

Disclosed and Undisclosed GPU Hardware Exclusion Across the Modern Protein Structure Prediction and Design Toolchain: A Compute-Accessibility Audit

Siddhardha Nanda

sn3199@columbia.edu

Columbia University <https://orcid.org/0009-0007-3825-0098>

Research Article

Keywords: GPU hardware accessibility, protein structure prediction, generative protein design, compute capability, bfloat16, Triton kernels, research equity, AlphaFold3, ESMFold2, Boltz

Posted Date: September 24th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11133200/v1>

License:  This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: The authors declare no competing interests.

Disclosed and Undisclosed GPU Hardware Exclusion Across the Modern Protein Structure Prediction and Design Toolchain: A Compute-Accessibility Audit

Siddhardha Nanda¹

¹Department of Biomedical Engineering, Columbia University in the City of New York

Abstract

Generative and predictive protein structure tools increasingly rely on bfloat16-optimized, Triton-compiled kernels for performance, a dependency that silently requires Ampere-generation or newer NVIDIA GPUs (compute capability ≥ 8.0) and excludes the Turing-generation hardware (compute capability 7.0–7.5) still common in academic compute clusters. While benchmarking generative protein-binder-design tools against amyloid and tau fibril targets, we independently and repeatedly encountered this barrier with one specific tool (ESMFold2), which fails outright on Turing hardware with no documentation warning users in advance. Motivated by this, we conducted a systematic compute-accessibility audit of 13 major protein structure prediction and design tools released between 2021 and 2026, assessing each tool’s actual hardware requirements against primary sources (official documentation, source code, and GitHub issue trackers) and, where we had direct access, empirical testing on Turing-generation hardware. We find that hardware exclusion is not predicted by a tool’s age: RFdiffusion3 (2025) runs on Turing GPUs with no modification, while AlphaFold3 and Chai-1 (both 2024) do not. More strikingly, tools that fail on older hardware do so in qualitatively different and unequally documented ways. We propose a five-category taxonomy of outcomes: no exclusion; disclosed hard exclusion with a clean failure (Chai-1); disclosed hard exclusion with silent data corruption rather than an error (AlphaFold3, confirmed via the developers’ own bug-tracker responses); undisclosed hard exclusion with a clean failure (ESMFold2); and undisclosed exclusion with an available graceful-degradation fallback (Boltz-1/Boltz-2). We additionally document a symmetric, opposite-direction failure mode in which a tool (the original RFdiffusion) breaks on GPUs newer than its intended target due to stale CUDA/PyTorch version pinning. We argue that undocumented hardware requirements, and particularly silent-corruption failures, represent an underappreciated and unequally distributed barrier to computational biology research, and we recommend that tool developers disclose minimum compute capability explicitly alongside every release.

1 Introduction

Generative protein design and structure prediction have advanced rapidly since 2021, with a new major tool released every few months. This pace has been accompanied by a parallel, less-discussed trend: increasing reliance on bfloat16 (bf16) tensor-core arithmetic and Triton-compiled GPU kernels for performance, both of which require NVIDIA’s Ampere generation or newer (compute capability ≥ 8.0) [1]. Turing-generation GPUs (compute capability 7.0–7.5), including the TITAN RTX, RTX 20-series, and Tesla T4 and V100 cards still widely deployed in academic compute

clusters, cannot run bf16 tensor-core operations and are consequently excluded from any tool that hard-requires them, regardless of available VRAM or raw compute throughput.

We encountered this barrier directly. While benchmarking generative protein-binder-design tools against amyloid and tau amyloid fibril targets on a three-GPU Turing-generation workstation, we attempted to include a fourth tool, ESMFold2, alongside RFdiffusion3 [2], BindCraft [3], and BindEnergyCraft [4]. RFdiffusion3, BindCraft, and BindEnergyCraft all ran without modification. ESMFold2 failed unconditionally with a low-level compiler error (`ptxas` rejecting a bf16 instruction as requiring `sm_80` or higher) regardless of model size, and this requirement is not stated anywhere in the tool’s documentation prior to encountering the failure.

This raised a broader question: is ESMFold2’s undocumented hardware exclusion an isolated case, or a more general pattern across the current generative-biology toolchain? To answer this, we surveyed 13 major protein structure prediction and design tools spanning 2021–2026, combining our own empirical testing (four tools, tested directly on Turing hardware) with a systematic review of each remaining tool’s official documentation, source code, and public issue trackers for evidence of hardware compatibility or exclusion. We find that the pattern is neither universal nor simply age-correlated, but instead falls into a small number of qualitatively distinct failure modes that differ sharply in how dangerous and how disclosed they are.

