UNDER REVIEW

RESCIENCE C

Reproduction / Computational Biology

[Re] Robust deep learning–based protein sequence design using ProteinMPNN

Christopher Huang[1], **Ryan Ahn**[2], **and Wenhao Lu**[3]

[1]Heritage High School, California, United States – [2]Yongsan International School of Seoul, Seoul, South Korea – [3]Millburn High School, Millburn, NJ, United States

Edited by
(Editor)

Received
–

Published
–

DOI
–

Abstract We reproduce the core computational results of ProteinMPNN, a message-passing neural network for protein sequence design from backbone structure. Using the official code and pretrained weights on 30 curated PDB structures, we recover the paper's qualitative claims: full-backbone design outperforms C-alpha-only design at every sampling temperature (46.4% vs. 41.3% recovery at T=0.1), recovery falls and perplexity rises monotonically with temperature, and recovery is higher at buried than exposed positions. Absolute recovery is below the reported 52.4%, which we attribute to dataset mismatch rather than implementation divergence. Beyond the original paper, we find that the full-backbone and CA-only models cross over under Gaussian coordinate noise near 0.45 A, so the weaker model on clean structures becomes the more robust one on badly corrupted input; that no such crossover occurs under residue masking; that four of six deliberately constructed malformed inputs fail silently; and that Apple Silicon GPU acceleration cannot run the model at all. Structure-prediction validation with ESMFold shows three of five designs folding to within 1.0 A of the native backbone at only 32-55% sequence identity.

A reproduction of J. Dauparas et al. Robust deep learning–based protein sequence design using ProteinMPNN. Science 378, 49-56 (2022).

1 Introduction

The protein sequence design problem, also called inverse folding, asks for an amino acid sequence that will fold into a given backbone structure[1]. For two decades the dominant approach was physically based sidechain packing and energy minimisation, most prominently Rosetta[2]. ProteinMPNN[3] replaced that with a graph-based message-passing neural network that reads backbone coordinates and emits per-position amino acid probabilities autoregressively, building on an earlier structure-conditioned generative model[4]. The reported result is striking on two axes at once: 52.4% native sequence recovery against Rosetta's 32.9%, at roughly 1.2 seconds versus 4.3 minutes for a 100-residue chain. The paper backs this with X-ray crystallography, cryo-EM and functional assays, and ProteinMPNN has since become a standard component in de novo design pipelines[5].

We reproduce the computational core of that paper. Our interest is not only whether the numbers come out, but whether the method is genuinely usable by people without a wet lab, a cluster, or an industrial curation pipeline, which is the condition most students and small groups actually work in. The original paper makes a strong implicit accessibility claim: the released checkpoints are 1.66M parameters and 6.7 MB, which is three to four orders of magnitude smaller than contemporaneous structure models. If that claim holds up, ProteinMPNN is one of the few frontier-adjacent biological models a high school student can run on a laptop. We test it under exactly those conditions, and then try to break it.

Correspondence should be addressed to Wenhao Lu (wenhao@nxthorizon.org)
The authors have declared that no competing interests exist.
Code is available at https://github.com/AIscend-Research/protein-repro.

1.1 Scope: what we reproduce and what we do not

The paper's headline claims are a mixture of computational and experimental results, and the experimental half (crystal structures, cryo-EM maps and binding assays) is not reproducible by us at any budget. We state our target plainly rather than burying it in a limitations section: we reproduce the *computational design pipeline*. That is, official installation, pretrained inference, sequence recovery, scoring, and behaviour under perturbation. We make no claim about experimental success rates, and we did not attempt any.

Within that scope we report three kinds of outcome, and keep them separate throughout:

1. **Reproduced.** Qualitative claims that held: full-backbone design beats Cα-only design; recovery and perplexity trade off monotonically against sampling temperature; recovery is higher at buried than exposed positions; the pipeline installs and runs on commodity hardware.
2. **Not reproduced numerically.** Absolute sequence recovery is about six percentage points below the published figure. We give the most likely reason and are explicit that we could not verify it directly.
3. **New.** Findings that go beyond the original paper, chiefly a robustness crossover between the two model variants under coordinate noise, an asymmetry between noisy and missing structural context, and a set of silent failure modes in the input-handling code.

2 Methods

2.1 Model and checkpoints

We used the official implementation at commit `8907e6671bfbfc92303b5f79c4b5e6ce47cdef57`, unchanged across every experiment and verified with `git rev-parse HEAD` before each phase. No component of the model or the parser was reimplemented.

The repository ships three families of pretrained weights: four full-backbone ("vanilla") checkpoints, four soluble-protein checkpoints, and three Cα-only checkpoints, differing in the amount of Gaussian noise added to backbone coordinates during training. We used `v_48_020`, with 48 neighbours and 0.20 Å training noise, for *both* the full-backbone and the Cα-only model. This is a deliberate methodological choice: the Cα-only weights ship in three noise variants, and pairing a full-backbone model against a differently-noised Cα-only model would confound the architecture comparison with a training-noise comparison.

Model	Checkpoint	Params	Size	Train noise
Full-backbone	`v_48_020` (vanilla)	1.66M	6.68 MB	0.20 Å
Cα-only	`v_48_020` (Cα-only)	1.65M	6.62 MB	0.20 Å

Table 1. Pretrained checkpoints used throughout. All eleven shipped weight sets were verified present and loadable before evaluation began.

