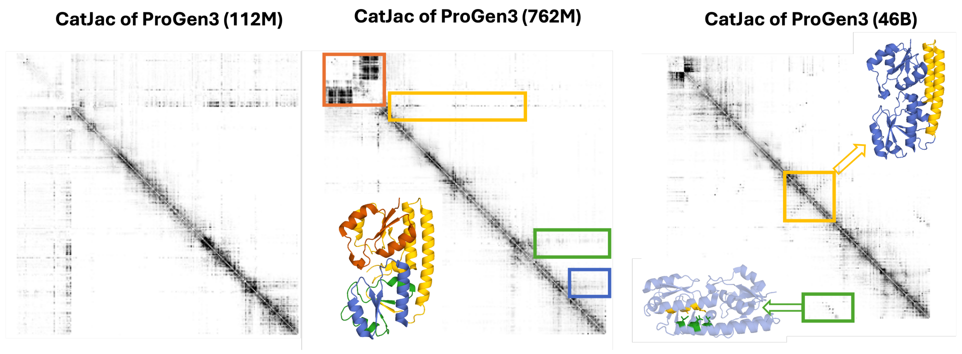


Figure 4: Scaling improved contact recovery for AR models. From left to right, contact maps show CatJac of protein 1PQ4 using ProGen3 models of increasing size: 112M, 762M, 46B parameters.

with further scaling to 46B parameters, the model recovers additional residue-level contacts. Full results, including the inverse covariance, CatJac of ESM2, Evo and Evo2 are shown in Figure 6.

4 Discussion

In this paper, we systematically evaluated the ability of AR glm and plm to extract protein coevolution using the Categorical Jacobian framework. Our analyses revealed that all AR models showed minimal residue-level contact recovery which were substantially lower than that of masked plm. This raises important questions: How do these models generate successful protein designs and what underlying patterns do they learn?

One hypothesis is that AR models do not learn protein coevolution and their successful designs arise from large-scale generation with extensive post hoc filtering. For example, Evo generated more than two million sequences of the CRISPR-Cas9 complex Nguyen et al. [2024], which were subsequently filtered down to 11 designs for experimental validation. This suggests that the viable designs may emerge from the massive sampling rather than an intrinsic understanding of protein coevolution. Furthermore, if generated sequences differ from natural ones only at highly variable positions, those positions are likely on the protein surface with weak coevolutionary constraints. Therefore, one could obtain functional proteins by chance even if models do not capture coevolution. This could explain the viability of some recent high sequence identity genome designs King et al. [2025].

An alternative hypothesis is that AR models primarily learn dependencies between sequence fragments, which is supported by our CatJac analyses. This aligns with recent observations that generations from AR models are often composed of fragments of natural sequences Naughton [2024]; and another observation that natural proteins contain reused fragments Ferruz et al. [2020], which explains why AR models would potentially learn these fragments first. As model scales up, these stripes shorten, indicating that the model is progressively learning smaller fragments and eventually approaching single-position resolution, thereby capturing coevolutionary signals of residues.

Finally, we highlight the importance for model to learn residue-level coevolution. First, it can make design generation more efficient. Models that understand residue-level coevolution could generate sequences that better recapitulate important residue-residue interactions in a structure, thereby reducing the need for extensive sampling and post-hoc filtering. In contrast, AR models that do not capture residue interactions might generate inconsistent mutations, hence requiring massive sampling and post hoc filtering to identify functional candidates. Second, learning coevolution could potentially increase sequence novelty. Current AR generated sequences often exhibit high sequence identity to natural proteins, whereas models that learn residue-level coevolution might explore new sequence combinations while maintaining residue dependencies. Thus, learning residue-level coevolution is crucial for generating both more efficient and more novel protein designs.

In conclusion, our results highlight a key limitation of current AR models that their limited ability to capture residue-level coevolution likely constraints both the efficiency and novelty of their designs. While continued scaling is one way toward generating more diverse sequences for AR models, as demonstrated in ProGen3 that sequence diversity increased with model size, more computationally efficient approaches are needed to enable practical and novel protein designs.

References

A. Bhatnagar, S. Jain, J. Beazer, S. C. Curran, A. M. Hoffnagle, K. Ching, M. Martyn, S. Nayfach, J. A. Ruffolo, and A. Madani. Scaling unlocks broader generation and deeper functional understanding of proteins. *bioRxiv*, pages 2025–04, 2025.

