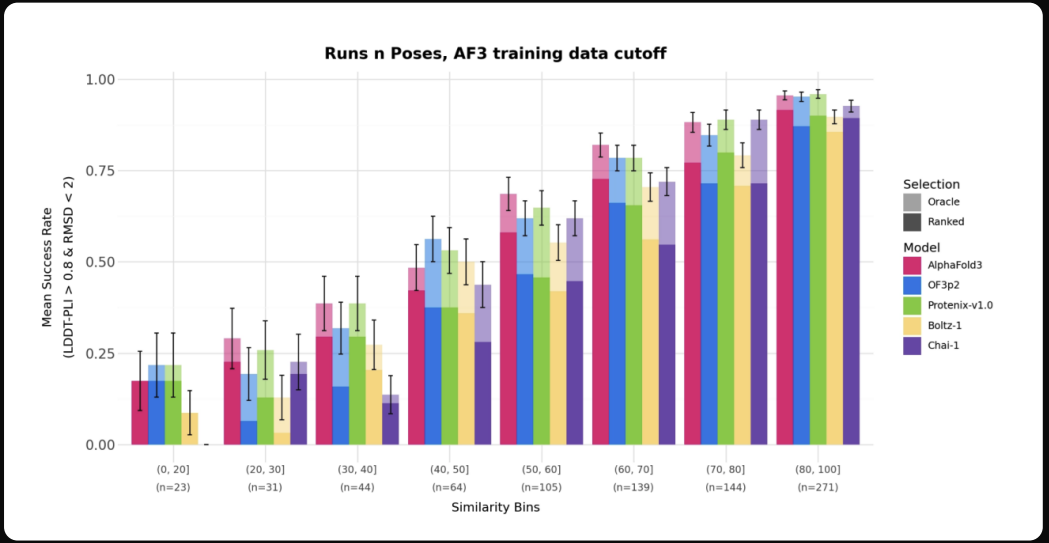
Lambda

Lambda x OpenFold

Enabling more scalable and efficient OpenFold3 fine-tuning

Jan Domański, Jianwen Xie

Overcoming the generalization gap



Co-folding models generalize poorly beyond their training data. On the Runs N' Poses benchmark (left), success rate falls as similarity to the training set decreases.

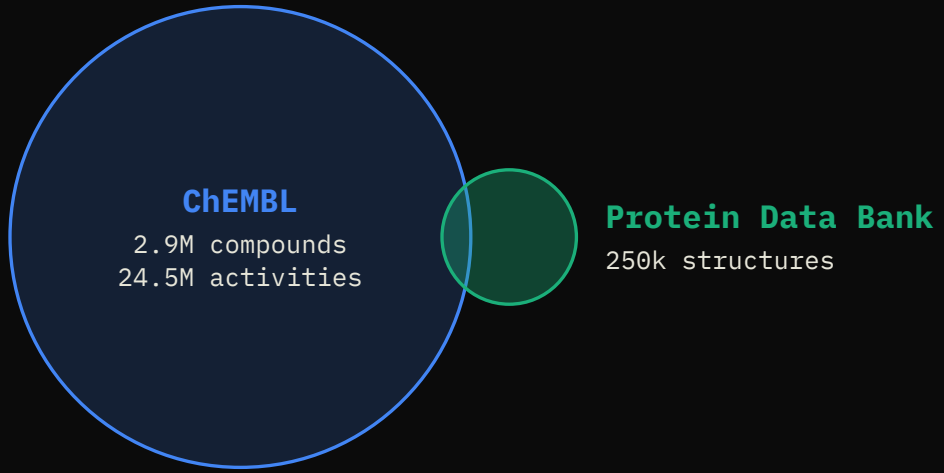
Fine-tuning shifts the model from left to right along the x-axis.

We developed data-augmentation strategies and performance optimizations to make fine-tuning workloads more efficient.

Training: datasets

Overlap: same target, near-neighbor ligand

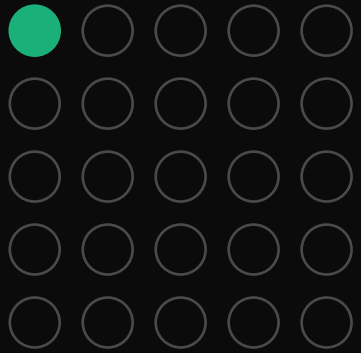
Circle areas to scale · ChEMBL 37



The synthetic complexes come from the overlap.

Kinases: a key protein class

STRUCTURALLY MAPPED



1 in 25

PDB entries contain a protein-kinase domain

10.6k

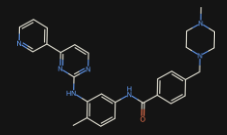
kinase entries, 8.6k of them human

One conserved ATP pocket, solved again and again with different ligands: dense training data for co-folding.

