SIMPLEFOLD2: EFFICIENT BIOMOLECULAR CO-FOLDING WITH TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Biomolecular co-folding models jointly predict the 3D atomic structure of systems where a protein interacts with other proteins, ligands, and nucleic acids. Most modern co-folding models build on the AlphaFold lineage, which designed bespoke architectures like pair representations, triangle updates, and deep MSA encodings. In this work, we ask if co-folding models require these domain specific designs or could instead be framed as general purpose generative models from the ground up. We introduce SimpleFold2, a flow matching model for co-folding built exclusively using general purpose transformers. Similar to text-to-image models, we use pre-trained protein language models (pLMs) to embed sequence information, and we do not incorporate triangle updates, template stacks, recycling, or equivariant modules. Because every layer in SimpleFold2 is ordinary attention or an MLP, it inherits the favorable behavior of modern transformers. We scale SimpleFold2 to 3B parameters and demonstrate strong performance on protein-ligand co-folding and competitive performance on monomers. We then show how the efficiency of SimpleFold2 leads to strong test-time scaling behavior. Overall, SimpleFold2's architecture ensures it is highly efficient on modern hardware: it uses $\approx 0.3\times$ the memory of AlphaFold-style models, runs $10\times$ faster for 1700-residue complexes, and can even fold a 4960-residue complex on a single NVIDIA H100 GPU in 39s using only 62 GB of memory, a task well beyond most comparable methods.

1 INTRODUCTION

Determining the 3D atomic structure of biomolecular complexes, where a protein interacts with other proteins, nucleic acids, or small-molecule ligands, is a critical challenge in frontier computational biology. Many downstream tasks in biology and drug discovery, from high-throughput screening to de novo design, are directly dependent on knowledge of the atomic structure (for Cell Biology, 2002; Watson et al., 2023; Lyu et al., 2019). While experimental methods like X-ray crystallography, cryo-electron microscopy, or nuclear magnetic resonance are the gold standard (Berman et al., 2000), they can entail months to years of experimental effort (Tunyasuvunakool et al., 2021). To address this cost, several attempts have been made to train ML models on publicly available experimental data to estimate protein structures from their corresponding amino-acid sequence information. Running inference on these models is much more cost efficient than experimentally determining biomolecular structures even when taking into account training cost.

Most notably, the AlphaFold series of works (Jumper et al., 2021; Abramson et al., 2024b) obtained strong performance on monomer and complex prediction by carefully designing both network architectures and training objectives specifically for biomolecular structure prediction. The original AlphaFold3 architecture has now been adopted in several co-folding models (both open and closed source) such as Protenix (Team et al., 2025), Boltz-1 (Wohlwend et al., 2024), Chai-1 (Boitreaud et al., 2024), RoseTTAFold (Baek et al., 2021), and SeedFold (Zhou et al., 2025), which differ slightly in data and training recipe but share the same overall architectural design and training objective. In particular, these AlphaFold-style architectures explicitly encode pairwise interaction tensors and process them via very deep encoders with several blocks of Pairformer layers Abramson et al. (2024a) with *triangular* updates, which encode a geometric inductive bias but scale cubically with sequence length (Abramson et al., 2024a).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

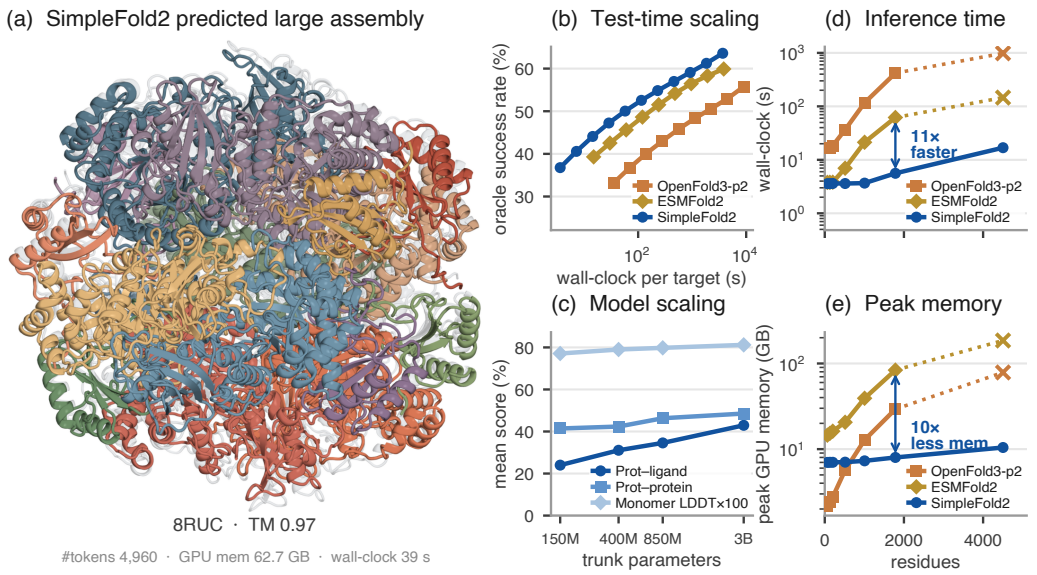


Figure 1: (a) SimpleFold2 enables scalable prediction of large complexes with $O(10^3)$ residues on a single H100, which is largely intractable using AlphaFold-style architectures . (b) SimpleFold2 high throughput leads to better test-time scaling performance for protein-ligand complexes. (c) Scaling SimpleFold2 yields consistent performance gains across different interaction categories. (d) Wall-clock and (e) peak memory for different co-folding models on a single NVIDIA H100 GPU, over different token counts (crosses on a dotted segment denote OOM and estimated usage).

In practical settings, these co-folding models are often deployed in larger pipelines for protein design, virtual screening, target identification, lead optimization, etc. These pipelines require to run many rounds of inference (sometimes in the order of 10^5 times) in order to select a few reasonably accurate candidate structures for downstream experimental validation. As a result, the value of co-folding models is a function of both their accuracy and their throughput. This is even more prominent in agentic workloads where agents can run co-folding models autonomously via tool calling (Anthropic, 2026; Merchant et al., 2026). In typical settings this is evaluated via inference-time scaling, where models are repeatedly run under a certain computational budget, expressed either as a number of samples to be generated by a model or the total amount of wall-clock time. The ability to maximize the throughput of a co-folding model is therefore very important for real deployment scenarios (see Figure 1b for test-time scaling results for protein-ligand interactions under different co-folding models).

In this work, we introduce SimpleFold2. Like SimpleFold (Wang et al., 2026), SimpleFold2 is based on the core hypothesis that co-folding is fundamentally a conditional generative modeling problem, not dissimilar to problems like text-to-image generation (Rombach et al., 2022) where scaling up generative models has yielded success (Peebles & Xie, 2023; Esser et al., 2024). While SimpleFold tackled the protein monomer case, SimpleFold2 extends to general biomolecular co-folding, which is a more complicated and challenging task. Unlike AlphaFold-style architectures, in SimpleFold2 every latent is a single-sequence representation, not a pairwise tensor. We do not build any bespoke architectural components (Abramson et al., 2024a) or optimized kernels (NVIDIA, 2025) and only leverage operators that standard transformer blocks already use (Dao et al., 2022). As a result, we can scale up model sizes without large memory or compute overhead. SimpleFold2 uses $\sim 0.3\times$ the memory of AlphaFold-style models and runs at least $\sim 10\times$ faster for complexes around 1700 residues, with favorable scaling behavior as the complex size increases. In particular, SimpleFold2-3B folds a 4960-residue complexe on a single H100 in 39s using 62GB of memory (Figure 1a), while AlphaFold-style models hit memory limits (see Figure 15 for examples of SimpleFold2 predictions on large protein assemblies). Our contributions are as follows:

- We introduce SimpleFold2, a co-folding model built using general purpose transformers. SimpleFold2 only uses single-token representations, a training objective with only two terms, and no pair representation, triangle updates, template stacks or recycling loops.

- We scale SimpleFold2 to 3B parameters. We benchmark it in three settings – FoldBench, OpenBind, and Runs N' Poses – and find that scale unlocks strong protein-ligand performance and comparable monomer performance at a fraction of the memory and runtime of AlphaFold-style models.
- We characterize the inference-time scaling of SimpleFold2. We find that the efficiency from using a simple architecture allows SimpleFold2 to set new Pareto boundaries in GPU time compared to success rate relative to other co-folding methods.

2 Background and related work

Diffusion and flow matching: Diffusion and flow matching models (Albergo et al., 2023; Lipman et al., 2022; Song & Ermon, 2019) use a neural network to parameterize a time-dependent flow that matches a tractable prior distribution to an empirical data distribution. To generate new samples, one samples from a prior (conventionally a standard Gaussian) and integrates the time-dependent flow by repeatedly running the network. Explicitly, we define the corruption process to be an interpolant between noise ϵ and data $\mathbf{x}_1$ and train a neural network to predict the instantaneous velocity $\mathbf{v}_\theta(\mathbf{x}_t, t) = \mathbb{E}[\mathbf{x} - \epsilon | \mathbf{x}_t, t]$. A common choice is the linear interpolant $\mathbf{x}_t = t\mathbf{x}_1 + (1-t)\epsilon$. Samples can be generated by integrating through the flow trajectory.

Biomolecular co-folding: Co-folding models like Boltz (Wohlwend et al., 2024), Chai (Boitreaud et al., 2024), Protenix (Team et al., 2025) and OpenFold3 (The OpenFold3 Team, 2025), closely follow the AlphaFold3 (Abramson et al., 2024b) network architecture (we refer to these as AlphaFold-style architectures). In these architectures, an explicit pair representation $\mathbf{z} \in \mathbb{R}^{L \times L \times c}$ is initialized using features from deep MSA stacks, and then updated via a large number of Pairformer blocks. Each Pairformer block consists of a sequence of triangular update layers (each of which scales as $O(L^3)$, where L is the sequence length). After the Pairformer stack, the resulting pair representation is used to condition a lightweight diffusion transformer by biasing the attention logits. ESMFold2 (Candido et al., 2026) maintains a similar structure, though the pair representation is initialized using features from a pLM pretrained on a large set of amino-acid sequences (Candido et al., 2026), as opposed to processing a deep MSA stack (Abramson et al., 2024b). We refer readers to Tab. 1 for more details on key architectural differences between models and Appendix A for a deeper related work discussion.