G. Brixi, M. G. Durrant, J. Ku, M. Poli, G. Brockman, D. Chang, G. A. Gonzalez, S. H. King, D. B. Li, A. T. Merchant, et al. Genome modeling and design across all domains of life with evo 2. *BioRxiv*, pages 2025–02, 2025.

S. Dunn, L. Wahl, and G. Gloor. Mutual information without the influence of phylogeny or entropy dramatically improves residue contact prediction. *Bioinformatics*, 24(3):333–340, 12 2007. ISSN 1367-4803. doi: 10.1093/bioinformatics/btm604. URL https://doi.org/10.1093/bioinformatics/btm604.

N. Ferruz, F. Lobos, D. Lemm, S. Toledo-Patino, J. A. Farías-Rico, S. Schmidt, and B. Höcker. Identification and analysis of natural building blocks for evolution-guided fragment-based protein design. *Journal of molecular biology*, 432(13):3898–3914, 2020.

N. Ferruz, S. Schmidt, and B. Höcker. Protgpt2 is a deep unsupervised language model for protein design. *Nature communications*, 13(1):4348, 2022.

H. Kamisetty, S. Ovchinnikov, and D. Baker. Assessing the utility of coevolution-based residue–residue contact predictions in a sequence- and structure-rich era. *Proceedings of the National Academy of Sciences*, 110(39):15674–15679, 2013. doi: 10.1073/pnas.1314045110. URL https://www.pnas.org/doi/abs/10.1073/pnas.1314045110.

S. H. King, C. L. Driscoll, D. B. Li, D. Guo, A. T. Merchant, G. Brixi, M. E. Wilkinson, and B. L. Hie. Generative design of novel bacteriophages with genome language models. *bioRxiv*, 2025. doi: 10.1101/2025.09.12.675911. URL https://www.biorxiv.org/content/early/2025/09/17/2025.09.12.675911.

Z. Lin, H. Akin, R. Rao, B. Hie, Z. Zhu, W. Lu, N. Smetanin, R. Verkuil, O. Kabeli, Y. Shmueli, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

A. Madani, B. Krause, E. R. Greene, S. Subramanian, B. P. Mohr, J. M. Holton, J. L. Olmos Jr, C. Xiong, Z. Z. Sun, R. Socher, et al. Large language models generate functional protein sequences across diverse families. *Nature biotechnology*, 41(8):1099–1106, 2023.

B. Naughton. [twitter post about novelty of profluent opencrispr protein], June 2024. URL https://x.com/btnaughton/status/1805732057827655905. Accessed: [Date you accessed it].

E. Nguyen, M. Poli, M. G. Durrant, B. Kang, D. Katrekar, D. B. Li, L. J. Bartie, A. W. Thomas, S. H. King, G. Brixi, et al. Sequence modeling and design from molecular to genome scale with evo. *Science*, 386(6723):eado9336, 2024.

J. A. Ruffolo, S. Nayfach, J. Gallagher, A. Bhatnagar, J. Beazer, R. Hussain, J. Russ, J. Yip, E. Hill, M. Pacesa, et al. Design of highly functional genome editors by modeling the universe of crispr-cas sequences. *Biorxiv*, pages 2024–04, 2024.

J. Trinquier, G. Uguzzoni, A. Pagnani, F. Zamponi, and M. Weigt. Efficient generative modeling of protein sequences using simple autoregressive models. *Nature communications*, 12(1):5800, 2021.

Z. Zhang, H. K. Wayment-Steele, G. Brixi, H. Wang, D. Kern, and S. Ovchinnikov. Protein language models learn evolutionary statistics of interacting sequence motifs. *Proceedings of the National Academy of Sciences*, 121(45):e2406285121, 2024.

A Methods

A.1 Categorical Jacobian

For a protein sequence, $X = (x_1, ..., x_L)$ where $x_i \in \{a_1, ..., a_{20}\}$. The CatJac (also referred as predicted contacts), $\mathbf{J}$ is given by

$$\mathbf{J} = \begin{bmatrix} f[X(x_1 \to a_1)] - f[X] & \dots & f[X(x_1 \to a_{20})] - f[X] \\ \vdots & \ddots & \vdots \\ f[X(x_L \to a_1)] - f[X] & \dots & f[X(x_L \to a_{20})] - f[X] \end{bmatrix}$$