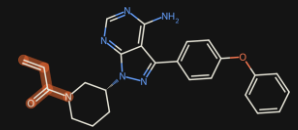
CLINICALLY VALIDATED

94

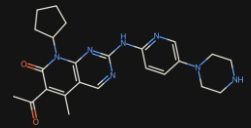
FDA-approved protein-kinase inhibitors
~80 of them for cancer



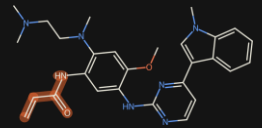
Imatinib — BCR-ABL · CML — **$4.7B** peak, 2015



Ibrutinib — BTK · CLL — **$4.9B** 2025



Palbociclib — CDK4/6 · breast — **$4.1B** 2025



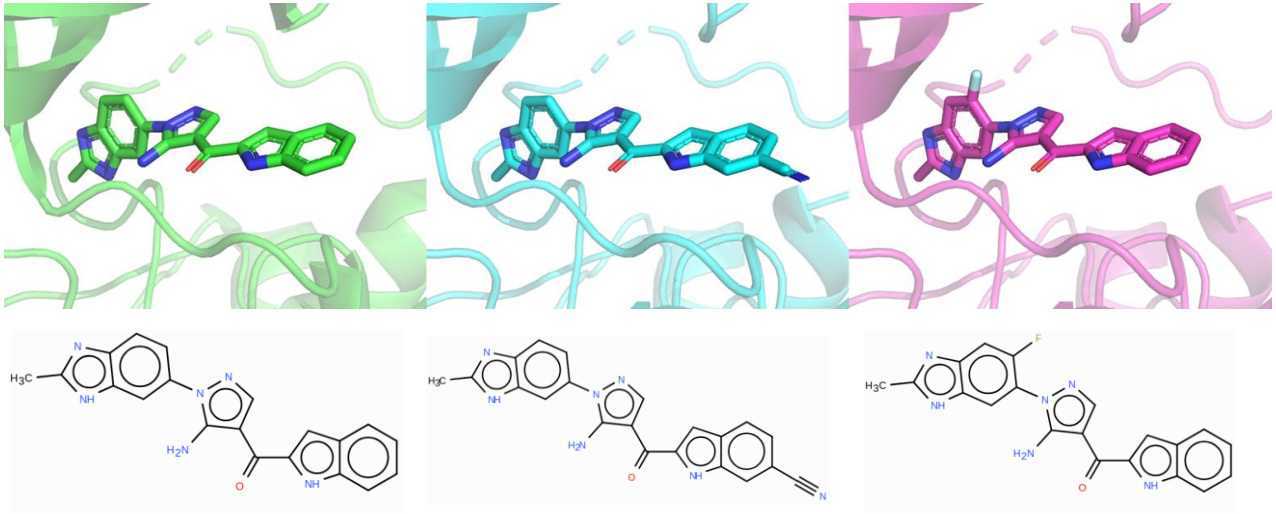
Osimertinib — EGFR · NSCLC — **$7.3B** 2025

Sources: RCSB PDB query, 21 Oct 2026 (experimental entries; InterPro IPR000719/IPR001245 or Pfam PF00069/PF07714) · Roskoski, Pharmacol Res 2026 · FY2025 reports: AstraZeneca, AbbVie (US) + J&J (ex-US), Pfizer · Gleevec: 2015 worldwide sales. Orange halo = covalent acrylamide warhead.