Although Pairformers and triangular update layers are powerful, they are also very expensive in both memory and compute. This has motivated recent work on hardware-aware implementations for these operations. Ouyang-Zhang et al. (2026) replaces triangle attention by triangle multiplication, which scales similarly but is more efficient to implement. Works like cuEquivariance (NVIDIA, 2025) and Lin et al. (2026) build out optimized versions of triangle operations to reduce peak memory load across GPUs. From a methodological viewpoint, the question of whether Pairformers are necessary remains open. Practically, the use of triangular updates limits both sequence length (a particular problem for multi-chain complexes) and inference scaling.

3 SimpleFold2

3.1 SimpleFold2 design space

SimpleFold2 is built from standard transformer blocks with adaptive layer norm (Peebles & Xie, 2023), QK-normalization (Dehghani et al., 2023) and SwiGLU (Shazeer, 2020). A high level architecture diagram is shown in Figure 2. We highlight significant design choices in Tab. 1 and describe them in this section, with detailed ablations in Section 4.6.

Architecture: Our primary departure from monomer-only SimpleFold (Wang et al., 2026) is to split the architecture into an encoder-decoder model (Figure 2). Where the encoder processes protein sequences (and other binders when required) and the decoder generates atomic coordinates. Both encoder and decoder are standard transformers models. The encoder is independent of t and is amortized across timesteps, while the decoder is conditioned on t using adaLN layers (Peebles & Xie, 2023).

Table 1: Key design comparison. SimpleFold2 discards the pair representation and every block used to process it, replacing the roles it played with standard single-sequence attention blocks (§3.1). While ESMFold2 also uses a large frozen pLM, it employes it to *initialize* a pair representation rather than to replace it. N_τ = number of tokens.

	SimpleFold2	**AlphaFold-style**	**ESMFold2**
Trunk representation	**Sequence states**	Pair states	Pair states
Trunk sequence scaling	$O(L^2)$ **attention**	$O(L^3)$ triangles	$O(L^3)$ triangles
Evolutionary input	Shallow in-context MSA	Deep MSA (trunk)	Deep MSA
Trunk passes per evaluation	One	Recycling	Up to 20 recycles
Chain/symmetry identity	Axial RoPE	Pair biases	Pair biases
Chemical graph	Laplacian eigenvectors	Pair features	Pair features
Generative objective	Flow matching (velocity)	Diffusion (EDM)	Diffusion (EDM)
Training objective terms	FM, lDDT	DM, lDDT, dist, bond	DM, lDDT, dist, bond

Encoder: Similar to text-to-image models (Rombach et al., 2022; Esser et al., 2024), we combine information from multiple language models to attain an initial embedding for our trainable encoder. In particular, we concatenate features from ESMC-6B (Candido et al., 2026) and E1-600M (Jain et al., 2025) across the channel dimension before being consumed by our encoder. E1-600M (Jain et al., 2025) allows us to integrate MSA information in-context, providing an efficient and simple way to encode evolutionary information. Different from text-to-image, we incorporate the encoder output with in-channel concatenation instead of cross-attention. This is both more compute-efficient than cross-attention and provides better-aligned locality information through conditioning.

Decoder: The flow matching decoder consumes noisy atomic coordinates $\mathbf{x}_t$ by Fourier-embedding the coordinates (Tancik et al., 2020) (along with element, charge, and atom information). An atom encoder projects atomic features and groups them at the residue level tokens. These grouped tokens are concatenated with the encoder output and fed to a deep residue trunk. Finally an atom decoder ungroups and projects residue tokens to be decoded into atomic coordinates.

Featurization: AlphaFold3-style models encode chain and symmetry IDs using pairwise biases. We instead encode chain and symmetry IDs using axial rotary embeddings (Su et al., 2024), where we rotate channels using the features (`residue_index`, `entity_id`, `asym_id`, `sym_id`). This provides relative information (e.g., are these residues from the same chain?) without requiring absolute positional embeddings.

Ligands: AlphaFold3 encodes ligands using pairwise distances from a reference conformer. To generate invariant per-token representations, we instead eigendecompose the bond-graph Laplacian and use the k lowest eigenvectors to featurize ligand connectivity (Wang et al., 2023).

Objective: We train SimpleFold2 using a combination of two objectives: a flow-matching objective and a smooth lDDT term. The former regresses the target velocity over heavy atoms with two co-folding-specific modifications relative to SimpleFold (Wang et al., 2026).

$$\ell_{\mathrm{FM}} = \mathbb{E}_{\mathbf{x},\mathbf{c},\boldsymbol{\epsilon},t}\left[\frac{w(t)}{N_a}\left\|\mathbf{v}_\theta(\mathbf{x}_t,\mathbf{c},t)-\mathbf{v}_t^\star\right\|^2\right], \qquad w(t)=t^2+(1-t)^2, \tag{1}$$

First, the timestep weight $w(t)$ normalizes the expected velocity-MSE to $O(1)$ at every noise level; without it the noisy end of the path dominates the gradient. Second, the regression target $\mathbf{v}_t^\star$ is taken against a *weighted-Kabsch-aligned* ground truth: we form the one-step estimate $\hat{\mathbf{x}}(\mathbf{x}_t) = \mathbf{x}_t + (1-t)\mathbf{v}_\theta$, align $\mathbf{x}$ onto it, and regress toward the aligned target. This removes the global rigid-motion component of the error. In addition, we include a smooth LDDT term on the same one-step estimate to capture fine-grained local geometry.

$$\ell_{\mathrm{LDDT}} = \mathbb{E}\left[1-\frac{\sum_{i\neq j}\mathbb{1}(\delta_{ij}<\mathcal{C}_{ij})\,\sigma(|\delta_{ij}-\hat{\delta}_{ij}|)}{\sum_{i\neq j}\mathbb{1}(\delta_{ij}<\mathcal{C}_{ij})}\right], \tag{2}$$

where δ_{ij} and $\hat{\delta}_{ij}$ are ground-truth and predicted inter-atomic distances and $\sigma(\cdot)$ averages four sigmoids at thresholds $\{0.5, 1, 2, 4\}$ Å. The neighbor cutoff $\mathcal{C}_{ij}$ is modality-dependent—30 Å for

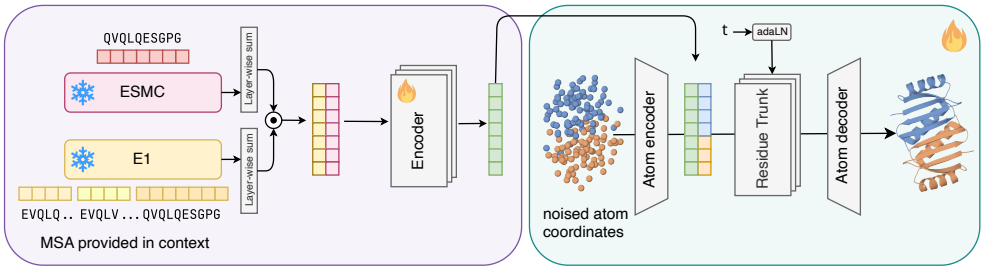


Figure 2: **SimpleFold2 architecture**. We first independently **encode** protein sequences via a learned weighted sum over hidden states from ESMC-6b and E1, where MSAs are provided in context to the latter. These features are jointly processed with other chains (e.g., ligands and nucleic acids) using a trainable encoder. To condition the **diffusion decoder**, we concatenate these features in-channel with a residue-level encoding of the noised atomic coordinates. The output of the residue trunk is passed through an atom encoder to return it to atom-space.

nucleic-acid pairs against 15 Å elsewhere—since nucleic acids are locally sparser than protein cores. The total objective is $\ell = \ell_{\text{FM}} + \ell_{\text{LDDT}}$.

Self-conditioning. At each step the model is shown its own estimate of the clean structure, i.e., $\hat{\mathbf{x}} = \mathbf{y}_t + (1-t)\,\mathbf{v}_\theta(\mathbf{y}_t, t)$ (Chen et al., 2022). We only keep to one coordinate per token by taking the token's representative atom, embed that coordinate with the same Fourier encoding the model uses for positions, and concatenate it to the token latent entering the flow trunk. During training the estimate comes from an extra forward pass run under `no_grad` and detached; self-condition is applied on 90% of training steps and replaced by the unconditioned null input on the remainder 10%. At inference self-condition is applied for free: the denoised output of one integration step is the estimate fed to the next, and only the first step uses the null input.

Timestep sampling. Training timesteps are drawn from a mixed logit-normal and uniform distribution, $t \leftarrow 0.98\,\texttt{LN}(m, s^2) + 0.02\,\mathcal{U}(0,1)$, with $m = 1.2$, $s = 1.5$. This shifts the timesteps in training toward clean data (close to $t = 1$), concentrating capacity on the low-noise end of the trajectory where fine atomic detail is resolved.

Inference. At inference we take the variance-exploding EDM schedule in AlphaFold3 and drive it with our velocity field. See Appendix B for details on our sampler ablations and exact configuration.

Confidence Module: The confidence module architecture is similar to Abramson et al. (2024b), where it reads both latents from encoding-trunk and flow-trunk at $t = 1$ as well as a binned distance embedding of the predicted coordinates; since SimpleFold2 has no pair track, the confidence module builds its own as a rank-64 outer sum. Four blocks alternate pair-biased self-attention with a triangle multiplicative update on that pair representation, and emit pAE, pDE, per-atom pLDDT, a resolved mask, and an auxiliary distogram supplying the contact probabilities the global pDE needs.

4 Experiments

4.1 Setup

Models. Our largest model SimpleFold2-3B uses hidden size 2048, a 20-block encoding trunk, a 28-block flow trunk, and 2- and 6-block atom encoder and decoder respectively. Detailed model configurations can be found in Appendix E.