where $f[X]$ denotes the model logits given X with shape $(L, 20)$, $f[X(x_i \to a_j)]$ denotes the logits using perturbed input where ith position of X changed to an amino acid a_j. The resulting CatJac has dimension $(L, 20, L, 20)$. After centering and normalization, we obtained a contact matrix $\hat{\mathbf{C}}$ with shape (L, L). To correct for background correlations, we applied average product correction (APC) Dunn et al. [2007], yielding the final predicted contact matrix $\hat{\mathbf{C}}^{\mathbf{APC}}$ with entries defined as:

$$\hat{C}_{ij}^{APC} = \hat{C}_{ij} - \frac{\sum_{i'=1}^{L} \hat{C}_{i'j} \sum_{j'=1}^{L} \hat{C}_{ij'}}{\sum_{i'=1}^{L} \sum_{j'=1}^{L} \hat{C}_{i'j'}}$$

A.2 CatJac for Evo – Perturb nucleotide

We perturb each position of the DNA sequence to all four nucleotides. Given a protein sequence X, we denote the corresponding DNA sequence as $S = (s_1, ..., s_{3L})$ with $s_t \in \{A, C, G, T\}$. The CatJac, $\mathbf{J_{nuc}}$ is given by

$$\mathbf{J_{nuc}} = \begin{bmatrix} f[S(s_1 \to n_1)] - f[S] & \dots & f[S(s_1 \to n_4)] - f[S] \\ \vdots & \ddots & \vdots \\ f[S(s_{3L} \to n_1)] - f[S] & \dots & f[S(s_{3L} \to n_4)] - f[S] \end{bmatrix}$$

where $f[S]$ denotes the logits of glm given S with shape $(3L, 4)$. $f[S(s_t \to n_j)]$ means the logits of mutated input where tth position change to a nucleotide n_j. We reshaped $\mathbf{J_{nuc}}$ from $(3L, 4, 3L, 4)$ to $(L, 12, L, 12)$ before applying the transformations above to obtain a (L, L) contact map.

A.3 CatJac for Evo – Perturb codon

We first transformed Evo's nucleotide probabilities $P(s_t|s_{<t})$ at position t to codon probabilities. For the ith codon $c_i = (s_{3i-2}, s_{3i-1}, s_{3i})$, we compute: $P(c_i|c_{<i}) = P(s_{3i-2}|s_{<3i-2})P(s_{3i-1}|s_{<3i-1})P(s_{3i}|s_{<3i})$. For computational efficiency, we approximate this by assuming conditional independence of nucleotides within codon. Then we perturbed each codon to the corresponding codons of 20 amino acids and computed CatJac $\mathbf{J_{codon}}$ as follows:

$$\mathbf{J_{codon}} = \begin{bmatrix} f(S[s_{1:3} \to codon_{a_1})] - f[S] & \dots & f[S(s_{1:3} \to codon_{a_{20}})] - f[S] \\ \vdots & \ddots & \vdots \\ f(S(s_{3L-2:3L} \to codon_{a_1})) - f[S] & \dots & f(S(s_{3L-2:3L} \to codon_{a_{20}})) - f[S] \end{bmatrix}$$

where $f[S]$ denotes the log probability given S with shape $(L, 20)$ and $codon_{a_j}$ denotes the codon for amino acid a_j. We chose a random set of codon for each amino acid and used it to compute codon probability and CatJac.

A.4 ProGen3 CatJac Infill

ProGen3 has infilling capabilities that allow it to fill in segments in the middle of a protein sequence. When an infilling span is introduced in the middle of a sequence, the model processes all other positions autoregressively. We therefore use infilling as a way to mimic masked langauge model in an autoregressive setting. We tokenize the sequence such that the infilling span serves as the ”masked token”. Our goal is to obtain an output where row i represents the masked probability of position i given all other positions. For example, given a protein sequence, ”MQTI”, the first row in a masked language model output would be $P(a_j|QTI)$ where $j = 1, ..., 20$. To approximate this, we tokenize the sequence as `<bos_glm>1<span>QTI2<EOS><span>M<EOS_span>`. The autoregressive model then allows us to approximate the masked probability as $P(a_j|QTI) \approx P(a_j| < bos\_glm > 1 < span > QTI2 < EOS >< span >)$ for $j = 1, ..., 20$. Following the same logic, we introduce infilling spans at other positions to reconstruct the full model output and compute CatJac as usual.