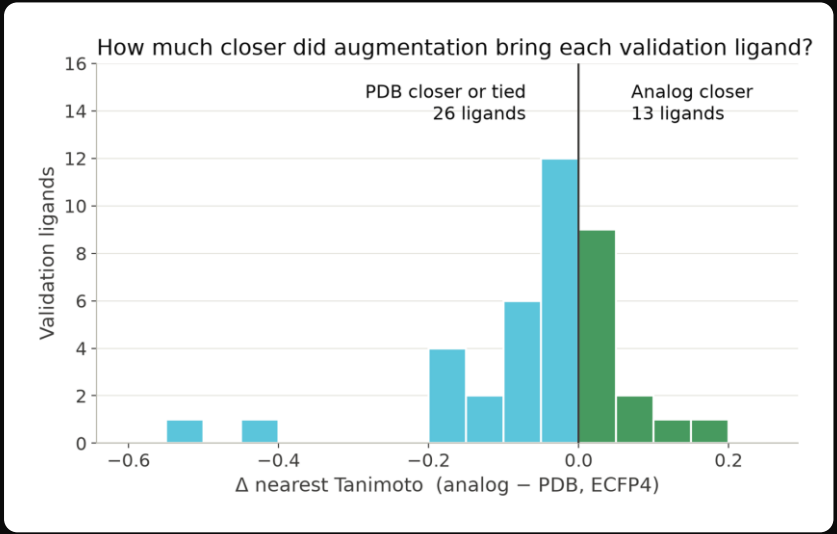
Augmentation strategy

5B7V is in OpenFold3's training set; we build synthetic ‘analogs’ in its crystal pose by adding small ChEMBL-derived R-groups, e.g. a **nitrile** or a **fluorine**.

This makes the dataset denser, adding local information. In drug-discovery programs, binding/activity data is typically 10–100× more abundant than structural data.

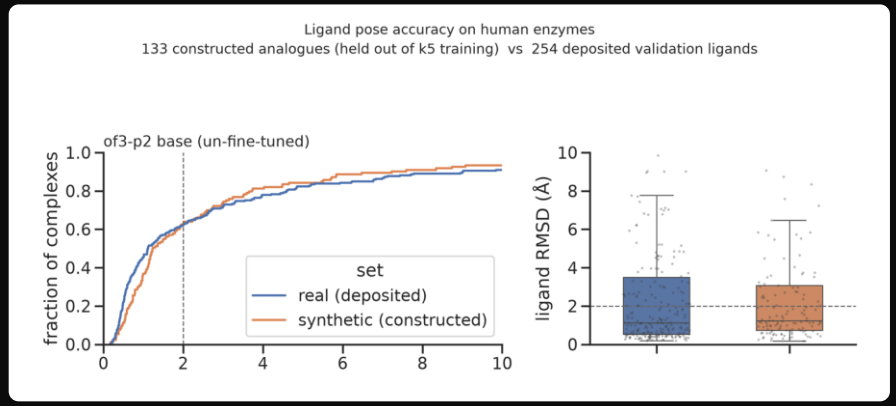


Controlling for augmentation leakage



Augmentation gives validation ligands new nearest neighbors in training: for 13 of 39, an analog is now closer than any PDB ligand.

Worst case: ~0.20 Tanimoto closer than its previous nearest neighbor (ECFP4, r = 2, 2048 bits).



Pose accuracy, base model: deposited vs constructed validation ligands are posed with similar accuracy.

Experiment A: match every data point

5.2k

PDB KINASE COMPLEXES

5.9k

CHEMBL-DERIVED SYNTHETIC

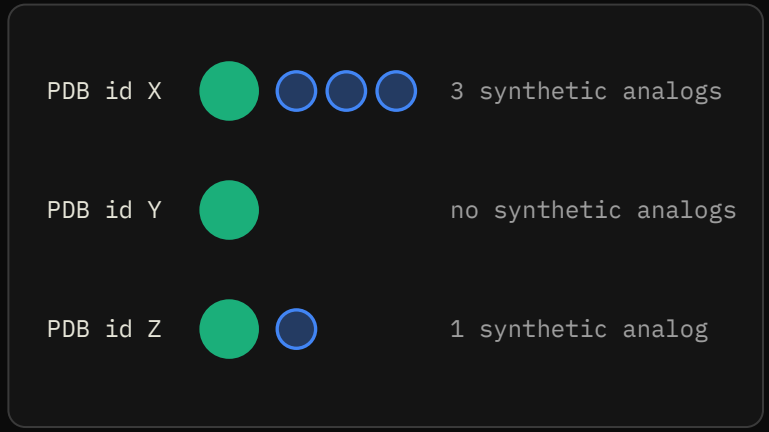
11.1k

COMBINED TRAINING SET

Amplification is uneven. A handful of PDB entries carry 50+ ChEMBL analogs; most carry one or two, and many produce none at all (“analog-less PDBs”).

Monomeric protein kinases only, each with a single drug-like ligand. Every synthetic ligand differs from its PDB parent by at most 3 heavy atoms and is backed by a measured binding record against the same target.

Every synthetic complex passes a protein–ligand clash screen. Strained and clashing poses are discarded rather than repaired.