Training and inference: SimpleFold2 is trained with a hybrid optimizer: We use Muon (Jordan et al., 2024) on two-dimensional weights and AdamW (Kingma & Ba, 2015) at half the learning rate on every remaining parameter. In the first training phase of SimpleFold2-3B, the model is trained for 150k iterations with crop size $C = 384$ and each crop replicated into $N_c = 36$ copies at different noise levels. We use $N_p = 64$ B200 GPUs and $n_{\text{msa}} = 8$. We use a learning rate of 0.0004 for Muon with a 5k linear warmup. In the second and third training phases, we apply a cosine LR decay to 1% of the original learning rate peak. In particular, the second phase trains the model for 20k iterations with $C = 640$ and $N_c = 22$ while the third phase continues the training for another 40k iterations

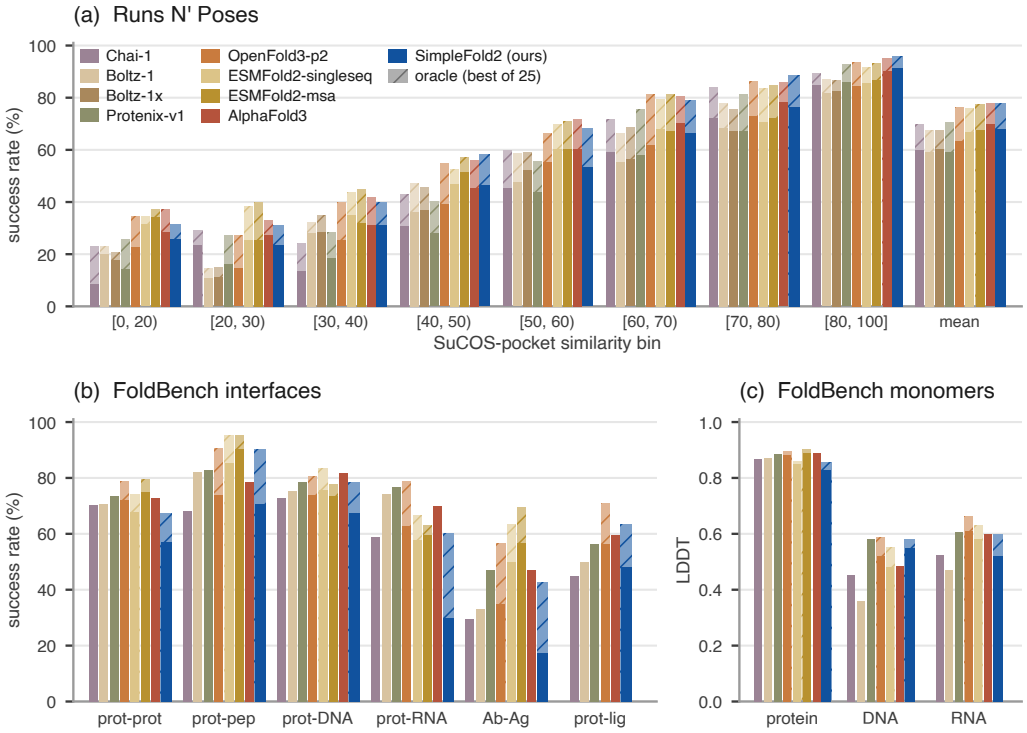


Figure 3: Results of SimpleFold2 compared with co-folding baseline models. **(a)** Runs N' Poses success rate (lDDT-PLI > 0.8 and RMSD < 2 Å) as a function of SuCOS-pocket similarity to the training set, with the mean pooled over all 1,735 ligand instances at right. **(b)** FoldBench interface success rate (DockQ ≥ 0.23; protein–ligand instead requires RMSD < 2 Å and lDDT-PLI > 0.8). **(c)** FoldBench monomer mean lDDT.

with $C = 768$ and $N_c = 14$. We also apply EMA=0.999 during training. Our confidence module training is conducted at a separate training phase using a frozen (i.e. pretrained) SimpleFold2-3B model on structures from rollouts of 25 steps. During inference, unless stated otherwise, we follow the 5×5 protocol where 5 seeds are drawn and 5 samples are generated per seed. Each model is evaluated at its training MSA depth (i.e., $n_{\text{msa}} = 8$). For each metric, we report the ranked performance over 25 samples and the *oracle* best-of-25 performance.

Data. We train on a mixture of experimental structures from the PDB (Berman et al., 2000) and distilled data used in OpenFold3 (The OpenFold3 Team, 2025), sampled per batch with fixed probabilities. In first training stage, we apply the ratio of 74%/1%/12.5%/12.5% for distilled long-monomers/distilled short-monomers/PDB structures/PDB interfaces and crop all structures to 384 tokens and apply both spatial and sequential cropping. In 2nd and 3rd training phases, the ratio changes to 49%/1%/25%/25% where PDB data is more frequently sampled and crop size increases to 640 and 768 respectively.

Baselines: We benchmark our performance against AlphaFold3 (Abramson et al., 2024a) and OpenFold3-p2 (The OpenFold3 Team, 2025), an open source AlphaFold reproduction. We also include ESMFold2 (Candido et al., 2026) in both single sequence and MSA modes. Finally we also compare with a full suite of AlphaFold-style models, including Boltz (Wohlwend et al., 2024), Protenix (Team et al., 2025), and Chai (Boitreaud et al., 2024).

4.2 Runs N' Poses

Runs N' Poses (RnP) (Škrinjar et al., 2025) is a protein-ligand co-folding benchmark in which data is stratified by the SuCOS-pocket similarity of the binding site to the training set. This provides a way to measure how models generalize to pockets beyond the training data. In RnP, a prediction is successful when lDDT-PLI > 0.8 and RMSD < 2 Å. We evaluate on the

shared set of 1,738 ligand instances across 1,543 systems, drawing 5 flow-matching samples from 5 seeds per target and selecting based on the model's own confidence score (Figure 3a).

On ranked performance across the whole dataset, SimpleFold2 attains a 68.1% success rate, slightly improving on ESMFold2 by 0.4%. SimpleFold2 clearly outperforms most existing co-folding models like OpenFold3-p2, Boltz-1x, Chai-1, Boltz-1 and Protenix-v1 by a marging ranging 4.8% to 9.0%. SimpleFold2 only underperforms AlphaFold3 by short gap of 1.6%. On oracle performance SimpleFold2 and AlphaFold3 both score 77.7% over 25 samples (5 samples by 5 seeds) per target for both, against 77.4 for ESMFold2, 76.2% for OpenFold3-p2 and 67-70% for the rest of models. SimpleFold2 thus generates protein-ligand poses of the same quality as AlphaFold3 and of higher quality than all other co-folding models. Figure 4 shows a docked sample.

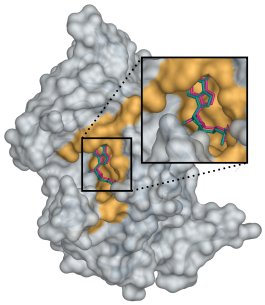


Figure 4: Structure `8fx3` from RnP. Our **SimpleFold2 prediction** closely matches the **ground truth pose.**

A natural expectation is that SimpleFold2 would perform poorly on low similarity bins since it does not encode the inductive biases in AlphaFold3-style architectures. In Figure 3a we find that for high similarity bins SimpleFold2, attains the highest success rate of any model in the $[80, 100]$ bin, 91.4% against 90.3% for AlphaFold3, and 76.5% in $[70, 80]$. More surprisingly, we also find that on the 372 instances in the four lowest-similarity bins (the regime in which pocket novelty is greatest and most models degrade sharply) SimpleFold2 attains 36.8% SR. This result closely matches AlphaFold3's 37.1% and is well ahead of all other models including OpenFold3-p2, Boltz and Protenix-v1 by 6.7 to 14.5 points. To our knowledge, this is the first empirical result of its kind, demonstrating the generalization capabilities of general purpose transformer architectures for co-folding.

4.3 FoldBench

FoldBench (Xu et al., 2025) evaluates monomer folding and assembly co-folding on a low-homology set spanning nine categories: protein, DNA, and RNA monomers (scored by mean lDDT) and protein-protein, protein-peptide, protein-DNA, protein-RNA, antibody-antigen, and protein-ligand interfaces, scored by success rate (i.e., DockQ ≥ 0.23 or RMSD < 2 Å and lDDT-PLI > 0.8 for protein-ligand). On monomers SimpleFold2 is competitive across all three modalities (Fig. 3(c)). On protein monomers, SimpleFold2 attains an 0.83 LDDT, comparable to other baseline models. SimpleFold2 is the second best DNA model at a mean lDDT of 0.548, ahead of models including AlphaFold3 and OpenFold3-p2. On RNA it obtains an LDDT of 0.521, at level with Chai-1 and ahead of Boltz-1.

On interfaces, SimpleFold2 shows comparable performance on several co-folding categories including protein-ligand, protein-DNA, and protein-peptide (Fig. 3b). In particular, SimpleFold2 achieves 48.0% success rate on protein-ligand, ahead of Chai-1 (44.9%); on protein-peptide 70.6%, within 3.2 points of OpenFold3-p2 (73.8%); and on protein–DNA 67.5%, close to ESMFold2-MSA and OpenFold3-p2 (both 73.7%). When analyzing oracle performance over the pool of 25 generated structures SimpleFold2 reaches 63.3% on protein-ligand; 90.2% on protein-peptide, matching the oracle performance of OpenFold-p2. Two categories worth further analysis are antibody-antigen and protein-RNA, for which SimpleFold2 lags behind existing models. We attribute this to the fact that PDB data contains a very limited number of structures for protein-RNA and antibody-antigen interactions. This contrasts with protein-ligand interactions, which are more abundant in PDB data and for which SimpleFold2 outperforms most methods (see Sect. 4.2). We strongly believe that collecting synthetic training sets of antibody-antigen structures, (similar to TorchFold (Team et al., 2026)) will close this gap. We show visualizations of predicted structures and complexes in Figures 13 and 14.