A.5 ProGen3 double mutation

Let $E(x_1, ..., x_L) = -logP(x_1, ..., x_L)$ be the statistical energy of a sequence X. For a single mutation $x_i \rightarrow a_j$ where position i is substituted with amino acid a_j, we determine the statistical energy difference as:

$$\Delta E(x_i \rightarrow a_j) = -log\frac{P(x_1, ..., x_{i-1}, a_j, x_{i+1}, ..., x_L)}{P(x_1, ..., x_{i-1}, x_i, x_{i+1}, ..., x_L)}$$

To predict contact, we compute the epistasis effect between two positions as follows: $\Delta E(x_i \rightarrow a_j, x_{i^*} \rightarrow a_{j^*}) - \Delta E(x_i \rightarrow a_j) - \Delta E(x_{i^*} \rightarrow a_{j^*})$. For each pair of positions, we mutate to all 20×20 possible combinations, yielding a output of dimension $(L, 20, L, 20)$. Then one can apply the same normalization and APC correction as above to obtain the predicted contact.

B Differences in architectures of models

ESM2 uses a BERT style encoder-only transformer architecture trained on single protein sequences with a masked language model (MLM) objective. The model minimizes the loss:

$$L_{MLM} = -\sum_{i \in M} logp(x_i|x_{\backslash M})$$

where M represents a randomly chosen set that includes 15% of positions i in the sequence x, and the model was trained to predict the masked amino acids x_i from the context $x_{\backslash M}$.

In contrast, ProGen3 uses a decoder-only transformer architecture trained on full length protein sequences. In addition, it replaces all feed-forward layers with a Mixture of Experts (MoE) to introduce sparsity and increase efficiency. ProGen3 was trained with an autoregressive objective, minimizing

$$L_{AR} = -\sum_{i=1}^{n} logp(x_i|x_{<i})$$

where the model predicts each amino acid x_i based on preceding residues $x_{<i}$. The training supports generation from N-to-C terminal, C-to-N terminal, or infilling where a segment in the middle of the protein can be generated while keeping both prior and subsequence amino acids fixed.

Evo was trained on prokaryotic and phase genome using the StrippedHyena architecture which combines hyena layers interleaved with multihead attention equipped with rotary position embeddings (RoPEs). Evo2 expanded training to include eukaryotic genomes and adopted the StrippedHyena2 architecture, an improved variant that enhances training efficiency at scale on both short and long genomic sequences.

C Figures

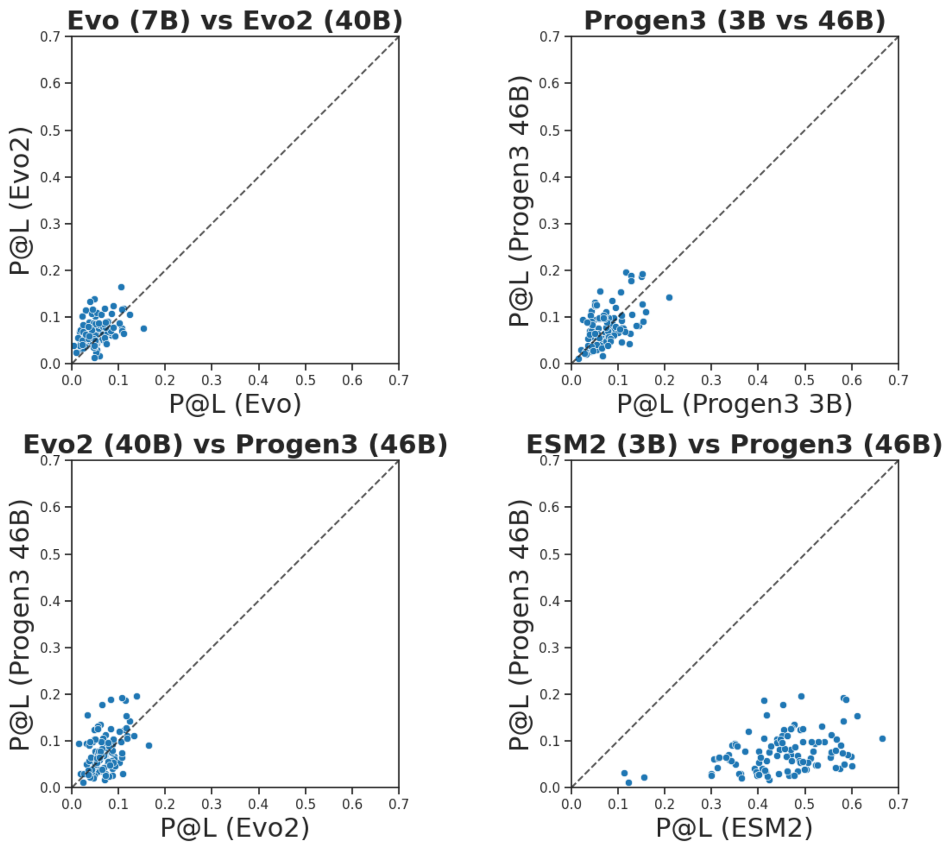


Figure 5: Pair-wise comparison of P@L between models across 105 proteins. Each dot represents a protein.