2 Methods

2.1 Tool selection

We selected 13 tools to span both protein structure *prediction* and *design*, and to span the period from the first deep-learning structure-prediction breakthrough (2021) through the most recent releases available at the time of this study (2026): AlphaFold2 and AlphaFold2-Multimer [5], RoseTTAFold [6], ESMFold [7], ProteinMPNN [8], RFdiffusion [9], AlphaFold3 [10], Chai-1 [11], ESM3 [12], Boltz-1 and Boltz-2 [13], RFdiffusion3 [2], BindCraft [3], BindEnergyCraft [4], and ESMFold2. RoseTTAFold2 [14] was additionally checked alongside RoseTTAFold given its shared lineage and near-identical hardware profile. This set is not exhaustive of the field but covers the tools most central to the current generative and predictive protein-structure ecosystem.

2.2 Assessment procedure

For four tools (RFdiffusion3, BindCraft, BindEnergyCraft, ESMFold2) we had direct empirical access: all four were installed and run on a workstation with three NVIDIA TITAN RTX GPUs (24 GB VRAM each, Turing architecture, compute capability 7.5) as part of our concurrent generative-binder-design benchmarking work, and their success or failure on this hardware is reported from first-hand testing, not documentation review.

For the remaining nine tools, we assessed hardware requirements through: (1) the tool’s official README, installation documentation, or methods section for any explicitly stated minimum GPU compute capability; (2) the tool’s source code and environment/dependency files (e.g., `environment.yml`, `pyproject.toml`) for explicit use of bfloat16 precision or Triton-compiled kernels; and (3) the tool’s public GitHub issue tracker, searched for reports mentioning specific older-GPU hardware (V100, T4, RTX 20-series, TITAN RTX) or architecture-related error signatures (e.g., “compute capability,” `sm_80`, `cutlassF`), to find direct user-reported evidence of successful or unsuccessful execution on pre-Ampere hardware. Where GitHub issue evidence conflicted with or extended beyond official documentation, we report both and note the discrepancy explicitly, since this discrepancy is itself the central finding of this study.

3 Results

3.1 No single pattern predicts hardware exclusion

Table 1 summarizes hardware compatibility across all 13 tools. Five tools (AlphaFold2/-Multimer, RoseTTAFold/RoseTTAFold2, ESMFold, ProteinMPNN, and the original RFdiffusion) use no bf16-gated or Triton-compiled kernels and run on Turing-generation hardware without modification; three of these have specific, publicly documented evidence of Turing or older execution: ColabFold’s long-standing default use of Tesla T4 GPUs for AlphaFold2 and RoseTTAFold2 [15], and ESMFold’s own paper reporting inference benchmarks on a V100, compute capability 7.0, older than Turing [7]. Notably, tool age does not predict this outcome: RFdiffusion3 (2025) and BindCraft and BindEnergyCraft (2025) all run on Turing hardware with no changes, while AlphaFold3 and Chai-1 (both 2024, a year earlier) do not.

Table 1: Hardware compatibility across 13 protein structure prediction and design tools. “Confirmed” entries marked with † are from our own direct empirical testing; all other entries are from [documentation](#) and [GitHub issue-tracker review](#).

Tool	Year	bf16/Triton gate	Turing outcome	Disclosed
AlphaFold2 / -Multimer	2021	No	Works	N/A
RoseTTAFold / RF2	2021/23	No	Works	N/A
ESMFold	2022	No	Works	N/A
ProteinMPNN	2022	No	Works	N/A
RFdiffusion (original)	2023	No	Works*	N/A
AlphaFold3	2024	Yes	Silent corruption	Yes
Chai-1	2024	Yes	Clean failure	Yes
ESM3	2024	Unclear	Insufficient evidence	–
Boltz-1 / Boltz-2	2024/25	Yes (optional)	Likely graceful fallback	No
RFdiffusion3	2025	No	Works†	N/A
BindCraft	2025	No	Works†	N/A
BindEnergyCraft	2025	No	Works†	N/A
ESMFold2	2025/26	Yes	Clean failure†	No

*RFdiffusion’s own dependency stack instead breaks on GPUs *newer* than compute capability 8.6 (see Section 3.3).