4.4 OpenBind

OpenBind (Nelen et al., 2026) scores 802 protein-ligand systems under a criterion stricter than pure geometric accuracy. In OpenBind a prediction counts as *success-valid* only if it places the ligand within 2 Å RMSD *and* attains lDDT-PLI ≥ 0.8 *and* clears PoseBusters (Buttenschoen et al., 2024) validity, which takes into account both chemical and physical implausibility. Fig. 5(a) shows the

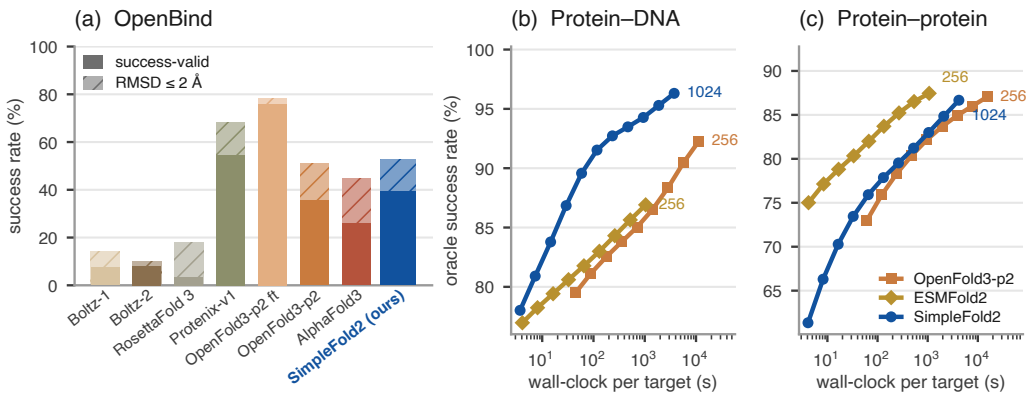


Figure 5: (a) Results of SimpleFold2 on OpenBind. (b)(c) Inference-time scaling: success rate as a function of runtime. Each curve shows oracle success rate as the number of independent seeds is increased, plotted against the resulting single-card GPU-seconds per target (log scale). FoldBench (b) protein–DNA and (c) protein–protein.

performance of SimpleFold2 compared to other co-folding models, where solid band reports the *success-valid* rate, and the shaded band is the relaxation to RMSD ≤ 2 Å only. SimpleFold2 obtains 39.3% success-valid rate, ahead of OpenFold3-preview-2 (35.5%) and AlphaFold3 (25.8%), with Boltz-2, Boltz-1 and RosettaFold 3 an order of magnitude behind. SimpleFold2 is only outperformed by Protenix-v1 and a modality specific fine-tuned version of OpenFold3-p2.

4.5 Inference Time Scaling

The throughput of a co-folding model, measured as the number of structures it can generate per unit of time, is a very important metric, particularly as these models are embedded in real world pipelines like virtual screens over compound libraries. To empirically analyze this we measure inference-time scaling as oracle success rate against single-card GPU-seconds per target (on an NVIDIA H100 GPU). During inference, we follow the common protocol where five samples are generated per seed, i.e., one encoding-trunk pass together with the five diffusion/flow-matching samples that share its MSA draw and trunk latent. We follow the configurations provided by official ESMFold2 and OpenFold2-p2 repositories which are listed in Appendix F.

We benchmark the inference-time scaling on protein-protein and protein-DNA targets in FoldBench (Fig. 5b and c), as well as the hardest targets in Run N' Poses (those with SuCOS-pocket similarity smaller than 40, Fig. 1b. More results in Appendix G). On Run N' Poses, running one SimpleFold2 seed takes on average 3.6 GPU-seconds versus 35.2 for OpenFold3-p2 and 15.0 for ESMFold2 (note that we adapt ESMFold2 with its fused Triton kernel to accelerate its inference). On FoldBench protein-ligand targets SimpleFold2 takes 3.5 GPU-second on average versus 41.3 for OpenFold3-p2 and 14.8 for ESMFold2.

SimpleFold2 dominates the Pareto frontier of GPU time vs success rate on Runs N' Poses, as shown in Figure 1b. SimpleFold2 is 15.0 points better at a budget of 35 seconds per target and 12.5 points better at 281 seconds. The 55.8% SR that OpenFold3-p2 reaches after $9{,}004$ seconds is obtained by SimpleFold2 in 300 seconds, a $30\times$ reduction. Protein-DNA behaves similarly, achieving the same quality as OpenFold3-p2 in roughly $11\times$ less time. On protein-protein interactions SimpleFold2 stays ahead at matched wall-clock budget despite having a lower single-seed accuracy. The general purpose architecture of SimpleFold2 enables a very high throughput without hitting the memory or compute limits of AlphaFold-style models.

4.6 Ablations

We investigated a sequence of design choices for SimpleFold2. We discuss three choices below: the choice of sequence encoder, the use of synthetic data, and the method of conditioning. We enumerate findings in Table 2 and present further ablations and detailed results in Appendix B. The synthetic data ablation is run at 3B size; all others use 400M.

Sequence encoder	Monomer lDDT
E1-600m (MSA)	0.760
ESM-C 6B	0.760
Fused	**0.780**

(a) **Sequence encoder:** Fusion improves monomer lDDT.

Synthetic data	Monomer lDDT	Prot/prot SR (%)
0%	0.772	47.4
50%	0.814	**50.8**
75%	**0.816**	50.6

(b) **Synthetic data:** Improves monomer and prot–prot metrics.

Conditioning	Monomer lDDT	Prot/prot SR (%)	Ligand SR (%)
Pair bias	0.781	40.2	25.0
Concat	**0.787**	**43.9**	**29.9**

(c) **Conditioning:** Concatenation outperforms pair biasing across metrics.

Table 2: **Ablation study:** Typical setting is best-of-one eval on FoldBench. Default configurations are highlighted in gray. More detailed results in Appendix B.

Choice of pLM embedder: A standard choice in the text-to-image setting is to combine information from multiple language models (commonly CLIP (Radford et al., 2021) and a T5 embedder (Guo et al., 2022)) to generate a text encoding for an image diffusion model. Combining different models can capture complementary information, depending on their data and training objective. We consider two models: ESMC (Candido et al., 2026), the language model that forms the base for ESMFold2, and E1 (Jain et al., 2025), a language model that explicitly conditions on homologous sequences using in-context learning. Both models are trained similarly using self-supervised objectives on large sequence corpuses. Table 2a shows that fusing the models together (via in-channel concatenation, followed by a linear projection) outperforms either model individually.

Data corpus: Since the Protein Data Bank (Berman et al., 2000) contains a relatively small amount of experimental data ($\approx 2 \times 10^6$), all co-folding models use extended training sets of synthetic data from model predictions in order to boost their performance. In Table 2b, we find that this also generally helps SimpleFold2. More interestingly, we find that a SimpleFold2-3B model trained with PDB data only (0% synthetic data) obtains very good performance. One might think that either SimpleFold (Wang et al., 2026) or SimpleFold2, work well purely because they are distilling the data from AlphaFold-style models and their inductive biases. We empirically show that this is not the case and refer readers to Appendix B for a more in depth evaluation of the PDB-only model.

Conditioning pathway: The pair representation in AlphaFold-style architectures is updated using a sequence of expensive triangular operations, and it is projected to bias the attention logits in the diffusion decoder. We can decouple these operations; we explore conditioning the diffusion model by biasing the attention logits, but with a pair representation updated by a standard transformer (as opposed to triangular multiplication/attention). The details of this architecture are in Appendix B. Table 2c shows that in-channel conditioning, in addition to being more memory efficient, generally outperforms pair-biasing without triangular operations.

Scaling model size. We train four sizes: 155M, 411M, 856M and 3.3B #param, under one recipe and compare them at 70k training steps. Accuracy rises monotonically as model size increases (Figure 1c). Critically, we can scale up model sizes SimpleFold2 architectures without hitting the compute and memory limitations of AlphaFold-style architectures while also improving throughput.

5 Conclusion

In this work, we introduced SimpleFold2, a biomolecular co-folding model built using standard transformer blocks and single-token representations. SimpleFold2 departs significantly from standard AlphaFold architectures and provides much more efficient performance. Because of its efficiency in both runtime and memory, it provides a compelling approach for high throughput pipelines, folding large complexes, and other critical challenges in real world biological discovery. Although SimpleFold2 underperforms the AlphaFold-style models in categories where data is particularly sparse like antibody-antigen interactions, we believe that using synthetic training data will close this gap. Surprisingly, SimpleFold2 outperforms most AlphaFold-style models on co-folding protein-ligands, and is competitive across monomers. We hope our work encourages further research into how the paradigms of simplicity and scalability can be best used in frontier bioengineering.

AI USE STATEMENT

In this work, we used generative AI tools to design or provide feedback on experiments, implement methods, and interpret results. We used AI tools extensively for coding and to suggest possible hypotheses for data. We did not use generative AI to generate synthetic datasets or to develop theory, formulate mathematical claims, assist in proof writing, assist in translation, reformat datasets, or support data anlyses. We have reviewed all AI-assisted work; we tested and reviewed LLM generated code and experimental results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work engages with machine learning for bioengineering, which can broadly be used for both good and ill. We do not see any specific ethical issues that need to be highlighted, but strongly support the framework outlined in Responsible AI x Biodesign.

REPRODUCIBILITY STATEMENT

To support reproducibility, we have provided our model configurations in Appendix E and all training details required to reproduce our experiments in Section 3. We similarly describe our datasets in detail in Section 3, and note that all of our datasets (primarily the PDB (Berman et al., 2000) and OpenFold (The OpenFold3 Team, 2025)) are publicly available.

REFERENCES

Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J. Ballard, Joshua Bambrick, Sebastian W. Bodenstein, David A. Evans, Chia-Chun Hung, Michael O'Neill, David Reiman, Kathryn Tunyasuvunakool, Zachary Wu, Akvilė Zemgulyte, Eirini Arvaniti, Charles Beattie, Ottavia Bertolli, Alex Bridgland, Alexey Cherepanov, Miles Congreve, Alexander I. Cowen-Rivers, Andrew Cowie, Michael Figurnov, Fabian B. Fuchs, Hannah Gladman, Rishub Jain, Yousuf A. Khan, Caroline M. R. Low, Kuba Perlin, Anna Potapenko, Pascal Savy, Sukhdeep Singh, Adrian Stecula, Ashok Thillaisundaram, Catherine Tong, Sergei Yakneen, Ellen D. Zhong, Michal Zielinski, Augustin Zidek, Victor Bapst, Pushmeet Kohli, Max Jaderberg, Demis Hassabis, and John M. Jumper. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630:493–500, 2024a.

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, 2024b.

Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

Anthropic. How Claude is uplifting biomolecular modeling, September 2026. URL `https://www.anthropic.com/research/claude-uplifts-biomolecular-modeling`. Published September 17, 2026.

Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N. Kinch, R. Dustin Schaeffer, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021.

Helen M. Berman, John D. Westbrook, Zukang Feng, Gary L. Gilliland, Talapady N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, 2000.

Jacques Boitreaud, Jack Dent, Matthew McPartlon, Joshua Meier, Vinicius Reis, Alex Rogozhnikov, and Kevin Wu. Chai-1: Decoding the molecular interactions of life. *BioRxiv*, 2024.

Martin Buttenschoen, Garrett M Morris, and Charlotte M Deane. Posebusters: Ai-based docking methods fail to generate physically valid poses or generalise to novel sequences. *Chemical science*, 15(9):3130–3139, 2024.