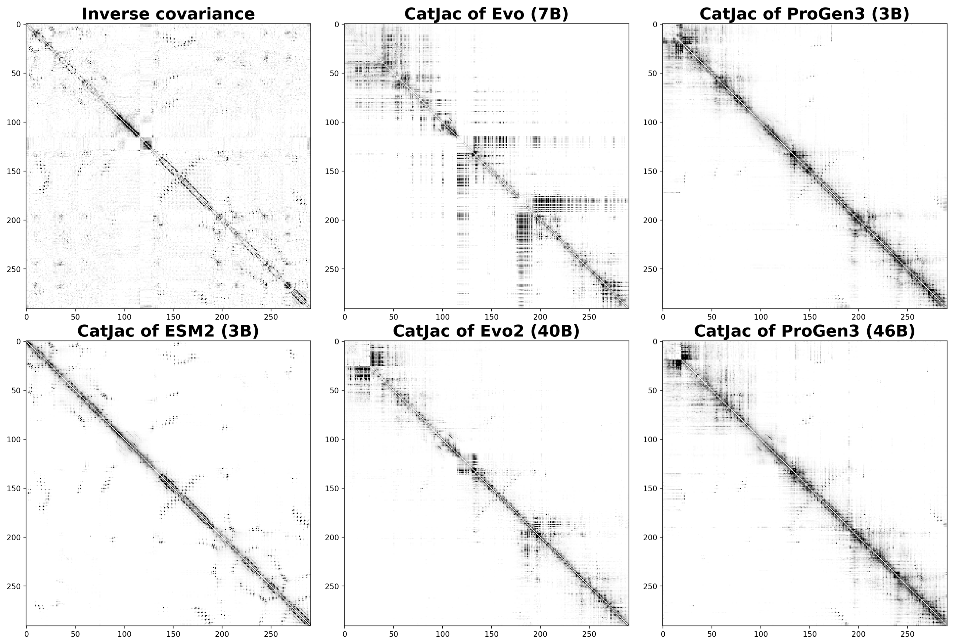


Figure 6: Left top shows inverse covariance computed from the multiple sequence alignment (MSA) of 1PQ4. Left bottom shows the normalized/APC corrected CatJac computed using ESM2. Middle panel shows that using Evo and Evo2. Right panel shows that using ProGen3.