2.2 Dataset

We curated 30 structures from the Protein Data Bank[6], spanning 36–627 resolved residues, X-ray and NMR determinations, monomers and oligomers, and both fully-resolved and naturally incomplete backbones:

```
1AKI 1BPI 1CRN 1CSP 1ENH 1FKB 1HEL 1IGD 1LZ1 1MBN 1PIN 1R69 1RIS 1SHG 1STN
1TIM 1TUP 1UBI 1UBQ 1VII 2ACY 2CI2 2CRO 2GB1 2LZM 2PCY 2PTL 3CHY 3PGK 4HHB
```

The set splits 27 monomers / 3 multimers, and 21 "clean" structures (no missing or gapped residues) / 9 "stress-test" structures with naturally unresolved or internally gapped residues (1PIN, 1R69, 1RIS, 1SHG, 1STN, 1TIM, 1TUP, 2CI2, 2CRO). We stress that this stress-test subset is *naturally occurring* incompleteness, as distinct from the synthetic corruption introduced later.
Parsing used the repository's own bundled chain-parsing and chain-selection helper scripts. We wrote no custom PDB parser, since divergent parsing is among the most common silent causes of failed reproductions. Structural metadata that the official parser does not expose (missing-residue counts, resolution, chain composition) was computed independently with Biopython[7].
Two dataset properties deserve to be stated rather than quietly fixed.

Two accidental duplicate identities — A post-hoc sequence-identity check found that 1AKI and 1HEL are both hen egg-white lysozyme (129 aa, exact match) and that 1UBI and 1UBQ are both human ubiquitin (76 aa, exact match). Each pair consists of separate legitimate PDB depositions, selected independently while assembling a list of classic small test structures. The set therefore contains 28 unique sequences, not 30. We left this in place rather than re-curating after inference had run, which would have invalidated the results; it slightly under-delivers on intended sequence diversity and is noted here so that the effective n is not overstated.

A reclassification, and what caused it — 1TIM was initially labelled clean and reclassified as stress-test on self-audit. Our first missing-residue count was derived from `REMARK 465` records, which reported zero. The structure in fact skips residue 3 in both chains, but 1TIM is a 1981 deposition with no `REMARK 465` section at all, so a metadata-based count cannot see the gap. The reliable signal turned out to be the official parser's own output: it inserts a `-` placeholder wherever residue numbering breaks, which `tied_featurize` later converts to `X`. Scanning the parsed JSONL for gap characters is a direct test of what ProteinMPNN actually perceives, and we adopted it as authoritative. The same audit explained a discrepancy in 1PIN, whose `REMARK 465` count of 10 exceeds its internal-gap count of 5 because five of those residues are N-terminal truncation that creates no gap in the parsed sequence.
The general lesson generalises past this dataset: structural metadata one would naively trust can be silently absent, and the parser's own view of the input is the thing worth checking.

A usability trap in 1TUP — The official parser has no concept of nucleic acid chains. 1TUP is a p53–DNA complex, and the parser read its two 21-nucleotide DNA chains as designable protein, inflating the reported length from 585 to 627 residues. Without an explicit restriction of the designed set to the three protein chains, using the repository's own chain-selection helper, the pipeline would silently redesign DNA as amino acids and report a sequence recovery number for it. Nothing warns the user. Anyone pulling multi-chain structures directly from the PDB should expect to hit this.

2.3 Inference and metrics

Both models were run over all 30 structures at sampling temperatures $T \in \{0.1, 0.2, 0.3\}$ with 8 sequences per backbone per temperature, seed 37, and `batch_size 1`, giving 720 designed sequences per model and 1,440 in total. Sequence recovery and negative log-likelihood come from the pipeline's own FASTA headers; perplexity is computed as $\exp(\text{score})$, where score is the mean per-residue NLL in nats, confirmed against the saved per-sample score arrays. Recovery is evaluated over the pipeline's own `mask_for_loss`, i.e. over designed, resolved positions only.
Three deviations from the documented defaults, all disclosed:

1. `--sampling_temp "0.1 0.2 0.3"` in a single invocation rather than three separate runs. The flag accepts a space-separated list and samples independently at each temperature; this is functionally identical with fewer model reloads.
2. `--num_seq_per_target 8` rather than the example scripts' 2, for more stable per-protein means.
3. `--ca_only` combined with `--jsonl_path` rather than `--pdb_path`, which is the only form shown in the documentation. We confirmed in `tied_featurize` that the Cα-only path subsets the already-parsed full-atom backbone internally, so no separate Cα-only parsing step exists or is needed. This is a supported code path, not a workaround.

All experiments ran on an Apple M3 laptop with 16 GB RAM under Python 3.10.20 and PyTorch 2.12.0, CPU only. Every table in this paper is regenerated from result CSVs by a committed script, so no reported number is transcribed by hand; the single exception is the ESMFold table, which is transcribed from a remote notebook run and is labelled as such.

3 Reproduction results

3.1 Installation and smoke test

Installation followed the official documentation without incident. Before touching our own dataset we reproduced the repository's example monomer script on 5L33 and 6MRR. 5L33 returned score 0.8576 and 38.68% recovery for its first sample at $T = 0.1$ with seed 37, and the saved score arrays matched the FASTA headers exactly. We treat this as confirmation that the environment, the weights, and the inference path were correct before any evaluation numbers were generated.

3.2 Sequence recovery, scoring, and the temperature sweep

	Full-backbone			Cα-only		
T	Recovery	NLL	Perplexity	Recovery	NLL	Perplexity
0.1	46.4%	0.866	2.397	41.3%	0.953	2.623
0.2	45.9%	0.898	2.478	40.5%	0.986	2.714
0.3	45.3%	0.954	2.629	39.9%	1.051	2.909

Table 2. Temperature sweep over 30 structures, 240 designed sequences per model per temperature.

Three qualitative claims from the original paper reproduce cleanly.

Full backbone context helps. The full-backbone model beats the Cα-only model at every temperature, by 5.1 percentage points at $T = 0.1$. This is the expected consequence of the N, Cα, C, O and virtual Cβ features being richer than a Cα trace alone.

