ALPHAFAST: HIGH-THROUGHPUT ALPHAFOLD 3 VIA GPU-ACCELERATED MSA CONSTRUCTION

Benjamin C. Perry[*], **Jeonghyeon Kim**[*] **& Philip A. Romero**
Department of Biomedical Engineering
Duke University
Durham, NC 27708, USA
{benjamin.perry,jeonghyeon.kim,philip.romero}@duke.edu

ABSTRACT

AlphaFold 3 (AF3) enables accurate biomolecular modeling but is limited by slow, CPU-bound multiple sequence alignment (MSA) generation. We introduce AlphaFast, a drop-in framework that integrates GPU-accelerated MMseqs2 sequence search to remove this bottleneck. AlphaFast achieves a 68.5× speedup in MSA construction and a 22.8× reduction in end-to-end runtime on a single GPU, and delivers predictions in 8 seconds per input on four GPUs while maintaining indistinguishable structural accuracy. A serverless deployment enables structure prediction for as little as $0.035 per input. Code is available at https://github.com/RomeroLab/alphafast

1 INTRODUCTION

The release of AlphaFold 3 extended accurate protein modeling from single protein chains to protein–ligand, protein–DNA, and protein–RNA complexes (Abramson et al., 2024). Despite its predictive power, inference remains computationally expensive, limiting practical use in high-throughput experiments in proteomics, interactomics, or synthetic biodesign. A key contributor to this cost is the construction of multiple sequence alignments (MSAs). MSAs encode evolutionary information that is critical to state-of-the-art folding and interaction prediction, but their generation requires searching giga-scale reference databases (Lee et al., 2024).

Over the past few decades, MSA construction has progressed from exact dynamic programming solutions to fast heuristic and sensitive profile-based aligners such as BLAST (Altschul et al., 1990), DIAMOND (Buchfink et al., 2021), HMMER (Eddy, 2011), and MMSeqs (Hauser et al., 2016; Steinegger & Söding, 2017). While these advances substantially reduce search time, large-scale MSA generation remains computationally demanding. As a result, many modern biomolecular prediction pipelines that strictly require (Jumper et al., 2021; Wohlwend et al., 2025; Mirdita et al., 2022) or obtain best results from (Discovery et al., 2024) inclusion of MSAs outsource their construction to hosted web servers (Mirdita et al., 2019). While this approach lowers the barrier to entry, it introduces additional latency, prevents MSA parameter tuning, complicates deployment in high-performance computing (HPC) environments, and remains low throughput.

Recently, MMseqs2-GPU demonstrated that GPU-accelerated sequence search can achieve an order-of-magnitude speedup over CPU-bound pipelines, enabling fast, fully local MSA generation while substantially reducing inference time in AlphaFold 2 workflows (Kallenborn et al., 2025). However, extending these gains to AF3 is non-trivial given incompatible data pipelines, the post- and pre-processing requirements of MMSeqs2-GPU, as well as file input/output (I/O) optimization in HPC environments. Furthermore, there exists no cost-effective, high throughput implementation of AF3 available to researchers without access to significant computational resources.

In this work, we introduce AlphaFast, a simplified pipeline that replaces CPU-bound JackHMMER with GPU-accelerated MMseqs2 while preserving the original AF3 folding module, weights, and end-to-end performance. AlphaFast supports cached batch processing for ligands and proteins in both single-GPU and multi-GPU setups. On a single GPU, AlphaFast achieves up to a $68.5\times$

[*]These authors contributed equally.