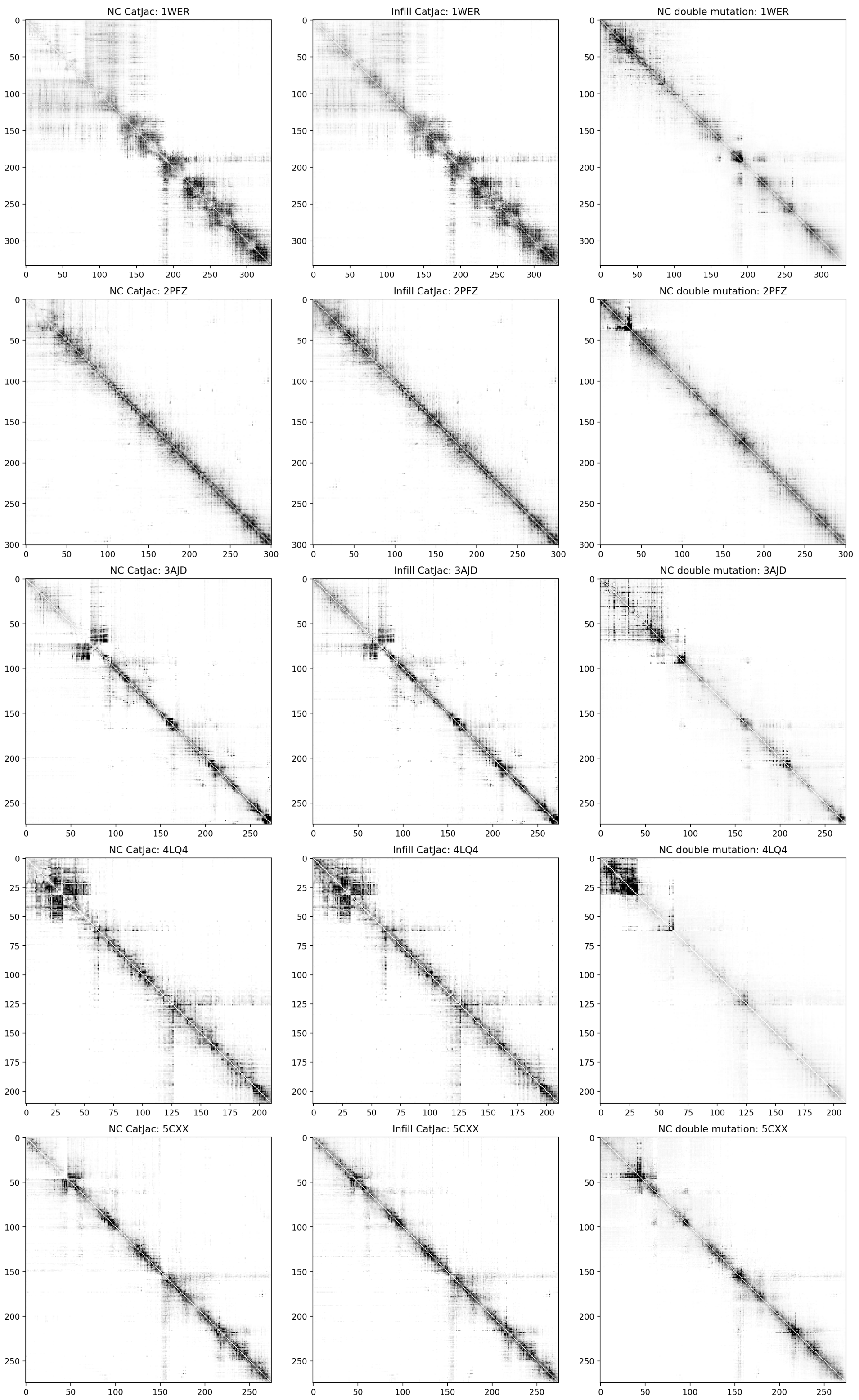


Figure 7: Categorical Jacobian of ProGen3 (3B) using methods: NC CatJac, infill CatJac and double mutation.