Experiment B: only match when analogs exist

1.5k

PDB KINASE COMPLEXES

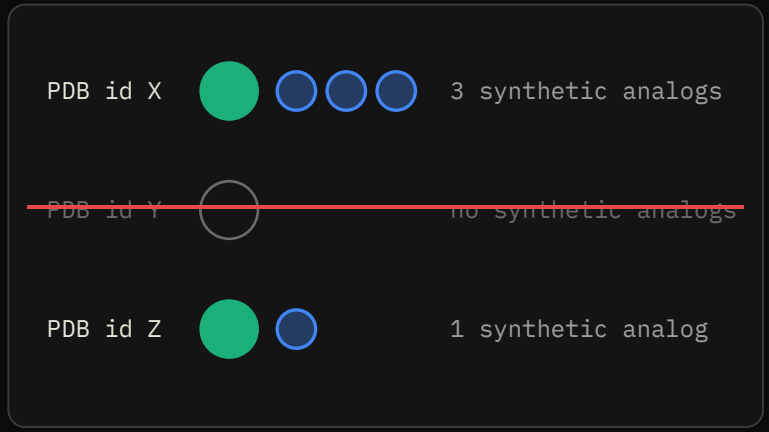
5.9k

CHEMBL-DERIVED SYNTHETIC

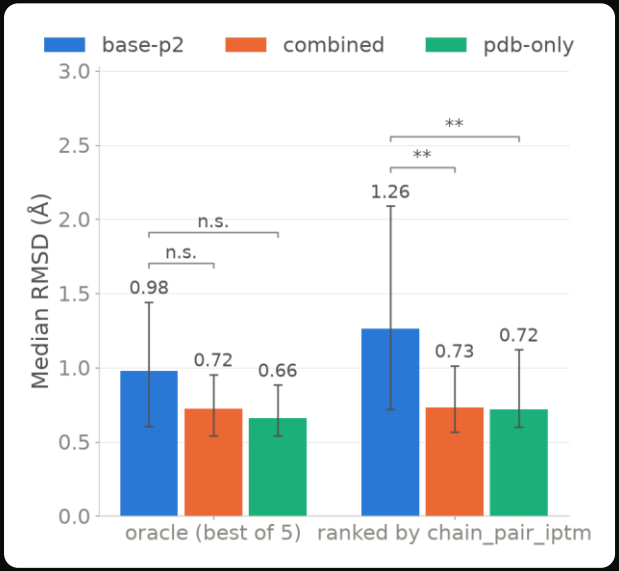
7.4k

COMBINED TRAINING SET

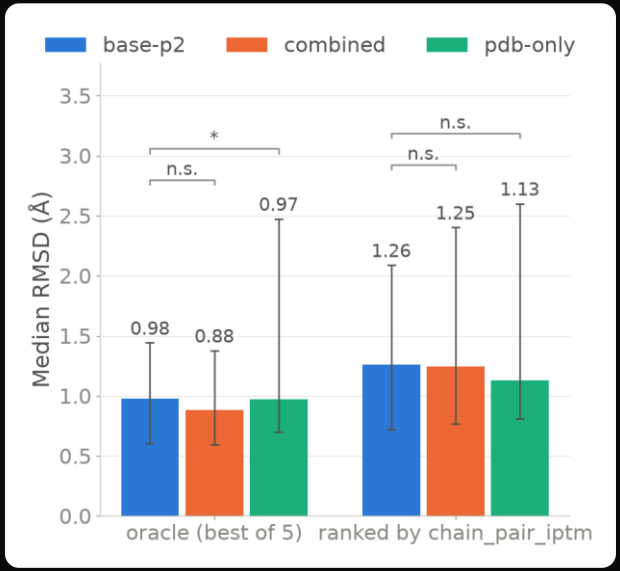
A fairer test of the augmentation's impact, at a cost: PDB entries without analogs are dropped, discarding ~70% of the ground-truth complexes (5.2k → 1.5k).



Fine-tuning improves model performance

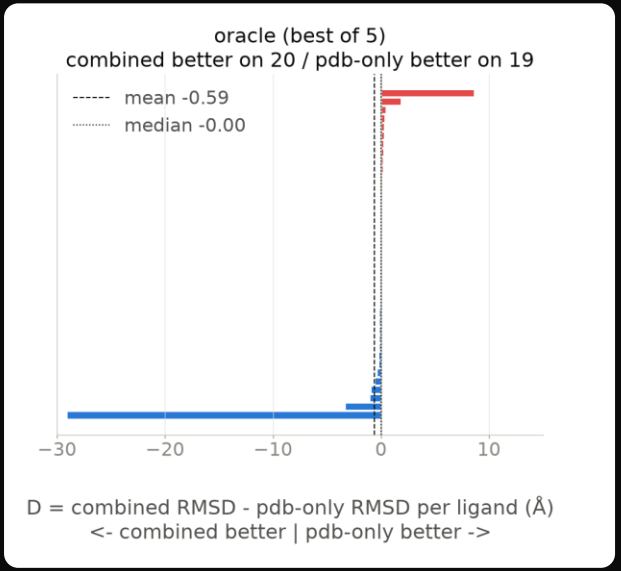


Experiment A: all training data points kept, including those without synthetic analogs. Fine-tuning helps; synthetic data adds no advantage.