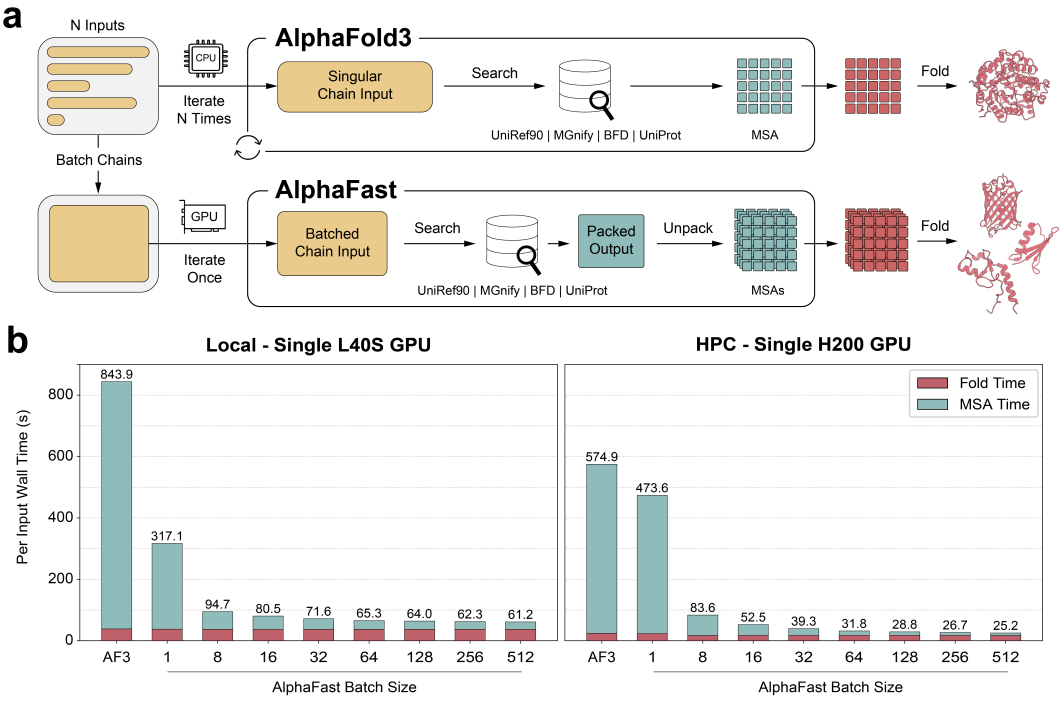

Figure 1: **AlphaFast architecture enables inference speeds scaling with batch size.** (a) Architectural comparison between AF3 (top) and AlphaFast (bottom). AF3 iterates N times over inputs on CPU, processing one chain at a time through parallel database searches. AlphaFast batches all chains, iterates once on GPU, searches sequentially, produces a packed output database, then unpacks and converts individual MSAs before folding. (b) Average total wall time running inference a single NVIDIA L40S GPU on a bare metal server with Docker (left) vs a single NVIDIA H200 GPU on an HPC environment with Singularity (right). Stacked bar charts demonstrate homology search dominates AF3 inference and is increasingly reduced at larger query batch sizes.

speedup in MSA generation and a $22.8\times$ reduction in end-to-end (wall clock). With four GPUs, AlphaFast reaches speeds of 8 seconds per input (a $71.2\times$ acceleration), scaling nearly linearly with additional GPUs. We verify that AlphaFast produces MSAs and structures that are statistically indistinguishable from AF3 MSAs and outputs. Finally, we hope AlphaFast's modular design will serve as an implementation guide to future folding architectures that decouple MSA construction from inference.

2 METHODS

AlphaFast introduces three key architectural improvements to AF3. First, unlike AF3's per-chain JackHMMER search (Figure 1a, top), AlphaFast consolidates unique sequences into a batched MMseqs2-GPU query for sequential database search (Figure 1a, bottom). Second, AlphaFast maximizes throughput by offloading the CPU-bound MSA post-processing of database N concurrently with GPU search of database $N+1$. Finally, AlphaFast enforces a strict two-stage architecture to resolve video memory (VRAM) conflicts between JAX initialization (Bradbury et al., 2018) and MSA generation. For fair comparison, we strictly match AF3's default E-values (10^{-4}) and sequence limits for all four reference databases: UniRef90 (Suzek et al., 2015), MGnify Mitchell et al. (2020), Small BFD (Jumper et al., 2021; Mirdita et al., 2022), and UniProt (Consortium, 2019).

To benchmark AlphaFast against AF3, we curated a set of 32 and 512 protein monomer, protein-protein, and protein-ligand targets sampled from the PDB (wwPDB consortium, 2018) (see Appendix A.2 for selection criteria). For each configuration, we ran benchmarks in triplicate, measuring average per-input wall time while varying the AlphaFast GPU query batch size from 1 to 512. For fair comparison, we run AF3 in batched mode with four parallel JackHMMER searches

and model compilation. We test AlphaFast in two common setups: a local, bare metal server with NVIDIA L40S GPUs and an HPC environment with the SLURM job scheduler (Yoo et al., 2003) and NVIDIA H200 GPUs. AlphaFast speedups are calculated relative to AF3 running on the exact same hardware. Further benchmarking information can be found in Appendix B.1.

For multi-GPU inference, AlphaFast employs a phase-separated parallel architecture. In Phase 1, inputs are partitioned round-robin across all available GPUs and each GPU independently runs the batched MSA pipeline on its partition. After all MSA processes complete, intermediate feature files are written to disk. Phase 2 then re-partitions the outputs across GPUs for parallel folding. This design ensures clean separation of GPU-bound MSA search from JAX-based inference while achieving near-linear scaling. We expand the test set to $2,048$ inputs to account for a batch size of 512 on 4 GPUs (see Appendix A.3)

3 RESULTS

3.1 GPU-ACCELERATED MSA LOOKUP SCALES WITH BATCH SIZE

Single NVIDIA L40S. On a single NVIDIA L40S GPU (Figure 1b, left), AF3 requires 843.9 seconds per input, with MSA construction accounting for the vast majority (95.4%) of wall time. Without batching, AlphaFast reduces this to 317.1 seconds and continues to improve with larger batch sizes. Efficiency gains begin to plateau around a batch size of 64 (65.3 seconds per input), likely due to GPU memory bandwidth saturation. The highest efficiency gain occurs at the maximum tested batch size of 512 (61.2 seconds per input), yielding a maximum $13.8\times$ end-to-end speedup on a single L40S GPU. Larger batch sizes likely will reduce this further, but diminishing returns are expected.

Single NVIDIA H200. On a single NVIDIA H200 GPU, performance gains are amplified further (Figure 1b, right). The increased HBM3e memory bandwidth enables continued scaling across all tested batch sizes without a clear plateau in MSA per-input time. AF3 requires 574.9 seconds per input, with MSA construction consuming 95.8% of total wall time. Wall time decreases rapidly to a minimum of 25.2 seconds per input at the maximal tested batch size of 512 (a $22.8\times$ speedup). At this batch size, MSA time consumes 31.9% of runtime and is likely to reduce further at even larger batch sizes. Notably, sequential processing yields only a 1.2-fold speedup on the H200, slower than the L40S at the same batch size. We attribute this to I/O bottlenecks on the shared HPC filesystem during intermediate file production; at larger batch sizes, GPU throughput compensates for this overhead.