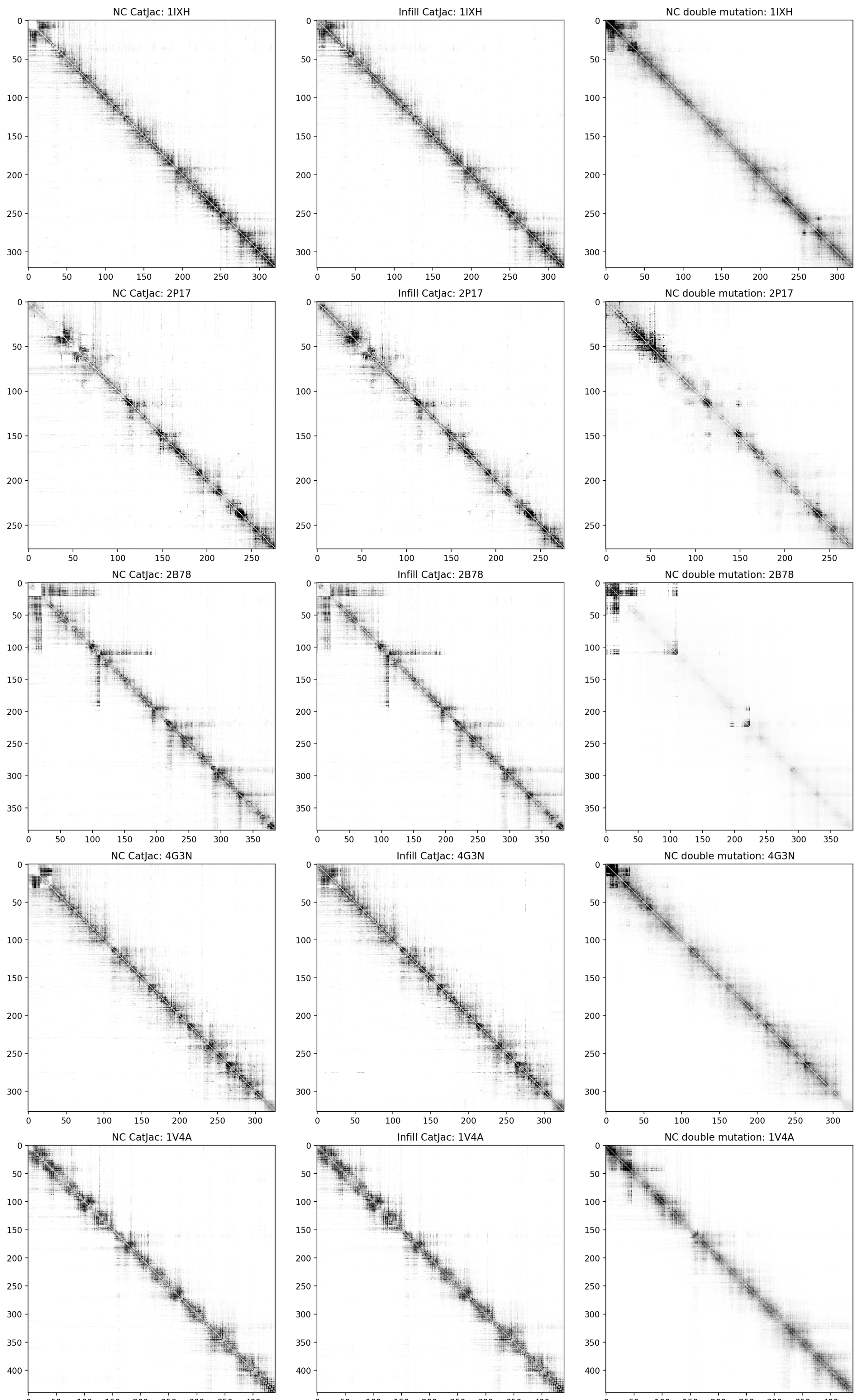


Figure 7: Categorical Jacobian of ProGen3 (3B) using methods: NC CatJac, infill CatJac and double mutation.(continued)