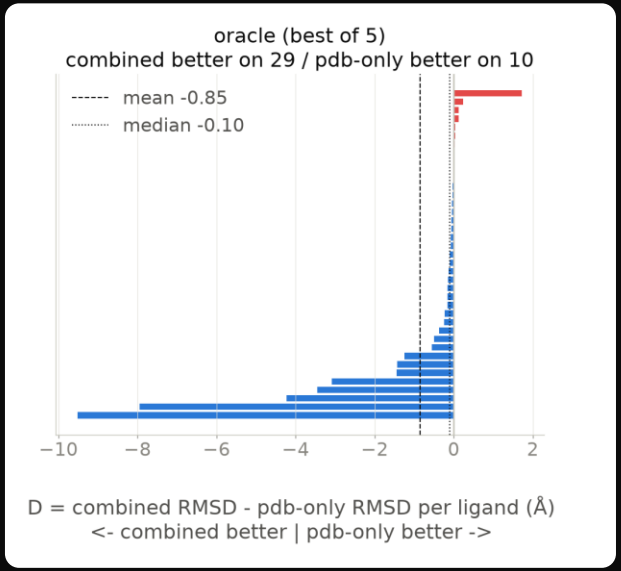


Experiment B: only data points with synthetic analogs kept, trading performance for a more balanced augmentation.

Per ligand: combined vs PDB-only



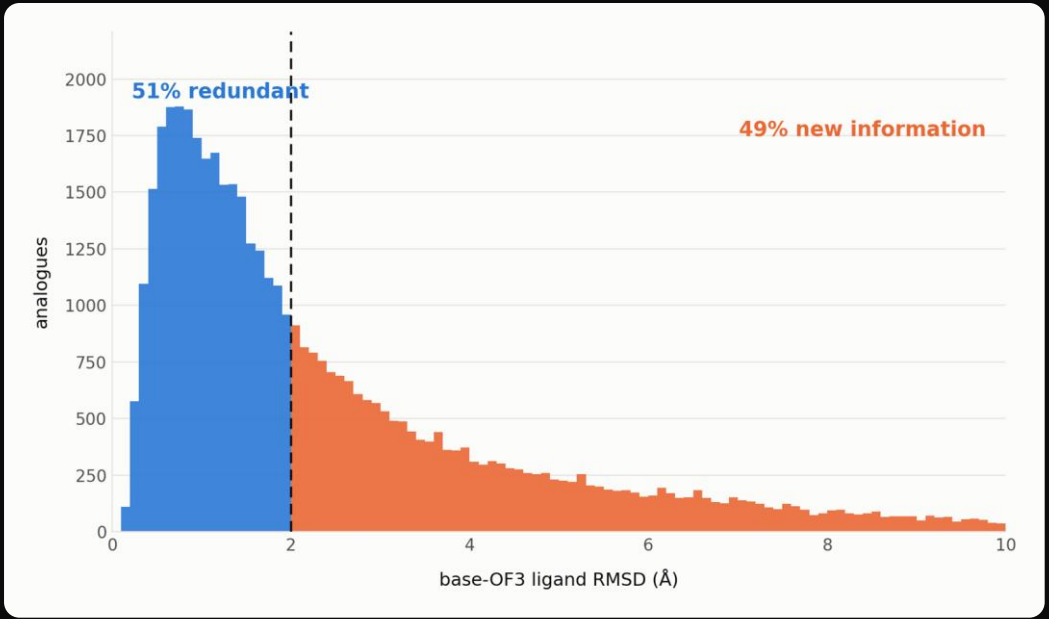
Experiment A: per-ligand RMSD, combined − PDB-only. A near tie: 20 vs 19 wins, median 0.00 Å; one outlier drives the mean.



Experiment B: combined is better on 29 of 39 ligands (median −0.10 Å): here the synthetic augmentation has a clear effect.

Co-folding data augmentation

Next steps



The base model already poses most synthetic analogs well **(blue: 51% under 2 Å)**, before any fine-tuning.

We made the dataset denser, but the synthetic arm added relatively little new information.

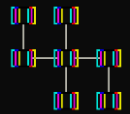
Could the key be to train on the synthetic poses the base model gets wrong **(orange: 49% above 2 Å)**?