Multi-GPU. On a 4xL40S setup, AlphaFast achieves a throughput of 19.4 seconds per input (8.2 second MSA, 11.2 second fold), a $3.1\times$ improvement over the single-GPU configuration and a $43.5\times$ end-to-end speedup over the AF3 baseline. On a 4xH200 setup, per-input time drops further to 8.1 seconds per input (3.3 seconds MSA, 4.8 second fold), yielding a $71.2\times$ speedup over the same-hardware baseline and reducing MSA construction time by over two orders of magnitude. Both configurations exhibit approximately 78% parallel efficiency across four GPUs. The remaining overhead is likely attributable to phase synchronization and I/O. Because the two phases are independent and require no inter-GPU communication, throughput is expected to scale near-linearly with additional GPUs.

Serverless Inference (Modal). To democratize access to high-throughput structure prediction, AlphaFast includes a serverless inference mode via the compute provider Modal. In this mode, users only require a copy of the AlphaFast weights, a billing account, and a directory of inputs. We benchmarked performance on both NVIDIA L40S and H200 GPUs to optimize for cost-efficiency. Remarkably, despite the H200's significantly higher hourly rental rate compared to the L40S, it proved to be the more economical choice due to its superior throughput. The H200 achieved a 2.3× speedup, reducing the inference time to 28.3 seconds and the total cost to $0.035 per target (lower than the L40S). Detailed cost benchmarks are provided in Appendix B.6.

3.2 ALPHAFAST MAINTAINS STRUCTURAL ACCURACY DESPITE REDUCED MSA DEPTH

To evaluate the numerical performance of AlphaFast relative to AF3, we used a Two One-Sided T-test (TOST) (Koch, 1991; Schuirmann, 1987). We applied distinct metrics and criteria to reflect

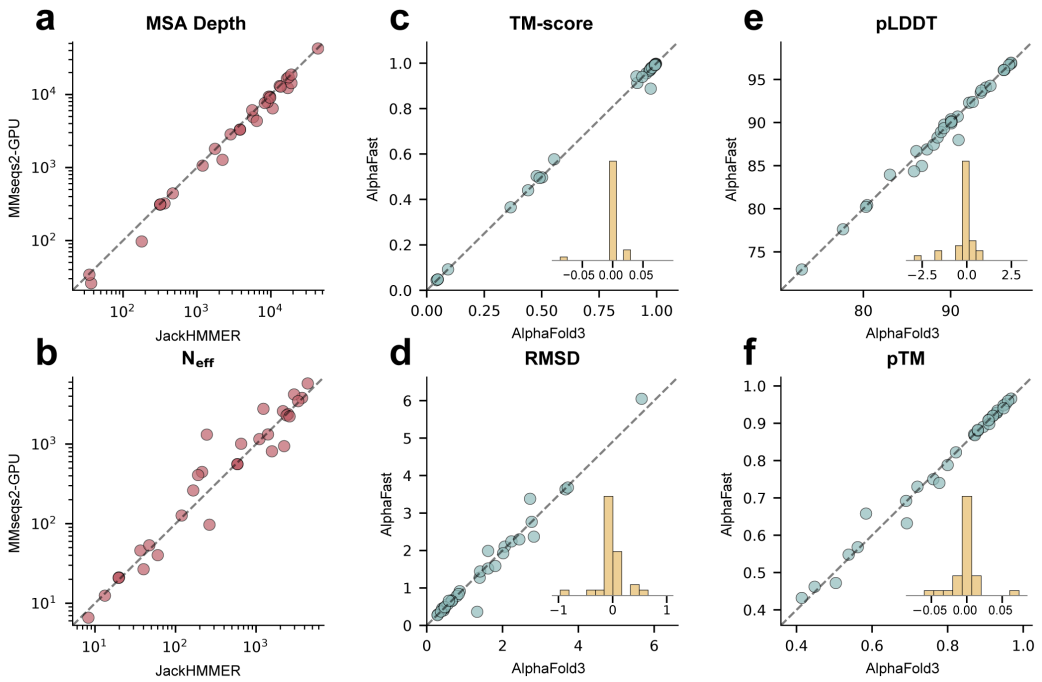

Figure 2: **AlphaFast achieves structural prediction accuracy equivalent to AF3.** (a-b) Log-log scatter plots of MSA Depth and N_{eff}. (c-d) Global structural accuracy (TM-score and RMSD) shows high concordance ($y = x$). Inset histograms display the distribution of pairwise differences ($\Delta = \text{AlphaFast} - \text{AF3}$), showing a sharp peak at zero. (e-f) Model confidence metrics (pLDDT and pTM) demonstrate equivalence within bounds. While MSA Depth is reduced, effective information content (N_{eff}) and structural metrics remain statistically indistinguishable.

the biological nature of each data type. For MSA inputs (N_{eff}, Depth), we performed a Log-Ratio TOST against the standard bioequivalence margin of $[0.80, 1.25]$. For structural outputs (TM-score, RMSD), we assessed the absolute mean difference (Δ) against strict margins (e.g., ± 0.02 for TM-score) (Table S2).

MSA Inputs. Input consistency is visualized in Figure 2a-b. While AlphaFast retrieves fewer raw sequences (MSA Depth GMR $\approx 87.1\%$), it successfully captures the full spectrum of effective evolutionary information (N_{eff} GMR $\approx 107.6\%$). As detailed in the appendix, these values fall well within or exceed the bioequivalence thresholds, confirming that AlphaFast selectively filters redundant sequences without compromising the information density required for prediction.