Temperature trades accuracy for diversity. Recovery falls and perplexity rises monotonically with T for both models (Figure 1), matching the tradeoff the paper describes.

The pipeline is robust on curated inputs. Zero of 30 structures failed to parse or to complete inference, for either model, including all nine naturally incomplete stress-test structures and the 1TUP DNA complex under chain restriction.

What did not reproduce numerically. Our absolute recovery, 40–46%, sits below the paper's 52.4%. We do not believe this indicates an implementation problem, for two reasons: the smoke test reproduces the official example exactly, and the relative ordering and trends all hold. The most likely explanation is dataset mismatch. Our 30 hand-picked structures are not the paper's held-out CATH-clustered[8] test split of 690 monomers, 732 homomers and 98 heteromers, and that split is not separately downloadable, because the 16.5 GB training tarball is the only distributed source of the test cluster assignments.

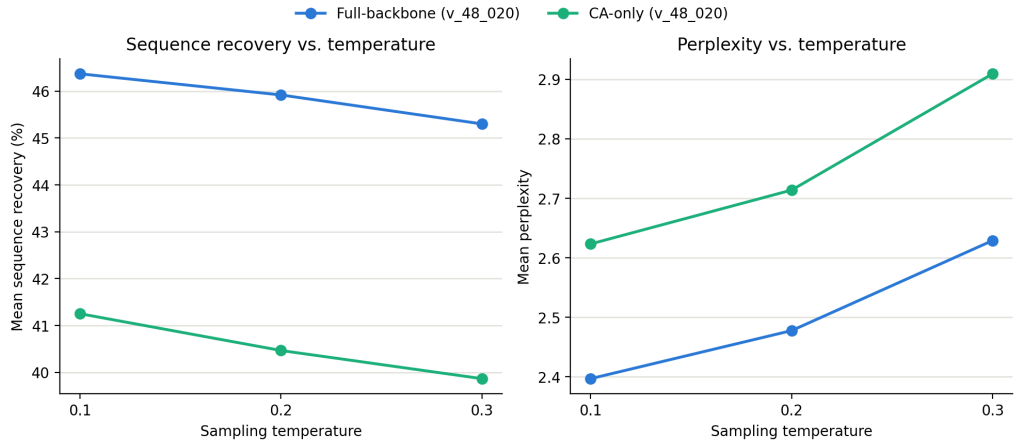

Figure 1. Sequence recovery (left) and perplexity (right) against sampling temperature for both model variants. Both trends are monotonic, and the full-backbone model leads at every temperature.

Our set also skews toward short, simple domains such as 1CRN at 46 residues. We say "most likely" rather than "is" because we could not test the hypothesis directly, which is itself a reproducibility finding: the evaluation split of a major published method is not practically obtainable at student scale.

3.3 Runtime, memory, and throughput

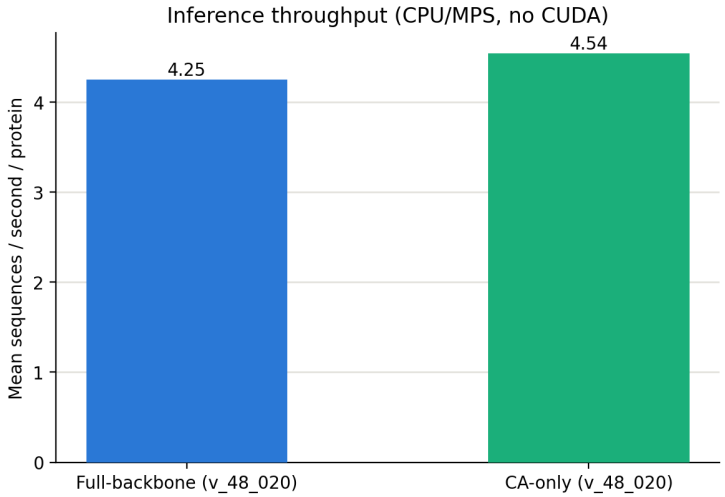

Figure 2. Mean inference throughput per protein on CPU (Apple M3). The Cα-only model is about 7% faster, consistent with its smaller per-residue feature tensor.

The full-backbone model produced 720 sequences in 281.9 s (153.2 sequences per minute, 9.40 s per protein); the Cα-only model took 262.6 s (164.5 per minute). Per-protein runtime ranged from 2.1 s for 1VII (36 residues) to 35.7 s for 1TUP (627 residues), for 24 sequences each.

Peak memory, measured with the system resource-usage timer, reached 862 MB RSS on the largest structure. This matters for the accessibility claim: a 16 GB laptop has an order of magnitude more memory than the largest structure in our set requires, and the model itself is 6.7 MB. Nothing about this workload needs a server.

3.4 Input completeness: a negative result

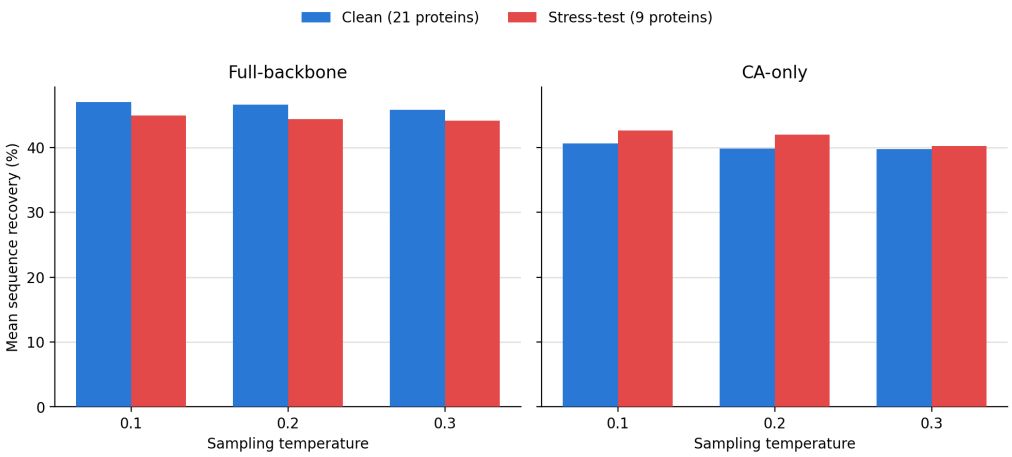

Figure 3. Recovery on the 21 clean versus 9 naturally incomplete structures. The full-backbone model loses about 2 pp on incomplete structures; the Cα-only model *gains* about 2 pp. The comparison is confounded and should not be read as evidence either way.