Salvatore Candido, Thomas Hayes, Alexander Derry, Roshan Rao, Zeming Lin, Robert Verkuil, Brian Z. Wu, Jae Shin Lee, Elisa S. Bruguera, and Alexander Rives. Language modeling materializes a world model of protein biology. *bioRxiv*, 2026. doi: 10.64898/2026.06.03.729735. ESMC and ESMFold2.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. *arXiv preprint arXiv:2208.04202*, 2022.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *International conference on machine learning*, pp. 7480–7512. PMLR, 2023.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. *arXiv preprint arXiv:2403.03206*, 2024.

Richard Evans, Michael O'neill, Alexander Pritzel, Natasha Antropova, Andrew Senior, Tim Green, Augustin Žídek, Russ Bates, Sam Blackwell, Jason Yim, et al. Protein complex prediction with alphafold-multimer. *biorxiv*, pp. 2021–10, 2021.

American Society for Cell Biology. *Molecular biology of the cell*, volume 13. American Society for Cell Biology, 2002.

Tomas Geffner, Kieran Didi, Zuobai Zhang, Danny Reidenbach, Zhonglin Cao, Jason Yim, Mario Geiger, Christian Dallago, Emine Kucukbenli, Arash Vahdat, et al. Proteina: Scaling flow-based protein structure generative models. *arXiv preprint arXiv:2503.00710*, 2025.

M. Guo, J. Ainslie, D. Uthus, S. Ontanon, J. Ni, Y. Sung, and Y. Yang. LongT5: Efficient text-to-text transformer for long sequences. In *NAACL*, 2022.

Sarthak Jain, Joel Beazer, Jeffrey A Ruffolo, Aadyot Bhatnagar, and Ali Madani. E1: Retrieval-augmented protein encoder models. *bioRxiv*, pp. 2025–11, 2025.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL `https://kellerjordan.github.io/posts/muon/`.

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.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Rohith Krishna, Jue Wang, Woody Ahern, Pascal Sturmfels, Preetham Venkatesh, Indrek Kalvet, Gyu Rie Lee, Felix S Morey-Burrows, Ivan Anishchenko, Ian R Humphreys, et al. Generalized biomolecular modeling and design with rosettafold all-atom. *Science*, 384(6693):eadl2528, 2024.

Dejun Lin, Simon Chu, Vishanth Iyer, Youhan Lee, John St John, Kevin Boyd, Brian Roland, Xiaowei Ren, Guoqing Zhou, Zhonglin Cao, et al. Fold-cp: A context parallelism framework for biomolecular modeling. *arXiv preprint arXiv:2603.14806*, 2026.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps. In *Advances in Neural Information Processing Systems*, 2022.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *Machine Intelligence Research*, 22 (4):730–751, 2025.

Jiankun Lyu, Sheng Wang, Trent E Balius, Isha Singh, Anat Levit, Yurii S Moroz, Matthew J O'Meara, Tao Che, Enkhjargal Algaa, Kateryna Tolmachova, et al. Ultra-large library docking for discovering new chemotypes. *Nature*, 566(7743):224–229, 2019.

Aditi T Merchant, Daniel Guo, Ben Viggiano, Lucas Brennan-Almaraz, Evelyn Hur, Tina Mai, Peter Yin, Samuel H King, Euan A Ashley, and Brian L Hie. A high-level programming language for generative biology with proto. *bioRxiv*, pp. 2026–06, 2026.

Jochem Nelen, Omeir Khan, Etowah Adams, Jasmin C Aschenbrenner, Warren Thompson, Ali Ebrahim, Eda Çapkin, Cédric Vallée, OpenBind, Elisabeth J Shotton, et al. The first openbind release: An open experimental structure–affinity dataset and benchmark for structure-based ai. *bioRxiv*, pp. 2026–08, 2026.

NVIDIA. NVIDIA cuEquivariance Documentation, 2025. URL `https://docs.nvidia.com/cuda/cuequivariance/`. Accessed: 2026-09-23.

Jeffrey Ouyang-Zhang, Pranav Murugan, Daniel Diaz, Gianluca Scarpellini, Richard Bowen, Nate Gruver, Adam Klivans, Philipp Krähenbühl, Aleksandra Faust, and Maruan Al-Shedivat. Triangle multiplication is all you need for biomolecular structure representations. In *International Conference on Learning Representations*, volume 2026, pp. 113263–113287, 2026.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning, ICML*, 2021.

R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Peter Škrinjar, Jérôme Eberhardt, Janani Durairaj, and Torsten Schwede. Have protein-ligand cofolding methods moved beyond memorisation? *BioRxiv*, pp. 2025–02, 2025.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*, 2019.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33:7537–7547, 2020.

ByteDance AML AI4Science Team, Xinshi Chen, Yuxuan Zhang, Chan Lu, Wenzhi Ma, Jiaqi Guan, Chengyue Gong, Jincai Yang, Hanyu Zhang, Ke Zhang, et al. Protenix-advancing structure prediction through a comprehensive alphafold3 reproduction. *BioRxiv*, pp. 2025–01, 2025.

TorchCraft Team, Yu Liu, Zhouhanyu Shen, Zhengyi Li, Xikun Huang, Jiaqi Liu, Shuxian Gao, Qilin Yu, Xiayan Qin, Yucheng Zhang, et al. Torchcraft: Unified binder design by inverting an all-atom structure predictor. *arXiv preprint arXiv:2609.19770*, 2026.

The OpenFold3 Team. Openfold3-preview, 2025. URL https://github.com/aqlaboratory/openfold-3.

Kathryn Tunyasuvunakool, Jonas Adler, Zachary Wu, Tim Green, Michal Zielinski, Augustin Žídek, Alex Bridgland, Andrew Cowie, Clemens Meyer, Agata Laydon, et al. Highly accurate protein structure prediction for the human proteome. *Nature*, 596(7873):590–596, 2021.

Yuyang Wang, Ahmed A Elhag, Navdeep Jaitly, Joshua M Susskind, and Miguel Angel Bautista. Swallowing the bitter pill: Simplified scalable conformer generation. *arXiv preprint arXiv:2311.17932*, 2023.

Yuyang Wang, Jiarui Lu, Navdeep Jaitly, Joshua Susskind, and Miguel Angel Bautista. Simplefold: Folding proteins is simpler than you think. In *International Conference on Learning Representations*, volume 2026, pp. 32027–32054, 2026.

Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

Jeremy Wohlwend, Gabriele Corso, Saro Passaro, Mateo Reveiz, Ken Leidal, Wojtek Swiderski, Tally Portnoi, Itamar Chinn, Jacob Silterra, Tommi Jaakkola, et al. Boltz-1: Democratizing biomolecular interaction modeling. *bioRxiv*, pp. 2024–11, 2024.

Sheng Xu, Qiantai Feng, Lifeng Qiao, Hao Wu, Tao Shen, Yu Cheng, Shuangjia Zheng, and Siqi Sun. Benchmarking all-atom biomolecular structure prediction with foldbench. *Nature Communications*, 17(1):442, 2025.

Yi Zhou, Chan Lu, Yiming Ma, Wei Qu, Fei Ye, Kexin Zhang, Lan Wang, Minrui Gui, and Quanquan Gu. Seedfold: Scaling biomolecular structure prediction. *arXiv preprint arXiv:2512.24354*, 2025.

A Related work

Biomolecular cofolding: The seminal work in protein folding was AlphaFold2 (Jumper et al., 2021), which achieved unprecedented accuracy in monomer structure prediction by parameterizing atomic structure as a sequence of rigid rotations. Works like ESMFold (Lin et al., 2023) built on this by using the same structure prediction module as AlphaFold2 but conditioning using a large language model. AlphaFold-Multimer (Evans et al., 2021) expanded the AlphaFold2 approach to fold complexes. RoseTTAFold All-Atom (Krishna et al., 2024) used a diffusion model to co-fold biomolecular assemblies.

In 2024, AlphaFold3 stripped away much of the equivariant/invariant machinery in favor of a lightweight diffusion model conditioned by a powerful Pairformer. This pattern was quickly replicated by a variety of works – Boltz, Chai, and Protenix (Wohlwend et al., 2024; Boitreaud et al., 2024; Team et al., 2025). ESMFold (Candido et al., 2026) initialized a pair representation from a language model and exclusively used triangular multiplication (instead of attention) to update the pair representation.

In monomer prediction, SimpleFold (Wang et al., 2026) explored conditioning a flow matching decoder on a language model. This showed strong performance on several monomer prediction tasks (including out-of-distribution tasks like orphan proteins).

B All ablations

In addition to the ablations presented in the main text, Table 3 shows three further ablations. We also present more detailed breakdowns of performance in Figures 7, 8,9, 10,11, and 12 all show more detailed results for each ablation segmented by category.

Encoder fraction: At fixed parameter count, we vary the share of parameters devoted to the encoder. Note that because the decoder is run many times, a larger relative encoder is generally cheaper at inference. Table 3(a) shows a defined peak at around 40% of total parameters.

Timestep sampling: The distribution used to sample t is important for performance (Esser et al., 2024). A complication unique to protein design is that both the high and low noise regimes are important, as the former determines global pose and the latter determines local geometry. Table 3(b) reports that while there is an optimal range, there is a tradeoff between these decisions.

Integrator: We can choose the integration scheme post hoc. Table 2(c) explores a mixed-SDE, a full-distribution ODE, a "low-temperature" ODE using an accelerated solver (Lu et al., 2025), and the integrator in AlphaFold3. We find the latter consistently outperforms ODE-based solvers across metrics. We describe the integrators we explored below:

$$d\mathbf{x} = \mathbf{v}_\theta(\mathbf{x}_t, t)\, dt \qquad \text{or} \qquad d\mathbf{x} = \left[\mathbf{v}_\theta(x_t, t) + \frac{g(t)^2}{2}\nabla_x \log p_t(\mathbf{x}_t)\right] dt + g(t)\, dW_t. \tag{3}$$

for arbitrary $g(t)$.

Flow ODE: Here, we integrate the standard ODE presented in Equation 3.

Mixed SDE: We integrate the SDE presented in Equation 3, but we anneal down γ. This is common in generative protein modeling (Geffner et al., 2025), and biases the model towards higher probability generations at the cost of no longer sampling the full distribution.