Structural Outputs. Consequently, this input efficiency translates directly to high-fidelity structural outputs. As shown in Figure 2c-f, the variance between methods is negligible. The mean difference for TM-score was virtually zero ($\Delta \approx +0.002$), and RMSD showed no deviation ($\Delta \approx 0.00$ Å). These results demonstrate that AlphaFast maintains strict structural equivalence to the baseline despite the optimization of input generation.

4 Discussion

In conclusion, AlphaFast overcomes the computational bottleneck of CPU-bound MSA generation in AlphaFold 3 by integrating a GPU-accelerated search via MMseqs2. This approach decouples feature generation from inference, delivering a $22.8\times$ speedup on a single GPU and up to $71.2\times$ in multi-GPU configurations, effectively reducing per-target runtime from nearly 20 minutes to under 10 seconds. Crucially, this acceleration preserves the quality of evolutionary information (N_{eff}) and structural accuracy, ensuring that speed does not come at the cost of model fidelity. To democ-

ratize access to this high-throughput capability, we provide a serverless implementation enabling researchers to fold large cohorts for approximately $\$0.035$ per target.

Despite these advances, AlphaFast has specific limitations. Efficiency gains are maximized for large batches of unique proteins; workflows centered on a single static target—such as repetitive small molecule docking against one receptor—may not realize the same benefits due to lack of caching opportunities. Furthermore, performance on extreme sequence lengths or non-natural inputs remains to be fully characterized. Lastly, we recently added RNA and DNA modalities to AlphaFast, but have not yet characterized wall time savings and statistical equivalence on such inputs.

Looking forward, the strategy of separating search from inference provides a generalizable template for removing bottlenecks in other structural biology models. Ultimately, AlphaFast demonstrates that using the right tools virtually eliminates MSA lookup time, enabling industrial-scale protein design for academic laboratories.

Acknowledgements We acknowledge and thank the Duke Compute Cluster and Pratt Information Technology office for providing and managing computing resources that have contributed to the research results reported within this paper. Furthermore, we thank members of the Romero lab for helpful discussions and figure preparation assistance related to this work. We thank the area chair and anonymous reviewers for their insightful comments that helped improve this paper.

Funding This research was supported by National Science Foundation award 2529581 (to P.A.R.) and National Institutes of Health award 5R01GM150929 (to P.A.R.).

Conflict of interest The authors declare no competing interests.

Data availability The benchmark datasets generated and analyzed during the current study are available in the Figshare repository, `https://doi.org/10.6084/m9.figshare.31343287`.

Code availability The source code for AlphaFast is openly available on GitHub at `https://github.com/RomeroLab/alphafast`.

Author contribution

Benjamin Perry: Conceptualization, Methodology, Software, Investigation, Formal Analysis, Visualization, Writing – Original Draft. **Jeonghyeon Kim:** Methodology, Software, Investigation, Formal Analysis, Visualization, Writing – Original Draft. **Philip Romero:** Conceptualization, Writing – Review & Editing, Supervision, Project Administration, Funding Acquisition.

REFERENCES

Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630(8016):493–500, 2024.

S F Altschul, W Gish, W Miller, E W Myers, and D J Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215(3):403–410, 5 October 1990. doi: 10.1016/S0022-2836(05)80360-2. URL http://dx.doi.org/10.1016/S0022-2836(05)80360-2.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. Jax: composable transformations of python+ numpy programs, 2018.

Benjamin Buchfink, Klaus Reuter, and Hajk-Georg Drost. Sensitive protein alignments at tree-of-life scale using DIAMOND. *Nat. Methods*, 18(4):366–368, 7 April 2021. doi: 10.1038/s41592-021-01101-x. URL https://www.nature.com/articles/s41592-021-01101-x.

UniProt Consortium. Uniprot: a worldwide hub of protein knowledge. *Nucleic acids research*, 47 (D1):D506–D515, 2019.

Chai Discovery, Jacques Boitreaud, Jack Dent, Matthew McPartlon, Joshua Meier, Vinicius Reis, Alex Rogozhnikov, and Kevin Wu. Chai-1: Decoding the molecular interactions of life. *bioRxiv*, 11 October 2024. doi: 10.1101/2024.10.10.615955. URL http://dx.doi.org/10.1101/2024.10.10.615955.

Sean R Eddy. Accelerated profile hmm searches. *PLoS computational biology*, 7(10):e1002195, 2011.

Maria Hauser, Martin Steinegger, and Johannes Söding. MMseqs software suite for fast and deep clustering and searching of large protein sequence sets. *Bioinformatics*, 32(9):1323–1330, 1 May 2016. doi: 10.1093/bioinformatics/btw006. URL http://dx.doi.org/10.1093/bioinformatics/btw006.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

Felix Kallenborn, Alejandro Chacon, Christian Hundt, Hassan Sirelkhatim, Kieran Didi, Sooyoung Cha, Christian Dallago, Milot Mirdita, Bertil Schmidt, and Martin Steinegger. GPU-accelerated homology search with MMseqs2. *Nat. Methods*, 22(10):2024–2027, 18 October 2025. doi: 10.1038/s41592-025-02819-8. URL https://www.nature.com/articles/s41592-025-02819-8.

Gary G Koch. One-sided and two-sided tests and ρ values. *Journal of biopharmaceutical statistics*, 1(1):161–170, 1991.

Sewon Lee, Gyuri Kim, Eli Levy Karin, Milot Mirdita, Sukhwan Park, Rayan Chikhi, Artem Babaian, Andriy Kryshtafovych, and Martin Steinegger. Petabase-scale homology search for structure prediction. *Cold Spring Harbor perspectives in biology*, 16(5):a041465, 2024.