We expected recovery to degrade on the naturally incomplete subset. For the full-backbone model it does, slightly: -2.1 pp at $T = 0.1$ (47.0% clean vs. 44.9% stress-test), with a consistent sign across temperatures. For the Cα-only model the effect runs the *wrong way*: stress-test structures score 2.0 pp *better* (42.6% vs. 40.7%).

We report this as a negative result rather than dropping it. Two mechanisms plausibly account for it. First, ProteinMPNN's score and recovery are computed only over resolved positions; unresolved residues are excluded from the loss entirely, so missing density elsewhere in a structure does not directly penalise the positions actually being evaluated. Second, and more decisively, the split confounds completeness with protein identity: 21 structures versus a different 9, differing in fold, size and composition, which at this sample size plausibly swamps any true completeness effect.

The methodological conclusion is that a natural-subset comparison cannot test this claim. What is needed is a matched comparison, the same backbone at varying degrees of induced incompleteness, which is exactly what the next section does. We flag this explicitly because the roadmap for this reproduction predicted that "model behaviour changes predictably with input completeness," and by this measurement it does not.

4 Extensions

The extensions test whether ProteinMPNN's accessibility survives the conditions a small lab actually encounters: imprecise coordinates, missing density, no GPU, and imperfectly curated files.

4.1 Gaussian backbone noise, and a crossover

We applied isotropic Gaussian noise at $\{0, 0.1, 0.2, 0.3, 0.5\}$ Å to backbone coordinates for all 30 structures and both models.

The result we did not anticipate is that **the two models cross over**. The full-backbone model starts 6 pp ahead and degrades faster; by 0.5 Å it has fallen to 23.9% while the Cα-only model retains 25.4%. The perplexity curves cross earlier, near 0.36 Å, and by 0.5 Å the full-backbone model is not merely less accurate but more confidently wrong (perplexity 7.75 vs. 6.90).

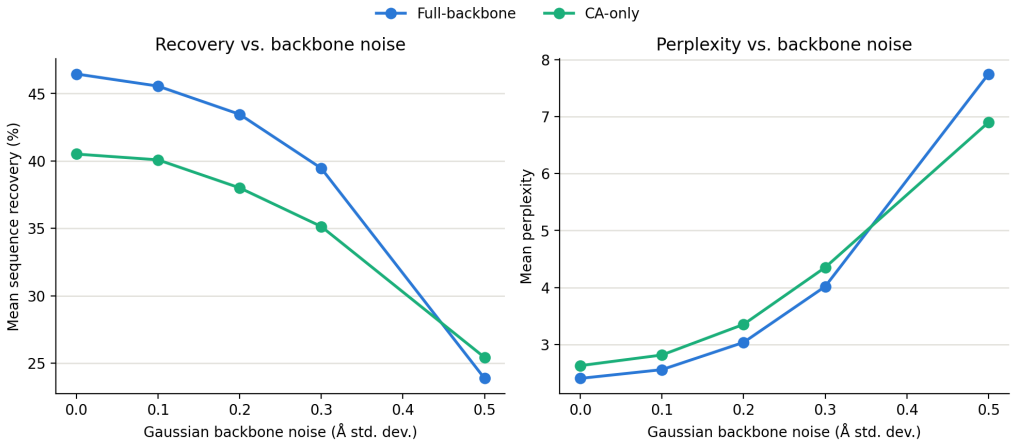

Figure 4. Recovery (left) and perplexity (right) against Gaussian noise applied to backbone coordinates, all 30 structures, $T = 0.1$. The two models cross over near 0.45 Å: past that point the Cα-only model, which is weaker on clean structures, becomes the more accurate one.

	Full-backbone		Cα-only	
Noise (Å)	Recovery	Perplexity	Recovery	Perplexity
0.0	46.5%	2.406	40.5%	2.630
0.1	45.6%	2.560	40.1%	2.817
0.2	43.5%	3.037	38.0%	3.353
0.3	39.5%	4.019	35.1%	4.353
0.5	23.9%	7.746	25.4%	6.899

Table 3. Robustness to Gaussian backbone noise. Note the reversal in both columns at 0.5 Å.

The mechanism we propose is that the full-backbone model's advantage (fine geometric detail from N, C and O positions, including implied backbone dihedrals) is precisely the part of the representation that noise destroys first. Displacing every atom independently by 0.5 Å corrupts inter-atomic geometry within a residue far more severely than it corrupts the Cα trace's coarser inter-residue distances. The richer representation is the more fragile one.

This has a practical reading. The Cα-only model is normally presented as a fallback for coarse-grained structures where nothing else is available. Our result suggests it is also the better *choice* when coordinates are present but untrustworthy: low-resolution cryo-EM, poorly refined models, coarse predictions. We would not put the crossover point at a precise value from five sampled noise levels on 30 structures, but the ordering reversal itself is unambiguous.