AlphaFold3: We use the variance exploding SDE where noise levels follow the Karras ladder $\sigma_i = \left[\sigma_{\max}^{1/\rho} + \frac{i}{N}(\sigma_{\min}^{1/\rho} - \sigma_{\max}^{1/\rho})\right]^\rho$ with $N = 200$, $\sigma_{\max} = 160$, $\sigma_{\min} = 4 \times 10^{-4}$, $\rho = 7$ and $\sigma_N \equiv 0$, all as ratios to $\sigma_{\text{data}} = 16$ Å. Starting from $\mathbf{x}_0 = \sigma_0 \boldsymbol{\epsilon}$, each step centers the current state, inflates the noise level to $\hat{\sigma}_i = (1 + \gamma_i)\sigma_i$ with $\gamma_i = \gamma_0\, \mathbf{1}[\sigma_{i+1} > \gamma_{\min}/\sigma_{\text{data}}]$, injects $\lambda\sqrt{\hat{\sigma}_i^2 - \sigma_i^2}\, \boldsymbol{\epsilon}_i$ to give $\tilde{\mathbf{x}}_i$, then takes one Euler step. With $t_i = 1/(1 + \hat{\sigma}_i)$ the denoiser is $D = t_i\tilde{\mathbf{x}}_i + (1 - t_i)\, \mathbf{v}_\theta(t_i\tilde{\mathbf{x}}_i, t_i)$, and the update collapses to

$$\mathbf{x}_{i+1} = \tilde{\mathbf{x}}_i + \eta\, \frac{\sigma_{i+1} - \hat{\sigma}_i}{1 + \hat{\sigma}_i}\left[\tilde{\mathbf{x}}_i - \mathbf{v}_\theta(t_i\tilde{\mathbf{x}}_i,\, t_i)\right], \qquad t_i = \frac{1}{1 + \hat{\sigma}_i}. \tag{4}$$

We use $\gamma_0 = 0.8$, $\gamma_{\min} = 1$ Å, $\lambda = 1.003$ and $\eta = 1.5$. The indicator switches the injection off once σ_{i+1} falls below $\gamma_{\min}/\sigma_{\text{data}}$, so the tail of every trajectory is deterministic; and the network is never evaluated on the variance-exploding state itself but on $t_i\tilde{\mathbf{x}}_i$, which stays of order one even at $\sigma = 160$.

Low-temperature ODE sampler: One advantage of ODE samplers compared to SDEs is we can use accelerated samplers like (Lu et al., 2022; 2025). These solvers exploit the semilinear structure of an ODE to reduce the number of function calls. However, a disadvantage is we cannot take advantage of the MixedSDE "low-temperature" sampling used in (Geffner et al., 2025), which anneals the effect of noise and biases the model using a score term. This is particularly important because SimpleFold2's conditioning is much less FLOP-heavy than Pairformermodels, so the posterior structure has higher variance.

We instead introduce a low-temperature ODE, shown in red in Figure 11. While this underperforms the AlphaFold3 sampler, it recovers essentially all of the MixedSDE performance, making it an interesting direction for situations where number of function calls is an important bottleneck (e.g., metagenomic screening). Our approach computes the velocity corresponding to a scaled score, appealing to the same intuition as gamma-sharpening used in CFG. Explicitly:

$$s_\theta(x, t) = \frac{t\, v_\theta(x, t) - x}{1 - t}, \qquad \tilde{v}_\theta(x, t) = v_\theta(x, t) + \frac{(w - 1)(1 - t)}{t} s_\theta(x, t) = w\, v_\theta(x, t) - \frac{w - 1}{t} x, \tag{5}$$

This is a heuristic, and we rely on empirical performance in our ablation. Figure 6 shows pseudocode for this integrator.

Details on pair-bias conditioning: We can decouple the pair bias from the triangular updates. We introduce an architectural primitive shown in Figure 6, where the pair representation is updated by a single sequence transformer. We then condition the diffusion decoder just as AlphaFold3 does, by biasing attention logits in the diffusion transformer. As shown in Figure 12, while we see some

Encoder frac.	Monomer lDDT	Ligand SR (%)
0.2	0.758	12.9
0.3	0.755	12.5
0.4	**0.765**	13.0
0.5	0.750	13.2
0.6	0.748	11.2
0.8	0.725	9.0

(a) **Encoder–decoder allocation:** Monomer lDDT peaks at 0.4, while ligand SR peaks at 0.5.

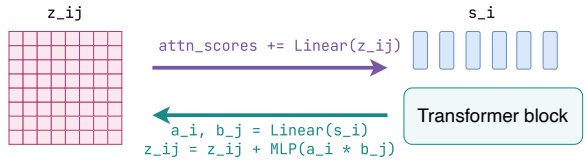
Distribution	med. t	Monomer lDDT	Ligand SR (%)
lognorm(0.0)	0.610	–	8.1
lognorm(0.6)	0.650	**0.740**	17.6
lognorm(1.2)	0.770	0.720	**25.1**
lognorm(1.8)	0.860	0.430	23.4
lognorm(2.4)	0.920	0.080	20.0

(b) **Timestep sampling:** Shifting coverage trades off interface and monomer quality.

Sampler	Monomer lDDT	Prot/prot SR (%)	Ligand SR (%)
Mixed SDE (γ=0.01)	0.772	44.8	**38.9**
ODE	0.613	41.9	30.4
Low-temp ODE	0.767	44.3	38.8
AF3 integrator	**0.816**	**50.7**	38.6

(c) **Sampler:** The AF3 integrator consistently leads on monomer and interaction metrics.

Table 3: **Ablation study:** Typical setting is best-of-one eval on FoldBench. Default configurations are highlighted in gray.



Low-temperature ODE algorithm

```
x = normal(shape)
for t, t_next in pairwise(times):
    v = v_theta(x, t, c)
    if w > 1 and t >= t_min:
        score = (t * v - x) / (1 - t)
        v += (w - 1) * (1 - t) / t * score
    x += (t_next - t) * v
return x
```

Figure 6: **Pair-bias architecture and low-temperature sampling.** *Left:* The pair representation receives low-rank updates from sequence transformer blocks and biases attention logits in subsequent blocks. *Right:* The low-temperature ODE sampler recovers the score from the predicted velocity and scales it by w, then uses this to reconstruct the velocity.

promising performance on categories like nucleic acids, on the whole it underperforms on proteins. For this reason, we focus on in-channel concatenation, though finding more efficient architectures that encode the pairwise inductive bias is an intriguing future direction.

C Sample Visualizations

D Super Large Protein Assemblies

E Model Configurations

Table 4 shows architectural settings for our model family. The atom encoder and decoder both use sliding window attention (window size 128) due to the large context size. We scale the atom decoder to atom encoder ratio as we scale the model, along with all hidden dimensions.

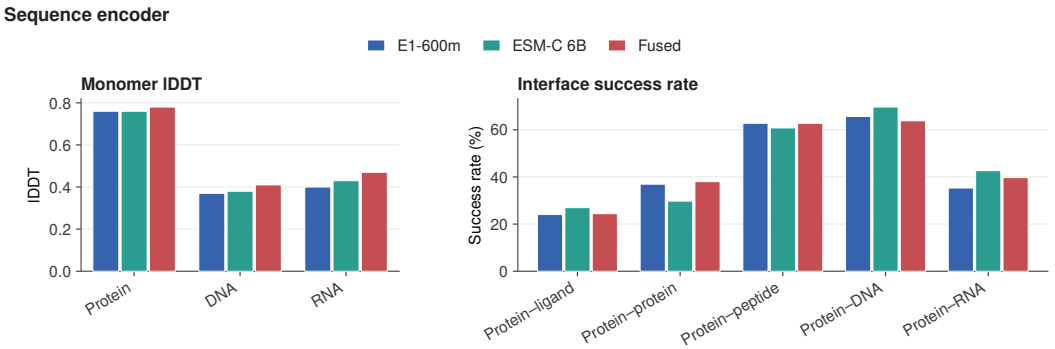


Figure 7: **Ablation on choice of sequence encoder.** On monomer quality, we consistently saw fusion improved over E1 or ESMC separately. Interface results saw more fluctuation. Because we concatenate in-channel, the FLOP difference between the fused and unfused models is negligible.

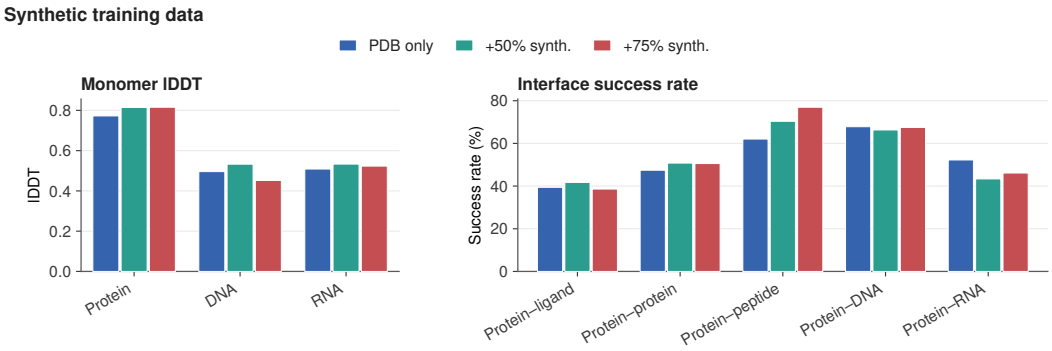


Figure 8: **Use of synthetic data:** Synthetic data consistently improved monomer quality. We saw particularly large gains in categories like protein–peptide. Note that additional synthetic data means less PDB data is seen for a fixed number of iterations, which explains drops in categories like protein–RNA.

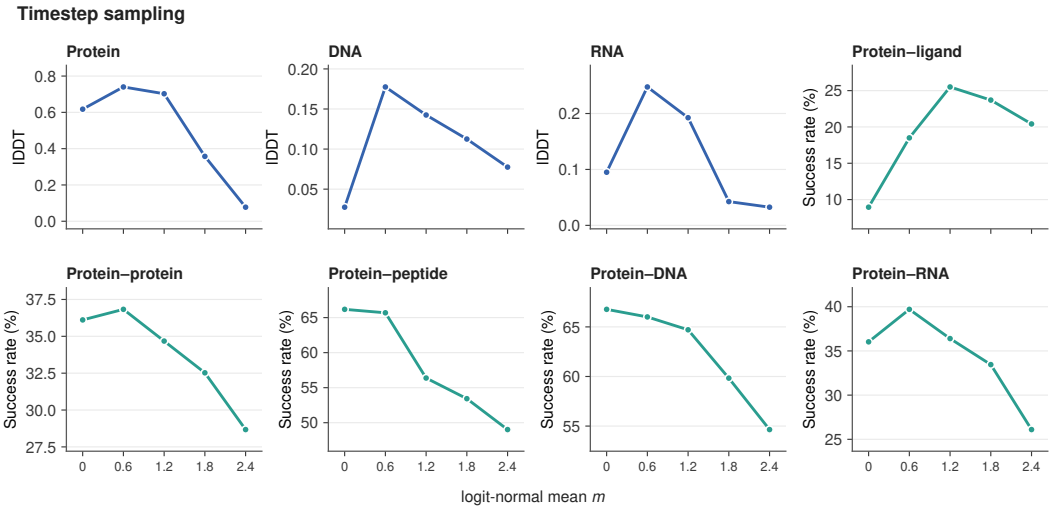


Figure 9: **Timestep sampling:** While logit-normal with logit-mean 0.6 slightly outperformed on monomer performance, it led to a drop in protein-ligand performance. Intuitively, global docking and local geometry are determined at different noise scales, so at fixed model capacity there is a tradeoff.