Milot Mirdita, Martin Steinegger, and Johannes Söding. MMseqs2 desktop and local web server app for fast, interactive sequence searches. *Bioinformatics*, 35(16):2856–2858, 15 August 2019. doi: 10.1093/bioinformatics/bty1057. URL http://dx.doi.org/10.1093/bioinformatics/bty1057.

Milot Mirdita, Konstantin Schütze, Yoshitaka Moriwaki, Lim Heo, Sergey Ovchinnikov, and Martin Steinegger. ColabFold: making protein folding accessible to all. *Nat. Methods*, 19(6):679–682, 30 June 2022. doi: 10.1038/s41592-022-01488-1. URL https://www.nature.com/articles/s41592-022-01488-1.

Alex L Mitchell, Alexandre Almeida, Martin Beracochea, Miguel Boland, Josephine Burgin, Guy Cochrane, Michael R Crusoe, Varsha Kale, Simon C Potter, Lorna J Richardson, et al. Mgnify: the microbiome analysis resource in 2020. *Nucleic acids research*, 48(D1):D570–D578, 2020.

Donald J Schuirmann. A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of pharmacokinetics and biopharmaceutics*, 15(6):657–680, 1987.

Martin Steinegger and Johannes Söding. Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature biotechnology*, 35(11):1026–1028, 2017.

Baris E Suzek, Yuqi Wang, Hongzhan Huang, Peter B McGarvey, Cathy H Wu, and UniProt Consortium. Uniref clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, 31(6):926–932, 2015.

Philippe Tillet, Hsiang-Tsung Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pp. 10–19, 2019.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

Jeremy Wohlwend, Gabriele Corso, Saro Passaro, Noah Getz, Mateo Reveiz, Ken Leidal, Wojtek Swiderski, Liam Atkinson, Tally Portnoi, Itamar Chinn, Jacob Silterra, Tommi Jaakkola, and Regina Barzilay. Boltz-1 democratizing biomolecular interaction modeling. *bioRxivorg*, 6 May 2025. doi: 10.1101/2024.11.19.624167. URL http://dx.doi.org/10.1101/2024.11.19.624167.

wwPDB consortium. Protein data bank: the single global archive for 3d macromolecular structure data. *Nucleic Acids Research*, 47(D1):D520–D528, 10 2018. ISSN 0305-1048. doi: 10.1093/nar/gky949. URL https://doi.org/10.1093/nar/gky949.

Andy B Yoo, Morris A Jette, and Mark Grondona. SLURM: Simple linux utility for resource management, 2003. URL http://dx.doi.org/10.1007/10968987_3.

A BENCHMARK DATA

A.1 REFERENCE SEQUENCE DATA & PREPARATION

For benchmarking, we acquire reference sequence databases identical to those fetched in AlphaFold 3 (AF3) via the `fetch_databases.sh` script. The raw sequence data were obtained in FASTA format, including UniRef90 v2022_05 (Suzek et al., 2015), MGnify clusters v2022_05 (Mitchell et al., 2020), BFD first non-consensus sequences Mirdita et al. (2022); Jumper et al. (2021), and UniProt v2021_04 (Consortium, 2019). These represent the four databases searched during MSA lookup in the AF3 pipeline. JackHMMER (Eddy, 2011) can utilize the FASTA (.fa) format without any further pre-processing; However, MMSeqs2-GPU (Kallenborn et al., 2025) requires databases to be converted into a memory-aligned "padded" binary format first. We implemented a two-step conversion process using MMseqs2: first converting FASTA files to binary, indexed databases, and subsequently generating padded sequence databases optimized for GPU memory access. We use the following commands per reference sequence database using 128 threads for speed:

```
# 1. Convert FASTA to standard MMseqs2 database
mmseqs createdb <database-name>.fa <database-name>

# 2. Generate Padded DB for GPU acceleration
mmseqs makepaddedseqdb <database-name> <database-name>_padded>
```

The resulting `*_padded` databases reduce the need for loading massive index files into RAM, instead streamlining data transfer to the GPU during inference. Below, we enumerate the disk space differences between database formats. For MMSeqs2-GPU databases, we sum the space used by the lookup, index, type, header and main database files.

Table S1: **Database Infrastructure Comparison: Raw FASTA vs. GPU-Padded Formats.** The table compares the storage footprint of the original raw sequences against the converted padded databases required for GPU acceleration. While the padded format increases storage usage by 27.3% due to memory alignment, it is strictly required.

Database	Raw FASTA Size (GiB)	GPU Padded Size (GiB)	Role
UniRef90	66.9	78.8	MSA Generation
MGnify	119.7	169.9	MSA Generation
Small BFD	16.9	22.1	MSA Generation
UniProt (Paired)	101.0	116.7	Paired MSA
Total	**304.5**	**387.5**	$\Delta =$ **83 GiB**

On our local system, the data setup took approximately 4 hours for download and conversion. However, we note actual times may vary quite significantly depending on several factors including but not limited to: download speed, system write I/O, host server API call limits, and download agent (`wget` vs. `aria2c`).

A.2 SCALING & NUMERICAL PERFORMANCE DATA SOURCE AND FILTERING CRITERIA

To ensure rigorous evaluation of both the standard and accelerated pipelines, we curated benchmark datasets from the Protein Data Bank (wwPDB consortium, 2018). We selected structures released after September 30, 2021, aligning with the training data cutoff of AF3 to prevent data leakage. The curation process excluded DNA/RNA complexes to focus strictly on protein targets. We applied the following filtering criteria to ensure high-quality structural data:

1. **Resolution:** Structures with resolution ≤ 3.0 Å (including NMR/EM structures).
2. **Chain Length:** Individual chains between 50 and 500 residues.
3. **Total Size:** Total residue count $\leq 1,500$ to fit within standard GPU memory constraints.
4. **Complex Types:** Targets were categorized into three groups: monomers (single chain), protein-ligand (single chain with small molecules), and protein-protein complexes (multimers).