3.2 A five-category taxonomy of hardware-exclusion outcomes

Among the tools that do impose a bf16/Triton gate, we observe four qualitatively distinct outcomes, plus a fifth “no exclusion” baseline, illustrated in Figure 1.

No exclusion. The majority of tools surveyed (AlphaFold2/-Multimer, RoseTTAFold/RF2, ESMFold, ProteinMPNN, RFdiffusion, RFdiffusion3, BindCraft, BindEnergyCraft) impose no bf16/Triton requirement and run on Turing-generation hardware without modification.

Disclosed hard exclusion, clean failure. Chai-1’s own README explicitly states a requirement for “GPU with CUDA and bfloat16 support” and recommends only Ampere-or-newer cards. Consistent with this stated requirement, users attempting to run it on a V100 receive an explicit, immediate error (`cutlassF: no kernel found to launch! ... likely your GPU doesn’t support bfloat16`), confirmed in a public GitHub issue [16]. This is the best-case failure mode: the requirement is disclosed before installation, and the failure is unambiguous.

Disclosed hard exclusion, silent corruption. AlphaFold3’s installation documentation explicitly states a requirement of “Compute Capability 8.0 or greater” [17]. However, the practical failure mode on pre-Ampere hardware is markedly worse than a clean error: a GitHub issue documents the pipeline running to completion on a V100 and multiple Turing-class cards without any error message, while silently producing structurally invalid output that a DeepMind maintainer described as “look[ing] like random noise” [18]. A researcher who did not already know to distrust output on this hardware class would have no signal that anything had gone wrong.

Undisclosed hard exclusion, clean failure. ESMFold2 imposes a hard bf16/Triton requirement that is not stated in any documentation we could find prior to encountering the failure. Attempting to run it on Turing hardware produces an immediate, low-level compiler failure (ptxas rejecting a bf16 instruction, requiring `sm_80` or higher) regardless of model size; we confirmed this independently on both the full 6-billion-parameter and a reduced 300-million-parameter checkpoint. Unlike Chai-1, no README, installation guide, or release notes gave any indication of this requirement before the failure was encountered.

Undisclosed exclusion, graceful degradation available. Boltz-1/Boltz-2 use bf16 automatic mixed precision and Triton-compiled kernels (via the `trifast` package) for triangle-attention operations by default, and Triton’s own NVIDIA backend has a hard floor of compute capability 8.0, unsupported on Turing [1]. However, a documented `--no_kernels` flag disables this accelerated path and falls back to standard PyTorch operations; users reported this resolves the corresponding failure mode on unsupported hardware configurations with no major perceived correctness cost, only a performance penalty [19]. We were unable to find a report from a user specifically running Boltz on Turing-class hardware with this flag, so we report this outcome as probable rather than confirmed. Notably, the underlying kernel-compatibility issue in this case was reported not only on Turing but also on some Ampere-and-newer cards (L40S, RTX 4070/4080) due to `cuequivariance`/Triton version pinning, indicating that even the “exclusion” side of this boundary is not cleanly architecture-determined for this tool.

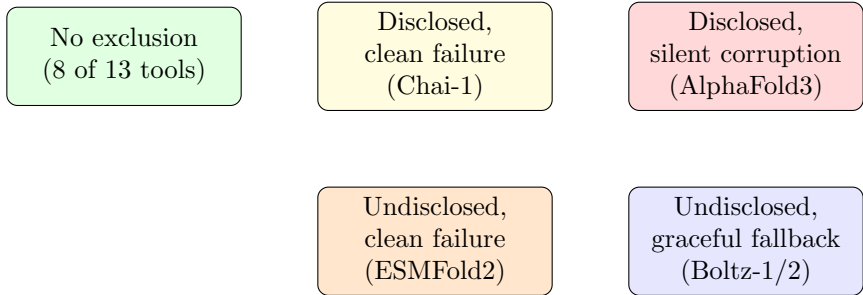


Figure 1: A five-category taxonomy of GPU hardware-exclusion outcomes observed across the 13 surveyed tools. Severity increases from left to right within each row (a clean, immediate failure is preferable to a silent one); disclosure status separates the top row (stated upfront in documentation) from the bottom row (discovered only through failure or community bug reports).