4.2 Missing residues behave differently from noisy ones

To get the controlled test that the clean/stress-test split could not provide, we masked a contiguous internal stretch of 10%, 20% or 30% of residues in each of the 21 clean structures, using the exact `-`/NaN convention the official parser produces for genuinely missing residues (verified against 1TIM's real gap). Recovery is computed by the pipeline's own `mask_for_loss` over non-masked positions only, so this measures how well the model designs the *still-visible* structure as context is removed, not how well it guesses the hidden region.

Both models degrade substantially, roughly 13.5 pp for full-backbone and 13.6 pp for Cα-only across a 30% mask, but they degrade in parallel, and the full-backbone model keeps its lead throughout. There is no crossover.

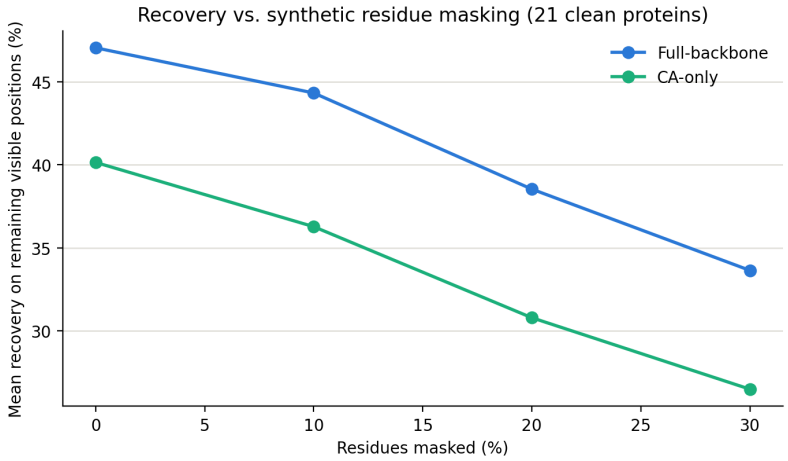

Figure 5. Recovery against synthetic contiguous residue masking on the 21 clean structures. Unlike the noise sweep, the full-backbone model keeps its lead at every masking level, and there is no crossover.

	Full-backbone		Cα-only	
Masked	Recovery	Perplexity	Recovery	Perplexity
0%	47.1%	2.422	40.1%	2.655
10%	44.3%	2.931	36.3%	3.172
20%	38.5%	3.507	30.8%	3.587
30%	33.6%	3.586	26.5%	3.307

Table 4. Robustness to synthetic contiguous residue masking, 21 clean structures.

Taken with the previous section, this is the sharper finding: **noisy structural context and missing structural context are different failure modes**, and they affect the two architectures differently. A practitioner with imprecise coordinates has a reason to prefer the Cα-only model; a practitioner with gaps and missing density does not. Collapsing both into a generic notion of "degraded input" would obscure this.

4.3 Low-resource inference

One provenance note applies to this subsection only. Every other number in this paper is regenerated from a result CSV by a committed script. The runtime and memory figures below were instead recorded in the experiment log at the time of the run, and the corresponding summary CSVs were never generated, so these particular values are not machine-traceable in the way the rest are. We report them because they are the measurements we made, and flag them so that a reviewer knows which numbers carry weaker provenance.

There is no GPU path on Apple Silicon — The official device selection reads `torch.device("cuda:0" if torch.cuda.is_available() else "cpu")`. It never checks `torch.backends.mps.is_available()`, so on Apple Silicon the GPU simply goes unused. We patched a separate copy to add an MPS branch to see what the acceleration was worth, and found that it does not merely under-accelerate; it crashes:

```
failed assertion 'Rank of updates array (1) must be greater than or equal
to inner-most dimension of indices array (48)'
```

The cause is in `gather_nodes`/`gather_edges`, which perform a 4-D `torch.gather` to build the $k = 48$ nearest-neighbour graph. Metal's `GatherND` kernel does not support that shape. This is not an edge case; graph construction runs on every forward pass. **MPS cannot currently run ProteinMPNN at all**, and CPU is the only working backend on this hardware. We report this as a finding rather than a limitation of our setup: for the large population of users on Apple laptops, "CPU versus GPU" has no meaningful answer here.

Fewer sequences, not smaller batches — On a 5-protein subset, reducing `num_seq_per_target` from 8 to 1 cut total time from 5.72 s to 0.78 s, a 7.3× speedup for 8× fewer sequences. The sublinearity is expected: the structure encoder's forward pass is shared across samples, and only the per-sample decode loop scales down. There is no accuracy cost; each sequence is independently valid.

Batch size behaves counter to intuition. Holding `num_seq_per_target` at 8, the subset ran in 5.08 s at `batch_size 1` and 2.94 s at `batch_size 8`: larger batches are *faster* on CPU, because vectorising across the batch dimension amortises fixed overhead. The cost is memory: peak RSS on the largest single chain rose from 641 MB to 2,258 MB, roughly 3.5×. The practical rule is therefore not "use small batches on weak hardware" but the more precise `batch_size=1` *is right when memory, not time, is the binding constraint*, which is the realistic laptop scenario.

4.4 Generalisation

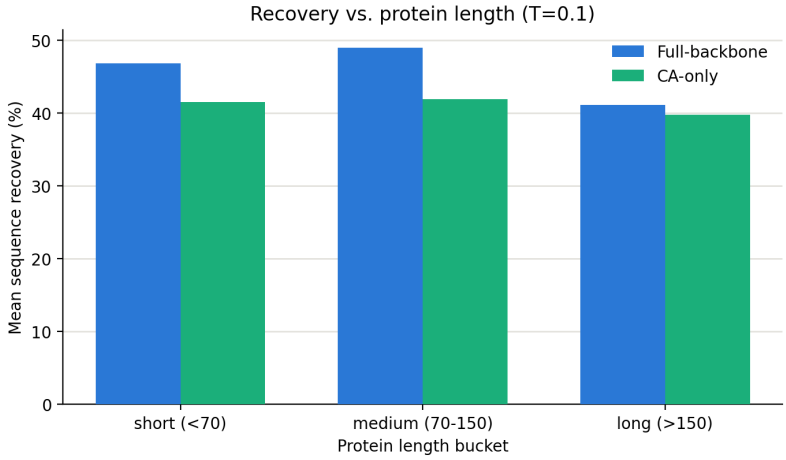

Figure 6. Recovery by protein length bucket at $T = 0.1$. Both models peak at medium length and fall off for chains above 150 residues.