A.3 MULTI-TIERED DATASET STRATEGY

Given the substantial computational resources required for end-to-end AF3 inference, we adopted a multi-tiered dataset strategy. This approach allows us to efficiently evaluate prediction accuracy, single-GPU throughput, and multi-GPU scalability without incurring prohibitive computational costs. All datasets were randomly sampled from the filtered PDB criteria to ensure they share representative structural characteristics (e.g., chain length, complex type distribution) without systematic bias. The exception is the multi-GPU benchmark set, which we relax the training cutoff date.

1. **Accuracy Benchmark ($N = 32$).** For the primary evaluation of MSA quality and structure prediction accuracy, we utilized a subset of 32 randomly sampled targets. The size of this dataset was constrained by the extreme computational cost of the baseline pipeline; generating reference MSAs and structures using the standard AF3 (JackHMMER) pipeline requires approximately 20 minutes per target on our hardware. A larger accuracy benchmark would have necessitated prohibitively long runtimes solely to establish the baseline ground truth.
2. **Single-GPU Benchmark ($N = 512$).** To rigorously evaluate throughput, latency, and cost-efficiency on single-device configurations (e.g., NVIDIA L40S vs. H200), we curated a mid-sized dataset of 512 targets. This volume is sufficient to saturate the GPU and mitigate cold-start overheads, providing a robust estimate for "production-grade" performance and serverless cost calculations.
3. **Multi-GPU Benchmark ($N = 2,048$).** For the most demanding stress tests, including multi-GPU scalability and storage I/O bottlenecks in an High Performance Computing (HPC) environment, we constructed a large-scale dataset of 2,048 targets. This tier validates the stability of the multi-gpu architecture under heavy load and demonstrates the system's capability to handle industrial-scale screening workflows. For this set only, we relax the training data cutoff to structures released after September 30, 2016 as our sampling strategy did not yield enough inputs to test this scale. We only utilize this large-scale set to assess inference speed and not numerical performance.

B ARCHITECTURE & EVALUATION METHODOLOGY

B.1 HOMOLOGY AND TEMPLATE SEARCH

To benchmark the performance of the accelerated pipeline, we established a controlled comparison between the default AF3 data generation process and our proposed MMseqs2-GPU implementation. Both pipelines search against identical sequence databases with attempts to match parameters to ensure comparable MSA composition.

1. **Baseline Pipeline (JackHMMER/hmmsearch).** The default pipeline utilized HMMER 3.4 for MSA generation. Four parallel subprocess calls query against sequence databases with the following per-database sequence limits:
   (a) **UniRef90.** Maximum of 10,000 sequences
   (b) **MGnify.** Maximum of 5,000 sequences
   (c) **Small BFD.** Maximum of 5,000 sequences
   (d) **UniProt.** Maximum 50,000 sequences (for taxonomic pairing)

   Search parameters were configured with a single iteration ($N = 1$), E-value threshold of 10^{-4}, and aggressive pre-filters ($F1 = 5 \times 10^{-4}$, $F2 = 5 \times 10^{-5}$, $F3 = 5 \times 10^{-7}$) to accelerate queries against large metagenomic databases. For template retrieval, HMM profiles constructed from the merged MSA were searched against the `pdb_seqres` database using `hmmsearch`, configured with relaxed pre-filters ($F1 = F2 = F3 = 0.1$) and an E-value threshold of 10^{-3}.
2. **Accelerated Pipeline (MMseqs2-GPU).** The experimental pipeline utilized MMSeqs2 with GPU-accelerated pre-filtering (`--gpu 1`) at sensitivity $s = 7.5$. To maximize GPU utilization while avoiding memory contention, we implemented a pipelined execution strategy:

(a) **Sequential GPU Search.** Database searches execute sequentially against padded databases (UniRef90, MGnify, Small BFD, UniProt), with each `mmseqs search` command using alignment backtraces (`-a`) for downstream MSA generation. Sequence limits match the baseline pipeline with $E \leq 10^{-4}$.

(b) **Parallel CPU Post-Processing.** Upon completion of each GPU search, post-processing tasks (`mmseqs result2msa` with `--msa-format-mode 2` followed by `mmseqs unpackdb` and a custom conversion to the expected A3M output) are offloaded to a thread pool executor, allowing CPU-bound conversion to overlap with subsequent GPU searches.

(c) **MSA Consolidation.** Individual database MSAs are merged with feature-equivalence deduplication to remove redundant sequences across databases.

For template retrieval, `mmseqs search` queries the `pdb_seqres` database ($E \leq 10^{-3}$), filtering for minimum alignment coverage of 40%. Template metadata (PDB ID, chain, residue ranges) are parsed from result alignments to map hits to mmCIF structures, excluding entries released after September 30, 2021 for fair comparison.

B.2 Folding Method & Template Filtering

Structure prediction was performed using the official **JAX-based** (Bradbury et al., 2018) implementation of AF3 within a custom Docker environment (Singularity was used on the HPC environment instead). To capture the model's stochasticity and robustly estimate performance, we generated a total of 25 predictions per target, derived from 5 distinct random seeds with 5 diffusion samples per seed. For scaling, we reduce this to 5 predictions per target, derived from 1 random seed with 5 diffusion samples. Inference was executed with standard hyperparameters, including 10 recycling iterations and Triton-based flash attention (Tillet et al., 2019).