3.3 A symmetric, opposite-direction failure: newer hardware breaking older tools

The original RFDiffusion presents a different problem entirely. Its README explicitly declines to provide a universally compatible environment, stating that the maintainers are “not able to make one environment that will run on all setups,” and ships a dependency specification pinned to CUDA 11.1 and PyTorch 1.9. A GitHub issue from a user running an RTX 5090 (a 2025-

generation consumer GPU) shows a CUDA error whose accompanying PyTorch warning lists the supported compute capabilities of the exact build RFDiffusion’s environment resolves to as `sm_37 sm_50 sm_60 sm_70 sm_75 sm_80 sm_86` [20] – explicitly including `sm_75` (Turing) but excluding the compute capability of GPUs released after this dependency specification was pinned. This is the mirror image of the undisclosed-exclusion pattern documented above: rather than a new tool excluding old hardware, an older tool’s frozen dependency stack excludes hardware newer than what existed when it was built. We did not find any Turing-specific compatibility complaint in RFDiffusion’s issue tracker, forum, or discussion pages, consistent with the tool working as intended on the hardware class this study is concerned with.

4 Discussion

The central finding of this study is that GPU hardware exclusion in the current protein-structure toolchain is neither universal, nor simply predicted by a tool’s release date, nor consistently disclosed. Three tools released in the same year (2024: AlphaFold3, Chai-1, Boltz-1) each handle the same underlying hardware boundary differently: one discloses it and fails loudly, one discloses it and fails silently and dangerously, and one does not clearly disclose it but offers an escape hatch. A fourth 2025-generation tool, ESMFold2, imposes the same class of hardware requirement with neither disclosure nor an escape hatch.

We consider the disclosed-versus-undisclosed distinction to be more practically important than the mere presence of a hardware requirement. A researcher planning a project around a tool with a clearly stated Ampere-or-newer requirement can check their hardware before investing setup time; a researcher using ESMFold2 cannot know this until the installation is already complete and the first inference attempt fails. We consider AlphaFold3’s silent-corruption failure mode to be the most concerning outcome identified in this study, not because the hardware requirement itself is undisclosed – it is clearly stated – but because a researcher who overlooks or is unaware of that documentation receives no runtime signal that their results are invalid, a failure mode with direct implications for research integrity if such output were used downstream without independent validation.

This pattern plausibly has an unequal impact across the research community. Turing-generation and older GPUs remain common in academic departments and shared computing clusters that were not recently refreshed, disproportionately affecting research groups without dedicated hardware budgets or access to cloud compute credits. An undisclosed hardware requirement converts what would otherwise be a simple compatibility check into wasted installation time, debugging effort, and in the silent-corruption case, potentially wasted downstream research effort building on invalid results.

We recommend that developers of generative-biology tools state minimum GPU compute capability explicitly in the first paragraph of their installation documentation, following AlphaFold3’s and Chai-1’s practice, and that they consider providing a graceful-degradation fallback (as Boltz-1/Boltz-2 does, even if undocumented) rather than requiring specialized kernels unconditionally.

5 Limitations

This study has several limitations that should inform interpretation of its results. First, our assessment method differs across tools: four tools (RFDiffusion3, BindCraft, BindEnergyCraft, ESMFold2) were tested directly by us on Turing hardware, while the remaining nine were assessed through documentation and public issue-tracker review rather than independent empirical

re-verification; the latter method is dependent on the completeness and accuracy of third-party bug reports, which may not surface every relevant compatibility issue. Second, for ESM3 we found insufficient evidence to classify its Turing compatibility and report this as an open gap rather than an inferred answer; we consider this preferable to speculation, but it means our taxonomy is not fully populated across all 13 tools. Third, our classification of Boltz-1/Boltz-2 as “likely graceful degradation” is based on a documented workaround and user reports of it resolving a related but not identical failure signature, not a direct confirmation from a Turing-hardware user; this should be read as a plausible inference, not a confirmed result. Fourth, our tool selection, while spanning both prediction and design tools across 2021–2026, is not exhaustive of the field and reflects the tools most directly relevant to our concurrent benchmarking work; other widely used tools (for example, OmegaFold or ESM Cambrian) were not assessed here. Fifth, hardware and software requirements for actively maintained tools can change after the time of this study; the compatibility status reported here reflects the specific versions and documentation available at the time of assessment and may not remain current.

Data Availability Statement

The list of tools surveyed, the search strategy used for each tool’s issue tracker, and the specific GitHub issues cited as evidence are available upon request from the corresponding author.

Ethics Declaration

This study involved no human subjects, animal subjects, or new experimental biological material; it is a documentation and software-compatibility review supplemented by hardware testing on computational infrastructure only.

Author Contributions

S.N.: Conceptualization, Methodology, Formal Analysis, Investigation, Writing – Original Draft, Writing – Review & Editing.