Lambda enables more scalable and efficient OpenFold3 fine-tuning

1. Compute

Scalable NVIDIA GPU infrastructure

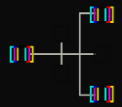
Provided a 48× H100 cluster with InfiniBand interconnects for distributed OpenFold3 fine-tuning and benchmarking.



2. Orchestration

Slurm-based distributed training

Deployed a customized Slurm environment for reliable multi-GPU and multi-node job scheduling and management.



3. Fine-tuning flexibility

LoRA/Frozen-confidence training

Expanded OpenFold3 fine-tuning capabilities with LoRA adapters and flexible module freezing, enabling more efficient training configurations.



4. Memory efficiency

Reduced training memory overhead

Optimized metric logging to reduce memory usage by 5%.



5. Speed & stability

Optimized triangle attention kernel

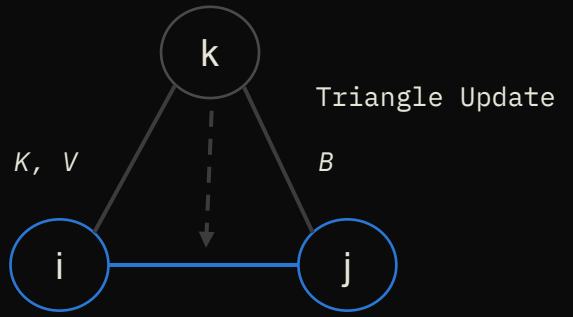
Co-developed TurboPairFormer to accelerate OpenFold3 training by up to 1.73×, with improved numerical accuracy and reproducibility.



Triangle attention: a core operation in OpenFold3

$$\text{Attention}(Q, K, V, B)_{ij} = \sum_k \text{Softmax}_k \left(\frac{Q_{ij}^T K_{ik}}{\sqrt{d}} + B_{jk} \right) V_{ik}$$

A three-token relationship



The pair (i, j) is updated by aggregating information through a third token k

1. **Captures pairwise geometry**

   Three-token interactions

   Update pair representations to reason about relationships among biomolecular tokens.

2. **A major training bottleneck**

   Cubic computational complexity

   Triangle attention scales as O(N^3) with the number of tokens.

3. **Performance and stability matter**

   Reliable forward and backward passes

   Efficient computation and accurate, reproducible gradients support stable training.

The opportunity: accelerate a core operation without sacrificing numerical reliability

TurboPairFormer: an optimized triangle attention kernel

1. Faster
Single probability recomputation
reuse each reconstructed tile

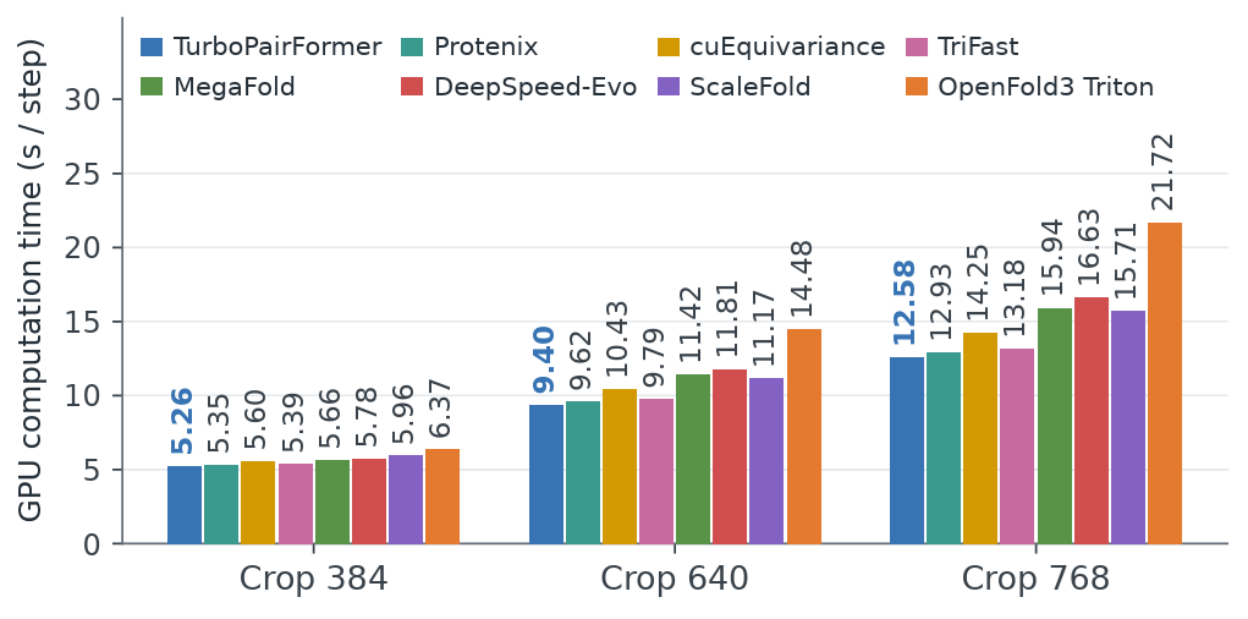
2. Reproducible & stable
Fixed accumulation order
reproducible gradient sums