To evaluate performance under controlled evolutionary contexts, we implemented two distinct template regimes by pre-filtering the template inputs before inference:

1. **Relaxed Regime (ID $< 90\%$).** A standard setting that filters out templates with $\geq 90\%$ sequence identity. This setup removes self-templates to prevent data leakage while retaining close structural homologs.
2. **Stringent Regime (ID $< 30\%$).** A stress test that filters out all templates with $\geq 30\%$ sequence identity. This regime simulates "orphan" targets or de novo designs where no homologous structural information is available.

Template filtering was applied using a custom pre-processing script to generate separate input directories, which were then dynamically selected at runtime via the `--precomputed_templates_a3m_path` argument. In both regimes, a maximum of 4 top-ranked templates were passed to the model.

B.3 Statistical Analysis

To assess input feature quality, we measured MSA depth (raw count) and the effective number of sequences (N_{eff}). Structural prediction accuracy was quantified using TM-score and RMSD relative to ground truth. Model confidence was assessed using pLDDT and pTM.

Statistical comparisons were performed using the `SciPy` library (Virtanen et al., 2020). Differences in metrics were initially evaluated using paired t-tests. To rigorously assess the functional equivalence of the two pipelines, we employed a **Two One-Sided Test (TOST)** (Koch, 1991) procedure with regime-specific criteria:

1. **Input Equivalence (Log-Ratio):** For MSA metrics (N_{eff}, Depth), which follow log-normal distributions, we performed a Log-Ratio TOST. We applied the standard bioequivalence margin of $[0.80, 1.25]$ (Schuirmann, 1987), a rigorous threshold conventionally used to establish statistical equivalence in biological datasets.
2. **Output Equivalence (Absolute Difference):** For structural metrics (TM-score, RMSD, pLDDT), we applied *strict absolute margins* to ensure high fidelity. The equivalence

bounds (ϵ) were set based on community standards: $\epsilon = 0.02$ for TM-score and pTM, $\epsilon = 0.5$ Å for RMSD, and $\epsilon = 1.0$ for pLDDT.

A p-value < 0.05 in the TOST procedure indicates that the performance difference falls significantly within these specified equivalence bounds.

B.4 Template Retrieval Sensitivity and Structural Robustness

While the main text focuses on the functional equivalence of the final structures, we further analyzed the intermediate template retrieval steps and performance under a stringent evolutionary scenario (Figure S1).

Template Retrieval Sensitivity. We compared the ability of each pipeline to identify valid structural templates from the PDB70 database. Due to the heuristic nature of the accelerated search, we observed a trade-off in retrieval sensitivity. As shown in Figure S1a, the baseline hmmsearch pipeline retrieved templates for 27/32 targets, while the MMseqs2-GPU pipeline identified templates for 23/32 targets. Furthermore, Figure S1b indicates that for certain targets, MMseqs2-GPU retrieved templates with lower sequence identity compared to the exhaustive profile-based search of hmmsearch (points falling below the diagonal).

Structural Robustness Under Stringent Filtering ($< 30\%$). Crucially, however, this reduction in template quantity and quality did not propagate to the downstream prediction accuracy. To rigorously test this robustness, we evaluated performance under a "Stringent Regime" where all templates with $> 30\%$ identity were excluded. As shown in Figure S1c-f, the global accuracy metrics remained strictly aligned despite the differences in template inputs. Quantitative analysis confirmed this equivalence: the mean difference in TM-score was negligible ($\Delta = -0.0015$), and RMSD variation was minimal ($\Delta = +0.0037$ Å), both falling well within strict equivalence bounds. This result strongly suggests that AF3 is robust to minor variations in template assignment and that AlphaFast provides sufficient evolutionary signal to maintain high-fidelity modeling even when template retrieval is suboptimal.

B.5 Statistical Bioequivalence Verification

To quantify the agreement between AlphaFast and the baseline JackHMMER pipeline, we performed a Two One-Sided Test (TOST) as detailed in Section B. Table S2 presents the comprehensive results of this assessment on the Accuracy Benchmark set ($N = 30$).

The analysis confirms that AlphaFast is statistically bioequivalent to the standard pipeline. For input features, while the heuristic search of MMseqs2 yields a Geometric Mean Ratio (GMR) of 87.1% for raw MSA depth, the effective sequence count (N_{eff}) is preserved with a GMR of 107.6%, indicating no loss of evolutionary signal. Consequently, all downstream structural metrics (TM-score, RMSD, pLDDT, and pTM) fall well within the strict equivalence margins, demonstrating that the speedup is achieved without compromising modeling accuracy.

Table S2: **Bioequivalence Assessment.** Input quantity was assessed using Geometric Mean Ratio (GMR) against standard bioequivalence limits (0.80–1.25). Structural fidelity was assessed via absolute mean difference (Δ). Note that AlphaFast preserves effective information (N_{eff}) despite reducing raw depth.