F Inference Configuration for Test-time Scaling

We follow the official implementation described in publication of OpenFold3-p2 and ESMFold2 for test-time scaling experiments. Table 5 collects the settings that affect accuracy or cost. All three models follow the AlphaFold3 per-seed protocol where one encoding-trunk pass per seed feeds 5 diffusion samples that share that seed's MSA subsample and trunk latent. The trunk dominates cost for OpenFold3-p2 and ESMFold2.

OpenFold3-p2. Run from the stock `predict` preset at full `32-true` precision with 10 recycles, the PAE head enabled, and the official MSA depth caps (50,000 UniProt, 10,000 UniRef90, 5,000 MGnify, and the effectively unbounded BFD/CFDB/BFD-UniRef limits). Diffusion emits 5 samples per trunk pass, matching the per-seed protocol above, and the model runs on its default accelerated attention path.

ESMFold2. We use 10 recycling loops and 68 diffusion sampling steps, the paper's stated default. MSA depth is capped at 1,024 and a 10% MSA column mask. Execution uses the fused Triton kernel backend, which accelerates the inference efficiency as documented in the offical GitHub repository.

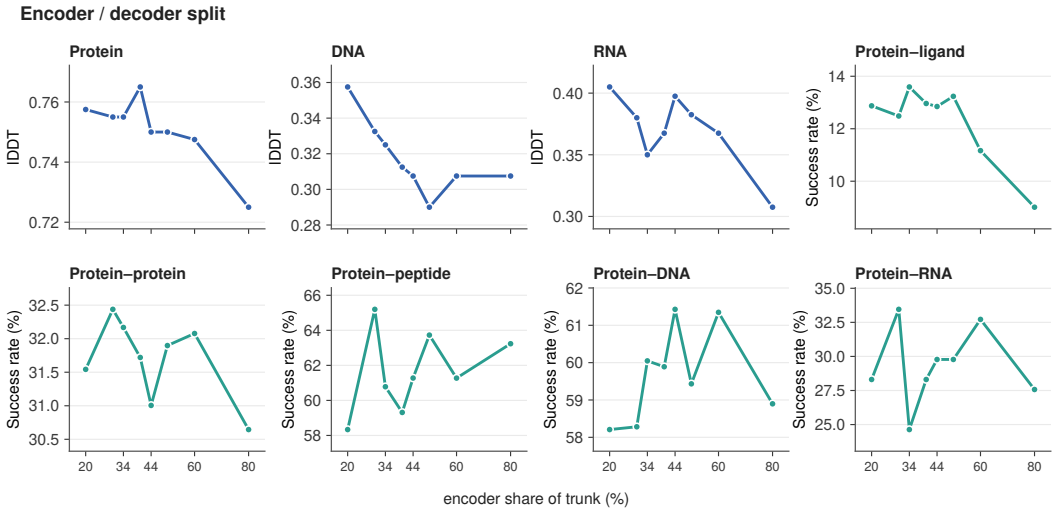


Figure 10: **Encoder capacity:** Due to adaLN layers, a DiT has slightly more parameters than an equal-layer transformer. For our 3B model, we slightly increase the decoder size.

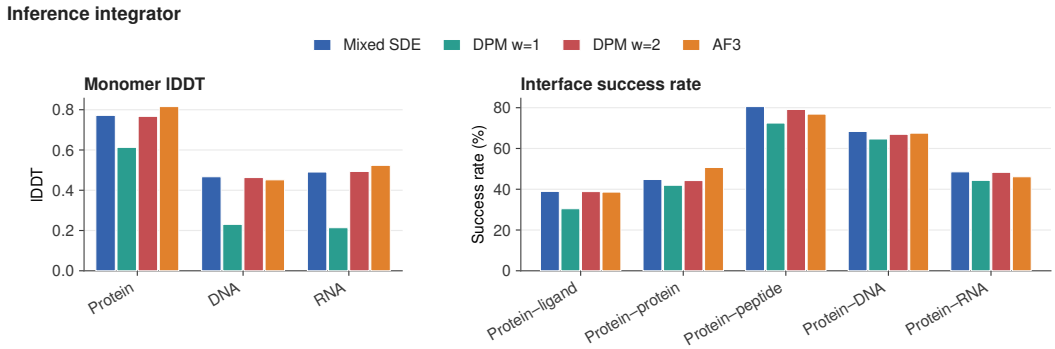


Figure 11: **Choice of integrator:** On most categories, AlphaFold3's EDM-style SDE either matches or outperforms flow-based methods.

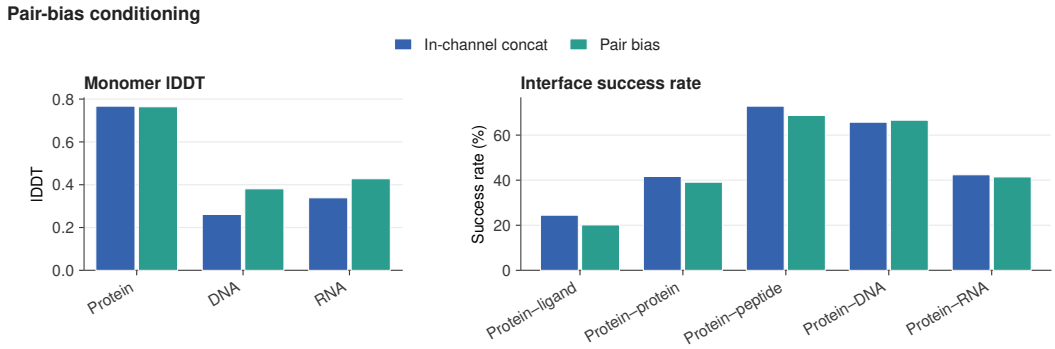


Figure 12: **Conditioning method:** We explored conditioning the diffusion model using an explicit pair representation without pair updates. Surprisingly, we found this helped significantly with monomer nucleic acids, but otherwise underperformed on most protein categories. We opted for the in-channel concatenation approach due to its simplicity.

	S	M	L	XL
Width				
Token dimension	768	1024	1280	2048
Token attention heads	12	16	20	32
Atom dimension	256	384	512	768
Atom attention heads	4	6	8	12
Head dimension	64	64	64	64
Token/atom width ratio	3.0	2.7	2.5	2.7
Depth and allocation				
Encoding trunk blocks	8	12	16	20
Flow trunk blocks	8	12	16	28
Total token blocks	16	24	32	48
Encoding share	50%	50%	50%	42%
Flow/encoding ratio	1.0	1.0	1.0	1.4
Atom encoder blocks	2	2	2	2
Atom decoder blocks	2	3	4	6
Total atom blocks	4	5	6	8
Atom dec./enc. ratio	1.0	1.5	2.0	3.0
Scale and training				
Parameters	155M	411M	856M	3.3B
Parameters vs. S	1.0×	2.7×	5.5×	21×
Velocity dimension	256	384	512	768

Table 4: **SF2 model-scale ladder.** We show relevant architectural differences between all model scales. All models are conditioned using the E1-600M (with MSA) and ESM-6b models. We do not include these in the parameter counts.

	SimpleFold2	OpenFold3-p2	ESMFold2
Trunk passes (recycles / loops)	1	10	10
Sampling steps	100	200	68
Diffusion samples per seed	5	5	5
MSA depth	8	official caps	1,024
Precision	bf16	fp32	bf16
Execution path	compile + adaLN	shipped attention	fused Triton

Table 5: Inference configuration in test-time scaling experiments

SimpleFold2. We follow the sampler as mentioned in Appendix B with ($\sigma_{\max} = 160$, $\sigma_{\min} = 4 \times 10^{-4}$, $\rho = 7$, $\sigma_{\text{data}} = 16$, $\gamma_0 = 0.8$, $\gamma_{\min} = 1$, noise scale 1.003, step scale 1.5) at 100 sampling steps. Each sampler call draws its own MSA subsample, so seeds differ in both the MSA draw and the flow noise. We also adapt 10% random token masking when extract embedding from ESMC as suggested by ESMFold2-fast (Candido et al., 2026). We run SimpleFold2 with bf16 and `torch.compile` on the trunks.

G TEST-TIME SCALING: ANTIBODY–ANTIGEN AND PROTEIN–LIGAND

Fig. 16 shows the test-time scaling results on antibody–antigen and protein–ligand from targets in FoldBench. On both categories SimpleFold2 is above OpenFold3 at every matched wall-clock budget, and below ESMFold2 at every matched budget. On antibody–antigen SimpleFold2 reaches 78.0% for $4{,}547$ s, where OpenFold3 has reached 75.2% and ESMFold2 85.7%; on protein–ligand it reaches 77.5% for $1{,}428$ s, against OpenFold3's 76.1% and ESMFold2's 81.4%. A defining feature of SimpleFold2 is that the order of magnitude efficiency increase often compensates for losses in single-shot performance.