3. Improved numerical accuracy
Output-residual compensation
save the rounding remainder

Core design:

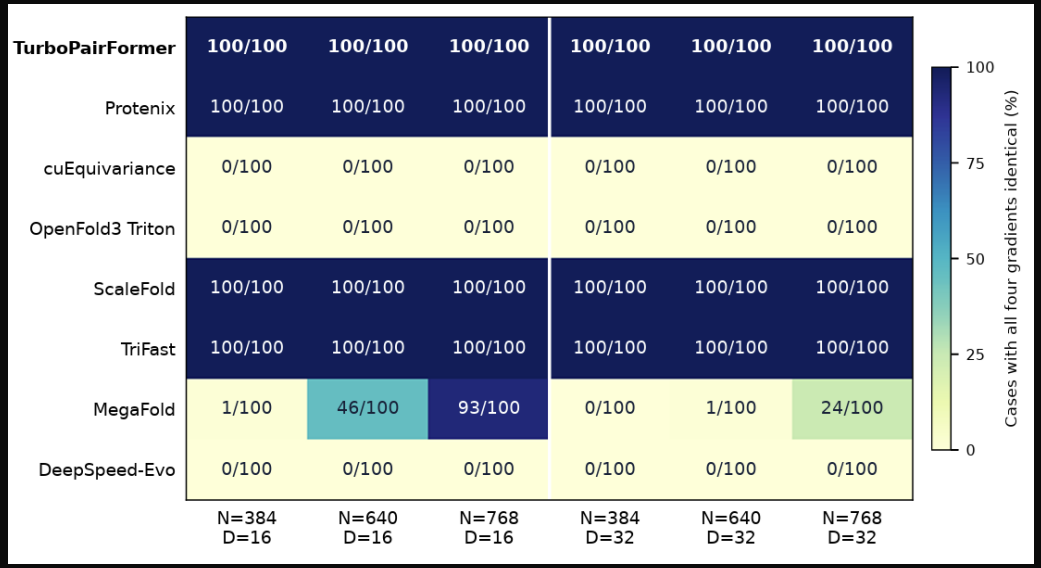
1. Recomputes each probability tile **only once** and reuses it for the query, key, value, and pair-bias gradients for backward computation
2. **Deterministic** backward (no atomicAdd, fixed order, reproducible gradient sums) for stability
3. **Saves the BF16 rounding residual** and uses it to compute the backward correction more accurately in FP32

TurboPairFormer: faster training



All kernels are tested on 16× H100 GPUs. TurboPairFormer has the lowest GPU computation time per optimizer step of the eight kernels tested. Achieves 1.73× speedup with crop_size = 768, compared with OpenFold3's Triton kernel.