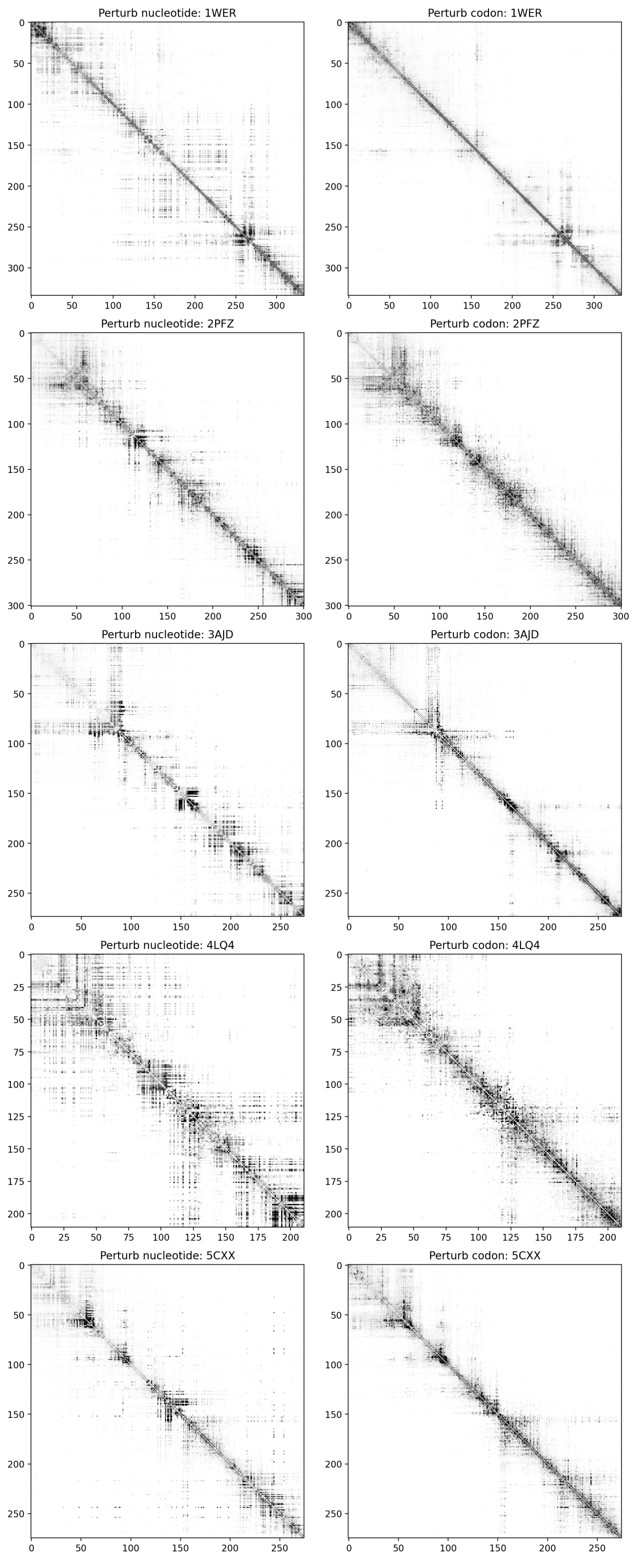


Figure 8: Categorical Jacobian of Evo using methods: perturb nucleotide and perturb codon.

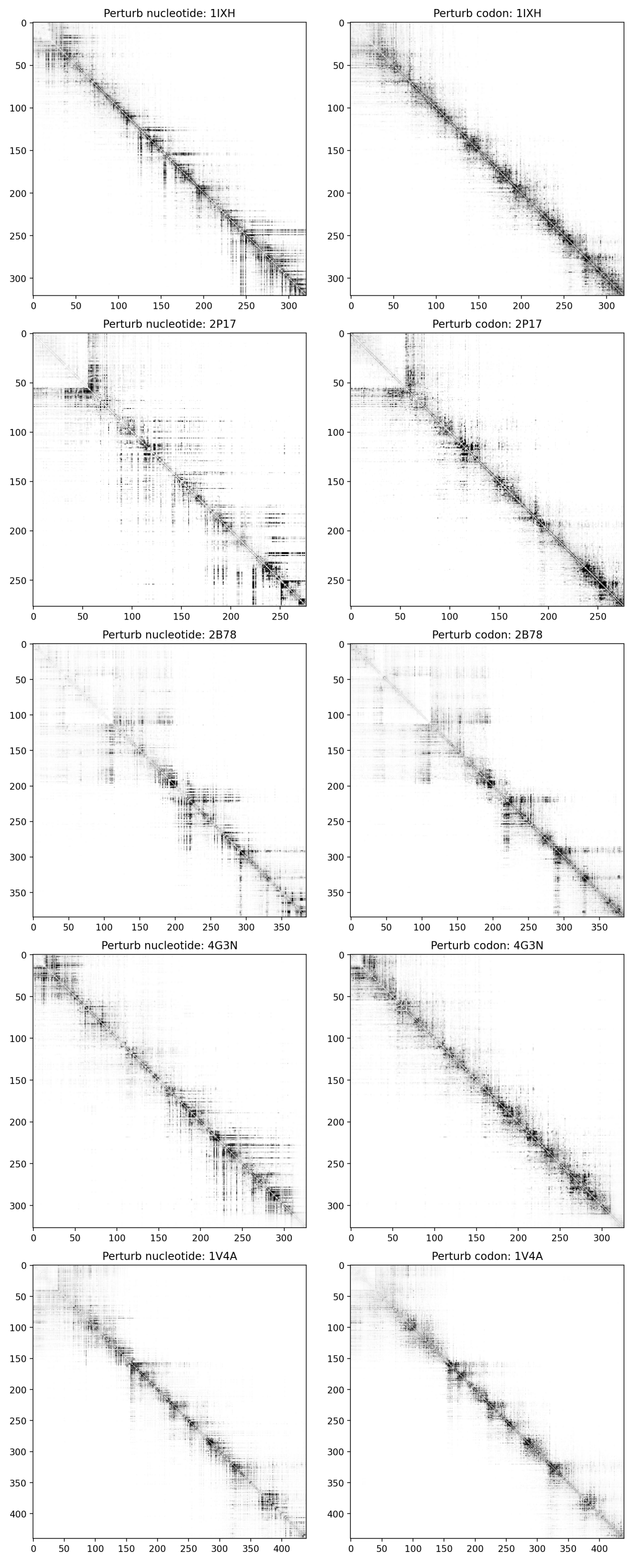


Figure 8: Categorical Jacobian of Evo using methods: perturb nucleotide and perturb codon.(continued)