Conflict of Interest Statement

The author declares no competing interests.

Funding

This work did not receive external funding.

AI Disclosure Statement

AI-assisted tools (Claude, Anthropic) were used during this study for literature search assistance, software debugging, and manuscript drafting support under direct author supervision. All scientific interpretations, conclusions, and final text were reviewed and approved by the author.

References

- [1] Triton Language and Compiler. NVIDIA backend architecture support documentation. GitHub repository: [triton-lang/triton](https://github.com/triton-lang/triton). Accessed 2026.
- [2] Butcher J, Krishna R, Mitra R, Brent RI, Li Y, Corley N, et al. De novo Design of All-atom Biomolecular Interactions with RFdiffusion3. *bioRxiv* [Preprint]. 2025. doi:10.1101/2025.09.18.676967
- [3] Pacesa M, Nickel L, Schellhaas C, Schmidt J, Pyatova E, Kissling L, et al. One-shot design of functional protein binders with BindCraft. *Nature*. 2025;646(8084):483-492.
- [4] Nori D, Parsan A, Uhler C, Jin W. BindEnergyCraft: Casting Protein Structure Predictors as Energy-Based Models for Binder Design. *arXiv* [Preprint]. 2025. arXiv:2505.21241.
- [5] Jumper J, Evans R, Pritzel A, Green T, Figurnov M, Ronneberger O, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*. 2021;596(7873):583-589.
- [6] Baek M, DiMaio F, Anishchenko I, Dauparas J, Ovchinnikov S, Lee GR, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*. 2021;373(6557):871-876.
- [7] Lin Z, Akin H, Rao R, Hie B, Zhu Z, Lu W, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*. 2023;379(6637):1123-1130.
- [8] Dauparas J, Anishchenko I, Bennett N, Bai H, Ragotte RJ, Milles LF, et al. Robust deep learning-based protein sequence design using ProteinMPNN. *Science*. 2022;378(6615):49-56.
- [9] Watson JL, Juergens D, Bennett NR, Trippe BL, Yim J, Eisenach HE, et al. De novo design of protein structure and function with RFdiffusion. *Nature*. 2023;620(7976):1089-1100.
- [10] Abramson J, Adler J, Dunger J, Evans R, Green T, Pritzel A, et al. Accurate structure prediction of biomolecular interactions with AlphaFold3. *Nature*. 2024;630(8016):493-500.
- [11] Chai Discovery Team, Boitreaud J, Dent J, McPartlon M, Meier J, Reis V, Rogozhonikov A, Wu K. Chai-1: Decoding the molecular interactions of life. *bioRxiv* [Preprint]. 2024. doi:10.1101/2024.10.10.615955
- [12] Hayes T, Rao R, Akin H, Sofroniew NJ, Oktay D, Lin Z, et al. Simulating 500 million years of evolution with a language model. *Science*. 2025;387(6736):850-858.
- [13] Wohlwend J, Corso G, Passaro S, Getz N, Reveiz M, Leidal K, et al. Boltz-1: Democratizing Biomolecular Interaction Modeling. *bioRxiv* [Preprint]. 2024. doi:10.1101/2024.11.19.624167
- [14] Baek M, Anishchenko I, Humphreys IR, Cong Q, Baker D, DiMaio F. Efficient and accurate prediction of protein structure using RoseTTAFold2. *bioRxiv* [Preprint]. 2023. doi:10.1101/2023.05.24.542179
- [15] Mirdita M, Schütze K, Moriwaki Y, Heo L, Ovchinnikov S, Steinegger M. ColabFold: making protein folding accessible to all. *Nat Methods*. 2022;19(6):679-682.
- [16] [chaidiscovery/chai-lab](https://github.com/chaidiscovery/chai-lab) GitHub repository, Issue #46: bfloat16 kernel failure on V100 GPU. Accessed 2026.

- [17] google-deepmind/alphafold3 GitHub repository, docs/installation.md: stated minimum Compute Capability 8.0 requirement. Accessed 2026.
- [18] google-deepmind/alphafold3 GitHub repository, Issue #59: model does not run correctly on CUDA Capability 7.x GPUs. Accessed 2026.
- [19] jwohlwend/boltz GitHub repository, Issue #485: Triton-based component import failure and `--no_kernels` workaround. Accessed 2026.
- [20] RosettaCommons/RFdiffusion GitHub repository, Issue #380: CUDA compatibility error on RTX 5090. Accessed 2026.