Recovery peaks in the 70–150 residue bucket (49.0% full-backbone, 41.9% Cα-only, $n = 12$) and drops for chains above 150 residues (41.1% / 39.7%, $n = 7$). Short chains under 70 residues sit in between (46.8% / 41.5%, $n = 11$). Notably, the full-backbone model's advantage nearly vanishes on long proteins, shrinking from 7.1 pp at medium length to 1.4 pp.

Monomers outperform multimers by 8.3 pp (47.2% vs. 38.9%, full-backbone). We report this and decline to conclude from it: $n = 3$ multimers is far too small, and the three (1TIM, 4HHB, 1TUP) differ in far more than oligomeric state. The original paper, evaluating on hundreds of each, found recovery comparable across monomers, homomers and heteromers, and our sample gives us no grounds to contradict that.

Burial behaves as the paper describes. Using a Cα–Cα contact number proxy on five representative monomers, split into tertiles per protein, buried positions recovered at 42.2% against 33.3% for exposed positions (full-backbone). The direction and rough

magnitude match the paper's account of recovery correlating with local geometric context. We disclose the substitution explicitly: this is *not* DSSP[9] solvent accessibility, because `mkdssp` has no osx-arm64 build available through either conda or Homebrew. That is a small but real accessibility gap in the surrounding tooling rather than in ProteinMPNN itself.

4.5 Deliberate failure cases

We constructed six malformed or extreme inputs to characterise how the pipeline fails, on the view that a reproducible tool should fail loudly.

Input	Behaviour	Detail
Malformed coordinate field	Hard crash	Unhandled `ValueError` in the coordinate parser; kills the entire batch parse, not just the bad file
Unknown residue code `ZZZ`	**Silent**	Becomes a gap character, no warning
Chain-ID collision	**Silent**	Colliding (chain, resnum) keys overwrite in the parser's dict; residues lost, sequence silently truncated
`–max_length` exceeded	Clean discard	Explicit message; the one well-behaved path
`–backbone_noise 5.0`	**Silent**	No crash; sequences collapse to one repeated residue, ∼3% recovery, below random guessing
3,386-residue synthetic chain	Succeeds	24.5 s, 2.03 GB, no degradation

Table 5. Six constructed edge cases. Only two behave as a user would want.

Two cases behave well: the length check discards gracefully with an explicit message, and the pipeline scales to a 3,386-residue structure, about 5.4× our largest real structure, with proportionally higher runtime and memory and no behavioural degradation.
The other four do not, and three of them **fail silently**. An unknown residue code becomes a gap. A chain-ID collision quietly drops residues. Extreme coordinate distortion produces a confidently degenerate sequence (nearly all one residue, recovering below the ∼5% expected from uniform random guessing) while the run reports success and writes a FASTA. In each case the pipeline emits output that looks valid and is not.
This is the responsible-modelling payoff of the whole exercise. The failure that matters for reproducibility is not the crash; the crash is informative. It is the run that succeeds and produces garbage. Anyone applying this pipeline to real-world, imperfectly curated PDB files should validate inputs *beforehand,* because the pipeline will not catch most of this for them.

4.6 Do the designed sequences actually fold?

Sequence recovery is a proxy. ProteinMPNN's actual objective is to produce sequences that fold to the target structure, and a design can be excellent while sharing little sequence identity with the native protein. To test the claim the paper actually makes, we folded five designed sequences and their native counterparts with ESMFold[10] and compared confidence and backbone deviation.
Three of five designs (1UBQ, 2GB1, 1ENH) fold with high confidence (pLDDT 83–93) and Cα RMSD of 0.64–1.00 Å to the native backbone[11], at only 32–55% sequence identity. This is the strongest single result in our reproduction, because it validates the paper's actual claim in a way sequence recovery alone cannot: a design at 32% identity to native still reproduces the target fold to within 1 Å.
1VII is a consistent weak case: lowest designed pLDDT, highest RMSD among the successes, and the lowest sequence recovery of the five. We deliberately do *not* count 1CRN

Protein	Designed pLDDT	Native pLDDT	Cα RMSD (Å)	Recovery
1UBQ	92.8	90.5	0.79	55.3%
1VII	69.9	92.1	2.59	22.2%
2GB1	83.1	88.3	1.00	32.1%
1CRN	88.7	50.4	8.08	58.7%
1ENH	88.8	91.2	0.64	40.7%

Table 6. ESMFold validation of five designs (full-backbone, $T = 0.1$, sample 1). Transcribed from the remote notebook run rather than auto-generated from a CSV.

as a failure. ESMFold is only 50.4 pLDDT confident about the *native* crambin fold, the superposition logged a convergence warning, and the resulting 8.08 Å most likely reflects ESMFold's own difficulty with crambin's disulfide-stabilised mini-fold rather than a defect in the design. With $n = 5$ we present this as supporting evidence, not as a rate.