Metric	Comparison Metric	Equivalence Margin	Result
Input Quantity (Log-Ratio TOST)			
MSA Depth	GMR $= 87.1\%$	$[0.80, 1.25]$	Equivalent
N_{eff}	GMR $= 107.6\%$	$[0.80, 1.25]$	Equivalent
Output Quality (Absolute Difference TOST)			
TM-score	$\Delta = +0.002$	± 0.02	Equivalent
RMSD (Å)	$\Delta = 0.00$	± 0.5Å	Equivalent
pLDDT	$\Delta = -0.16$	± 1.0	Equivalent
pTM	$\Delta = -0.002$	± 0.02	Equivalent

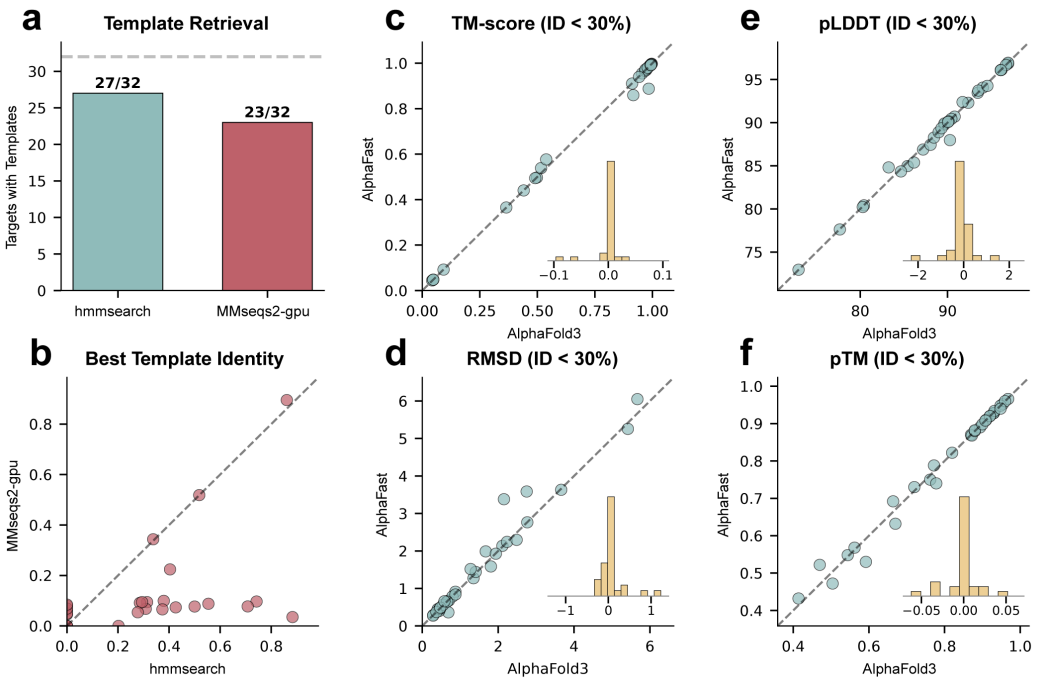

Figure S1: **Template retrieval performance and structural accuracy under the Stringent Regime ($<$ 30% Identity).** (a) Template retrieval success rate. MMseqs2-GPU shows slightly reduced sensitivity (23/32) compared to the exhaustive hmmsearch (27/32). (b) Comparison of the sequence identity of the best template found by each pipeline. Some targets show lower identity matches with MMseqs2-GPU (below diagonal). (c-f) Structural accuracy comparisons under the **Stringent Regime**. Remarkably, despite the reduction in template retrieval performance, MMseqs2-GPU maintains equivalent TM-score (c), RMSD (d), pLDDT (e), and pTM (f) to the baseline, demonstrating the robustness of the accelerated pipeline.

B.6 Serverless Inference Performance and Cost Efficiency

To ensure broad accessibility for researchers lacking dedicated HPC resources, we evaluated the cost-effectiveness of deploying AlphaFast on a serverless infrastructure (Modal). We benchmarked the performance of single-GPU instances—NVIDIA L40S versus NVIDIA H200—processing a cohort of 512 targets to simulate a production-scale high-throughput workload.

Table S3 summarizes the latency and cost metrics. When considering the GPU rental rate in isolation, the H200 incurs a slightly higher cost per input ($0.0357) compared to the L40S ($0.0349). However, because serverless providers bill for the concurrent usage of all provisioned resources (CPU and RAM), the H200's superior throughput—reducing average processing time from 64.3s to 28.3s ($\sim 2.3\times$ speedup)—leads to a significant reduction in auxiliary resource costs.

Consequently, the total cost per input (GPU + CPU + RAM) is lower on the H200 ($0.0385) than on the L40S ($0.0413). These results demonstrate that high-tier hardware like the H200 provides a dual advantage for serverless AlphaFast deployment, offering both drastic wall-clock time savings and superior overall cost efficiency for large-scale inference tasks.

We utilized Modal as the inference provider and calculated costs based on rates at the time of writing (February 2026).

B.7 Code Availability

The source code for AlphaFast is openly available on GitHub at `https://github.com/RomeroLab/alphafast`.

Table S3: **Cost and Performance Comparison of Serverless Deployment (Modal).** Benchmarks were conducted using a cohort of 512 inputs. While the GPU-only cost is marginally higher for the H200, its 2.3× speedup reduces total system resource consumption, making it the more economical choice.

Hardware	**Time/Input (s)**	**GPU Cost /Input ($)**	**Total Cost /Input ($)**	**Total Cost (512 inputs)**	**Performance**
NVIDIA L40S (Single)	64.3	0.0349	0.0413	$21.14	Baseline
NVIDIA H200 (Single)	**28.3**	**0.0357**	**0.0385**	**$19.71**	2.3× **Speedup**

B.8 Full Experimental Data

The benchmark datasets generated and analyzed during the current study are available in the Figshare repository, `https://doi.org/10.6084/m9.figshare.31343287`.