TurboPairFormer: reproducible

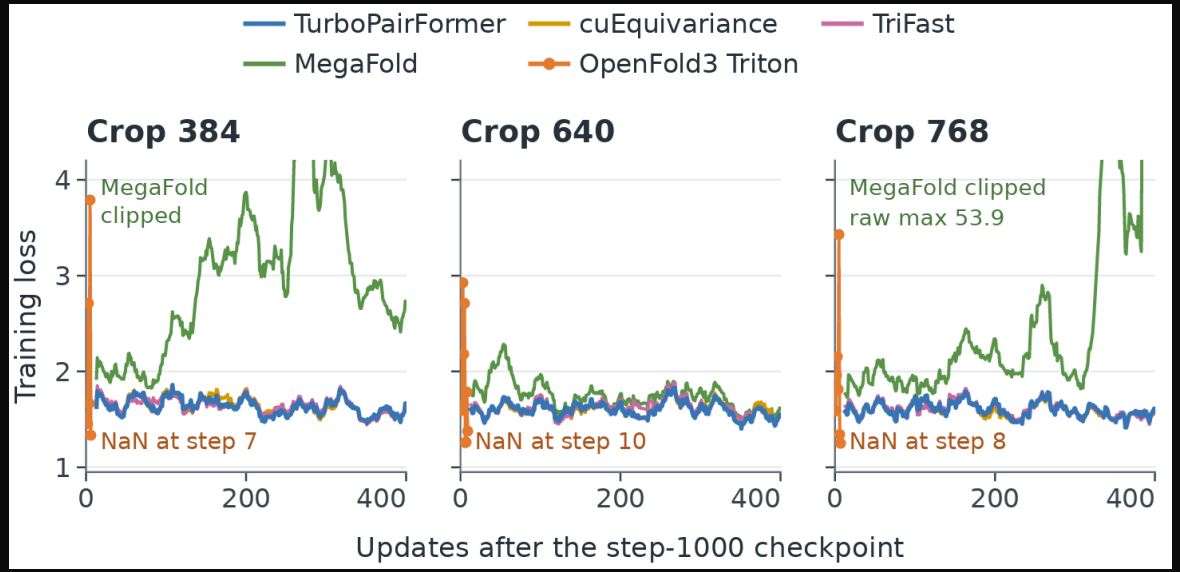


Each setting uses 100 fixed inputs, with 5 calls per input on one H100.

A case passes only when all 4 BF16 gradients (dQ, dK, dV, dB) are bitwise identical across all 5 calls.

All four gradients of TurboPairFormer are bitwise identical across five repeated calls in all 600 input cases.

TurboPairFormer: stable training

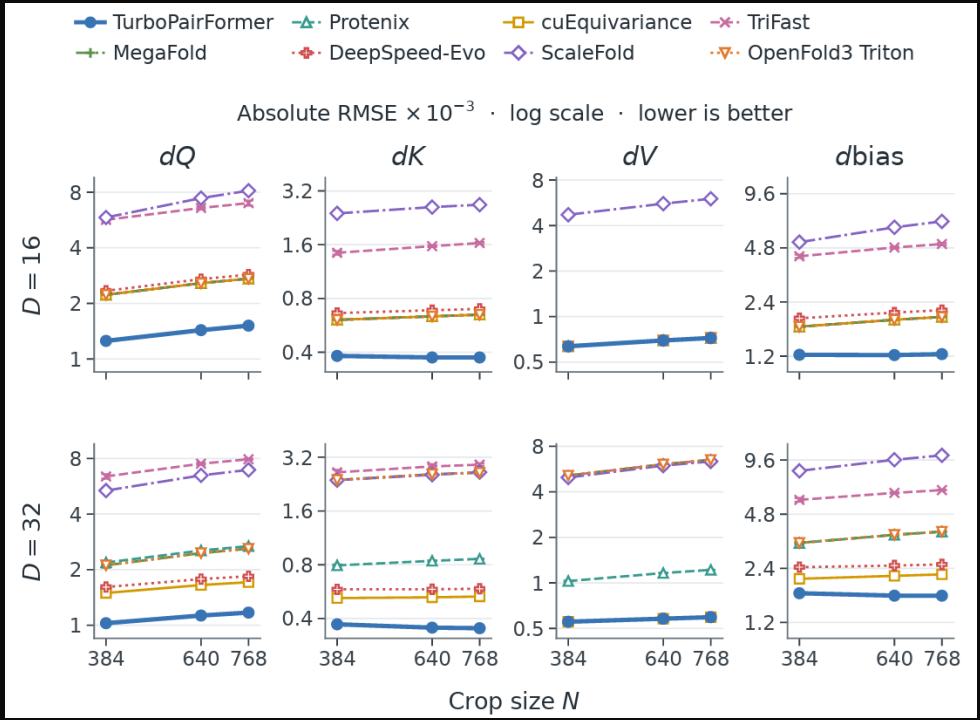


Experimental setup: Different kernels resume training from the same checkpoint on an H100 GPU, using identical random seeds and training examples.

Visualization: Curves represent training loss over time.

Key finding: TurboPairFormer demonstrates stable training, without significant loss fluctuations or NaN values.

TurboPairFormer: improved backward accuracy



All methods receive the same BF16 inputs. An FP64 reference is evaluated on the same BF16 inputs.

TurboPairFormer achieves the lowest mean query, key, value, and pair-bias gradient RMSE against the FP64 reference.

Residual compensation mitigates the precision loss from output rounding, reducing these RMSE values by 28–47% in controlled ablations.

Takeaways

~30%
Fine-tuning performance

Lower median ligand RMSD on the kinase validation set after class-specific fine-tuning, OpenFold team result

1.73×
Faster training

Faster OpenFold3 training vs the original Triton backend, 16× H100, crop 768

28–47%
Higher numerical accuracy

Lower gradient error with output-residual compensation

Bitwise
Stable training

Identical gradients across runs from a deterministic backward pass, stable training

PyPI
Open-source release

Available as a PyPI package, making the optimized kernels accessible to the protein folding research community.