An accessibility finding from the validation itself — Lacking a local GPU, we ran ESMFold on Kaggle. The platform assigned a Tesla P100 (compute capability 6.0) whose capability the platform's own preinstalled PyTorch build does not support. A naive `.cuda()` call does not fail; the error surfaces only on the first real forward pass, as `no kernel image available`. We confirmed this was platform-side rather than caused by our installation by installing dependencies with `--no-deps`, and worked around it by checking `torch.cuda.get_device_capability()` directly and falling back to CPU, which folded these 36–76 residue sequences in a few minutes anyway. Six push-and-debug iterations were needed in total. Even "free GPU" platforms carry non-obvious compatibility gaps that a first-time user would hit and struggle to diagnose.

4.7 Per-residue structure of the recovered sequences

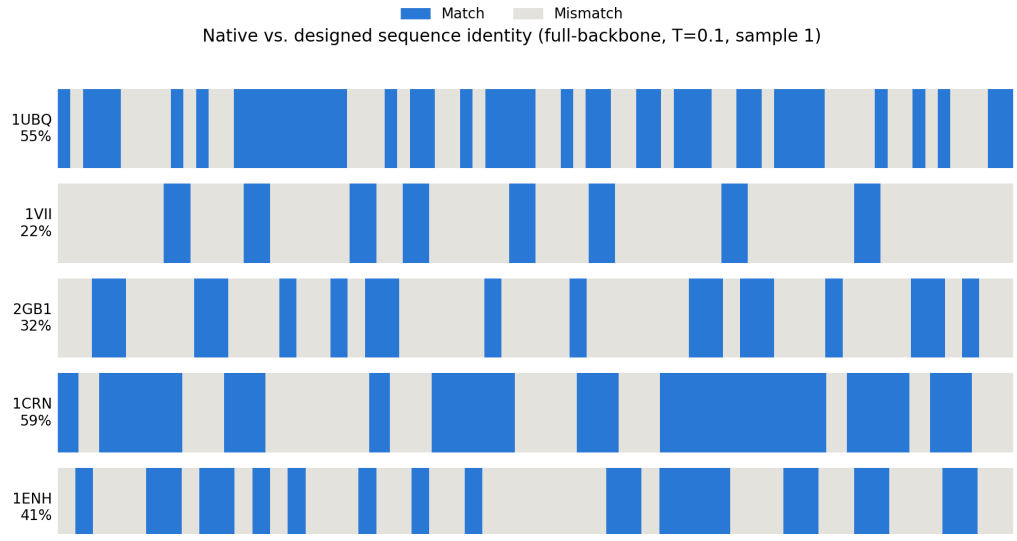

Figure 7. Per-residue match/mismatch between native and designed sequences for five structures (full-backbone, $T = 0.1$, sample 1). Each row is scaled independently to its own length: 1UBQ 76, 1VII 36, 2GB1 56, 1CRN 46 and 1ENH 54 residues. Matches cluster into contiguous runs rather than distributing uniformly.

Recovery is not uniformly distributed along the chain. Matches fall into contiguous blocks, with 1UBQ and 1CRN showing long recovered stretches and 1VII, the weakest

case by every measure we have, showing isolated single-residue matches with no runs at all. This is consistent with the burial result and with the paper's account: positions with strong local geometric context are strongly constrained and recovered reliably, while positions with weaker context admit many sequences and are recovered near chance. Aggregate recovery percentages conceal this structure entirely.

5 Discussion

5.1 What the reproduction says about the method

ProteinMPNN reproduces well in the way that matters most for a computational method: its qualitative claims survive contact with a different dataset, different hardware and an independent operator, using only the released code and weights. Full backbone context helps, temperature trades accuracy for diversity, burial predicts recovery, and the designs fold. Nothing required us to modify the model, and the one place where we needed to depart from documented usage was verified against the source as a supported path.

The accessibility claim holds up strongly. A 6.7 MB checkpoint that designs 150 sequences per minute on a laptop CPU, with peak memory under 1 GB, is genuinely usable by anyone. This is unusual and worth naming: reproducing this paper's computational core cost us tens of minutes of compute, not thousands of GPU-hours.

5.2 Where the reproducibility friction actually lives

None of our difficulties were with the model. They were with everything around it. The evaluation split is not separately downloadable, so the headline number cannot be checked directly. The parser silently accepts DNA chains as protein, silently converts unknown residues to gaps, and silently drops residues on chain-ID collision. Metadata that seems authoritative (`REMARK 465`) can be entirely absent from older depositions. A standard secondary-structure tool has no build for a common laptop architecture. A free GPU platform ships a PyTorch build incompatible with the GPU it assigns.

Individually these are small. Collectively they describe the actual reproducibility surface of computational biology, which is dominated by input handling and environment rather than by modelling. The silent failures concern us most: a pipeline that crashes teaches its user something, and a pipeline that returns a plausible-looking FASTA for a corrupted input does not.

5.3 The crossover, and why it might matter

The most interesting thing we found is not in the original paper. Under increasing coordinate noise the full-backbone and Cα-only models exchange places, and they do not do so under residue masking. If this holds on larger and more diverse sets, the practical guidance would be more nuanced than "use full-backbone unless you only have Cα atoms": the Cα-only model would be preferable whenever coordinate *precision* is doubtful even though coordinate *completeness* is fine, which describes a large class of low-resolution experimental and predicted structures. We regard this as a hypothesis our data supports rather than establishes.