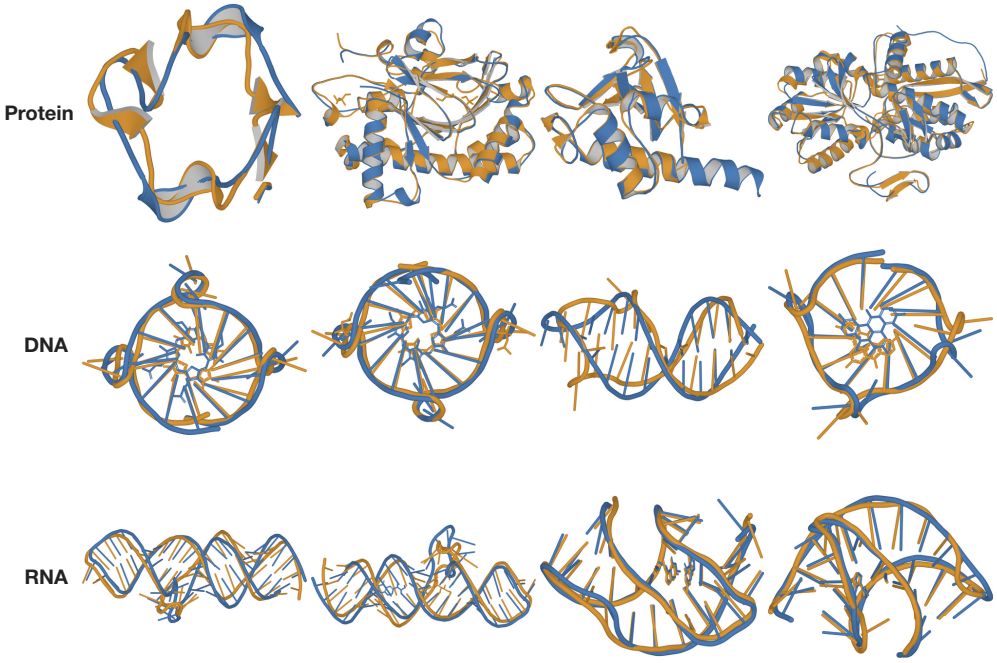


Figure 13: **Structure overlays** between **predicted** and **ground truth** across protein, RNA, and DNA monomers.

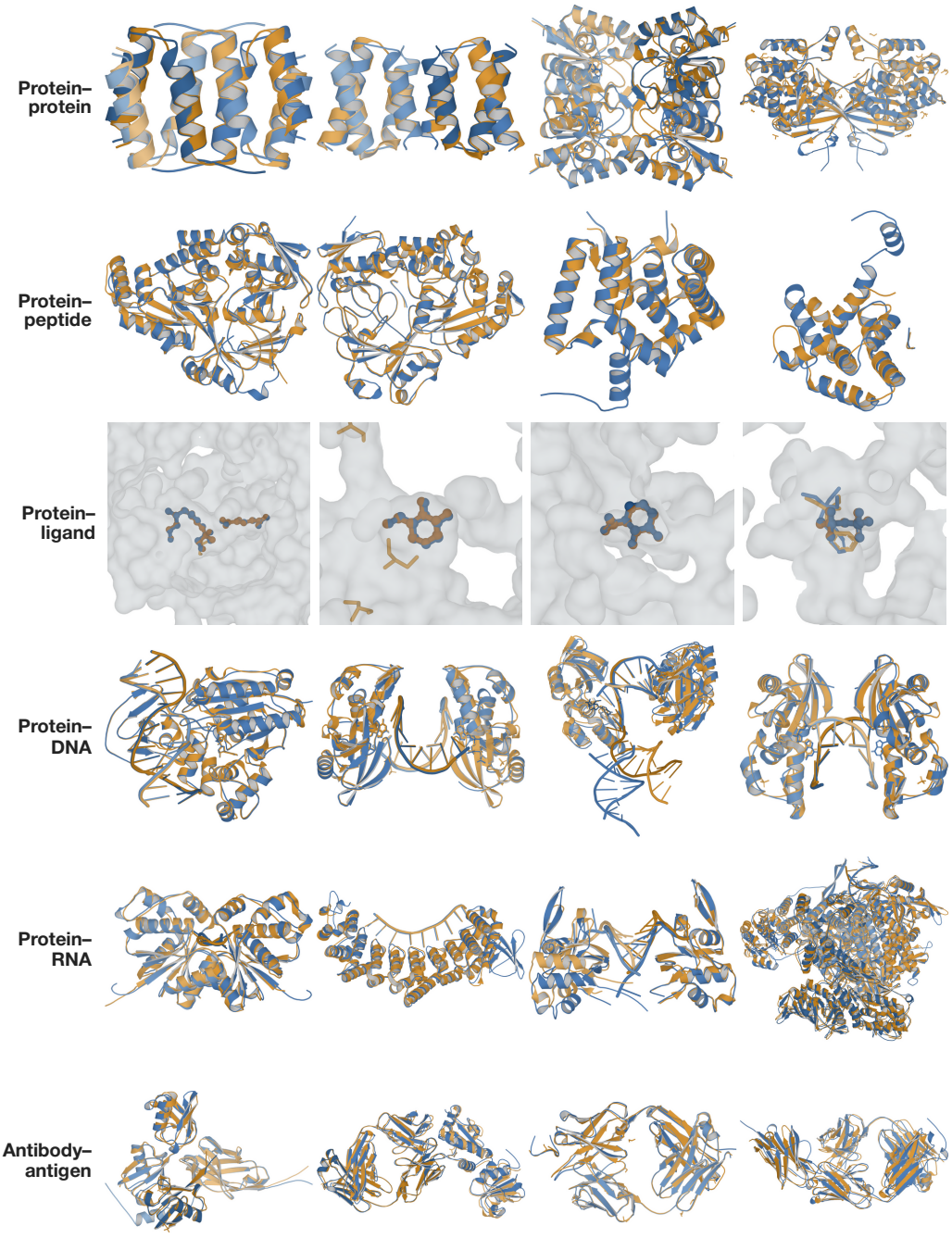


Figure 14: **Structure overlays** between **predicted** and **ground truth** interacting complexes.

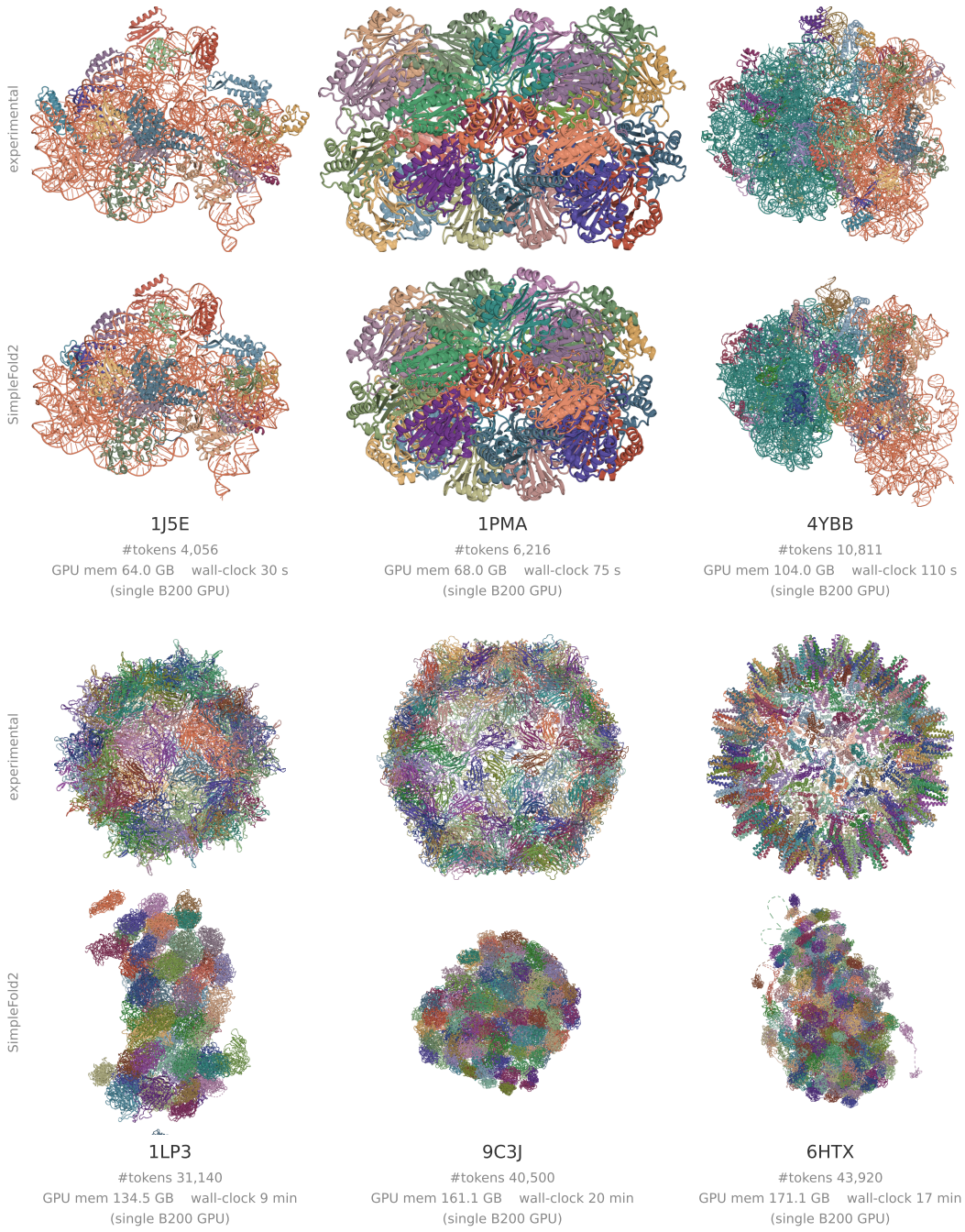


Figure 15: Examples of SimpleFold2 predictions on large protein assemblies.

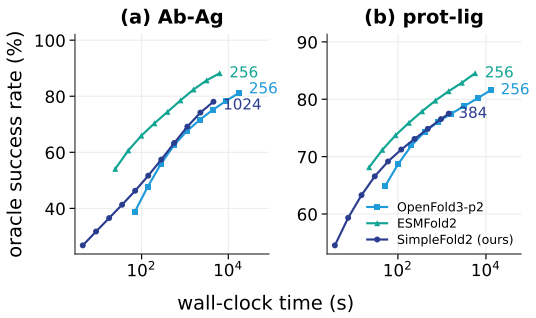


Figure 16: Oracle success rate against single-GPU wall-clock time per target, for k independently drawn seeds scored best-of-k, on the two categories held out of the main text. As in the main figure, one seed is one encoding-trunk pass feeding 5 diffusion samples. We run the baselines to a max of 256 seeds and SimpleFold2 up to 1024 seeds. At $\gg 10^4$ seconds, running more seeds on the baselines becomes computationally intractable. Success rate is defined as DockQ ≥ 0.23 in **(a)** and RMSD < 2 Å *and* lDDT-PLI > 0.8 in **(b)**. All three models are scored on the intersection of targets they all predicted.