6 Reproducibility statement

Everything needed to re-run this reproduction from a clean checkout is listed below. The full pipeline is driven by a single script and completes in roughly 30–45 minutes on CPU. The repository accompanying this paper contains the designed sequences (1,440 FASTA records), per-sample metrics, NLL score arrays, run logs, the five ESMFold structures and their native counterparts, and per-phase documentation giving the exact command

Item	Value
Original code	`github.com/dauparas/ProteinMPNN`
This reproduction	`github.com/AIscend-Research/protein-repro`
Commit	`8907e6671bfbfc92303b5f79c4b5e6ce47cdef57`
Checkpoints	`v_48_020` (vanilla and `ca_only`)
Hardware	Apple M3, 16 GB RAM, CPU only
Python	3.10.20 (conda env `proteinmpnn`)
PyTorch	2.12.0
Key packages	numpy 2.2.6, scipy 1.15.3, pandas 2.3.3, biopython 1.87, matplotlib 3.10.9
Seed	37 (all phases)
Temperatures	0.1, 0.2, 0.3
Sequences per backbone	8 per temperature
Batch size	1
Structures	30 (21 clean / 9 stress-test)
Designed sequences	1,440 (720 per model)
Full pipeline	single driver script, ∼30–45 min on CPU
Table regeneration	single script, from result files

Table 7. Reproducibility summary. The commit hash was verified unchanged before each phase.

line for every experiment reported here. Run logs are kept under version control, since the runtime figures are parsed out of them. The ESMFold validation is deliberately excluded from the main driver script: it requires a GPU and an external account, and is documented separately.

7 Limitations

1. Thirty structures with 28 unique sequences, hand-curated and skewed toward small classic domains, not the paper's CATH-clustered test split. Absolute recovery numbers should not be compared to the paper's as if measured on the same thing.
2. No experimental validation of any kind. ESMFold on five designs is the only structural check, and it is a model validating a model.
3. The burial analysis uses a geometric contact-number proxy, not DSSP solvent accessibility.
4. $n = 3$ multimers. The monomer/multimer gap is reported, not concluded from.
5. CPU only throughout. No CUDA device was available and MPS is unusable, so we have no GPU timings and cannot speak to GPU-specific behaviour.
6. The dataset-mismatch explanation for the recovery gap is the most plausible account available to us, but we could not test it, because the paper's test split is not separately obtainable.
7. The noise crossover rests on five sampled noise levels over 30 structures with a single seed. The reversal is clear; its precise location is not.

8 Conclusion

ProteinMPNN's computational core reproduces. Every qualitative claim we were able to test held: full-backbone design outperforms Cα-only design at every sampling temperature, recovery and diversity trade off monotonically, buried positions are recovered better than exposed ones, and designed sequences fold to the target backbone, three of five to within 1 Å at as little as 32% sequence identity. The pipeline installed from the official documentation, required no modification, and ran end to end on a laptop CPU

with zero failures across 30 structures and 1,440 designed sequences.
Absolute sequence recovery did not reproduce numerically: we measure 46.4% against the published 52.4%. We attribute this to dataset mismatch rather than implementation divergence, on the evidence that the official example reproduces exactly and all relative orderings hold, but we could not verify it, because the paper's evaluation split is not separately available. That inability is itself worth recording.
Beyond the paper, we find that the two model variants exchange places under coordinate noise near 0.45 Å but not under residue masking, that four of six constructed edge cases fail badly and three of those fail silently, that Apple Silicon GPU acceleration cannot run the model at all, and that larger batches are faster on CPU at a 3.5× memory cost.
For a method this central to modern protein design, the reassuring finding is how little stood between us and running it. The concerning finding is how much stood between us and running it *correctly*, and how little of that would have announced itself.

Acknowledgements

We thank the authors of ProteinMPNN for releasing their code, pretrained weights, and helper scripts publicly, without which this reproduction would not have been possible.

References

1. C. B. Anfinsen. "Principles that govern the folding of protein chains." In: **Science** 181.4096 (1973), pp. 223–230. DOI: 10.1126/science.181.4096.223.
2. A. Leaver-Fay, M. Tyka, S. M. Lewis, O. F. Lange, J. Thompson, et al. "ROSETTA3: an object-oriented software suite for the simulation and design of macromolecules." In: **Methods in Enzymology** 487 (2011), pp. 545–574. DOI: 10.1016/B978-0-12-381270-4.00019-6.
3. J. Dauparas et al. "Robust deep learning–based protein sequence design using ProteinMPNN." In: **Science** 378.6615 (2022), pp. 49–56. DOI: 10.1126/science.add2187.
4. J. Ingraham, V. K. Garg, R. Barzilay, and T. Jaakkola. "Generative models for graph-based protein design." In: **Advances in Neural Information Processing Systems 32**. 2019.
5. J. L. Watson, D. Juergens, N. R. Bennett, B. L. Trippe, J. Yim, et al. "De novo design of protein structure and function with RFdiffusion." In: **Nature** 620 (2023), pp. 1089–1100. DOI: 10.1038/s41586-023-06415-8.
6. H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. "The Protein Data Bank." In: **Nucleic Acids Research** 28.1 (2000), pp. 235–242. DOI: 10.1093/nar/28.1.235.
7. P. J. A. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, et al. "Biopython: freely available Python tools for computational molecular biology and bioinformatics." In: **Bioinformatics** 25.11 (2009), pp. 1422–1423. DOI: 10.1093/bioinformatics/btp163.
8. C. A. Orengo, A. D. Michie, S. Jones, D. T. Jones, M. B. Swindells, and J. M. Thornton. "CATH – a hierarchic classification of protein domain structures." In: **Structure** 5.8 (1997), pp. 1093–1108. DOI: 10.1016/S0969-2126(97)00260-8.
9. W. Kabsch and C. Sander. "Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features." In: **Biopolymers** 22.12 (1983), pp. 2577–2637. DOI: 10.1002/bip.360221211.
10. Z. Lin et al. "Evolutionary-scale prediction of atomic-level protein structure with a language model." In: **Science** 379.6637 (2023), pp. 1123–1130. DOI: 10.1126/science.ade2574.
11. D. L. Theobald. "Rapid calculation of RMSDs using a quaternion-based characteristic polynomial." In: **Acta Crystallographica Section A** 61.4 (2005), pp. 478–480. DOI: 10.1107/S0108767305015266.