PROTEINA-PRONTO: FAST DE NOVO PROTEIN DESIGN AND STRUCTURE PREDICTION VIA DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Accurate flow-matching protein design models and diffusion-based structure prediction models are bottlenecked by the hundreds of network evaluations required at inference time. This restricts large-scale discovery campaigns that require generating and scoring between 10^4 and 10^6 candidates per target. We introduce *Proteina-Pronto*, a framework for accelerating atomistic generation and folding-based evaluation. For generation, we replace low-temperature stochastic sampling with a deterministic ODE using Temporal Score Rescaling, then couple the modality-specific denoising schedules to a shared global time for diffusion distillation. This enables a $16\times$ reduction in sampling steps for both unconditional sequence-structure co-design and target-conditioned binder design, with design performance comparable to the original teacher models. For folding-based evaluation, we distill Boltz-2's diffusion module and accelerate its computationally expensive Pairformer trunk through representation distillation, achieving a $3.9\times$ end-to-end wall-clock speedup while retaining approximately 99% of its original lDDT on the PDB test set. Together, these accelerations enable more efficient inference-time scaling with best-of-N sampling and guided search, yielding up to $6\times$ as many unique successful designs on hard targets within the same total GPU budget as the original pipeline.

1 INTRODUCTION

Atomistic protein generative models have achieved strong performance in producing sequences and full-atom structures across multiple design tasks (Didi et al., 2026b; Geffner et al., 2026; Qu et al., 2024). Similarly, high-fidelity folding models (Abramson et al., 2024; Ahdritz et al., 2024; Jumper et al., 2021) have become indispensable for screening and scoring the large candidate pools produced in design campaigns (Cao et al., 2022; Watson et al., 2023; Didi et al., 2026a). The overall throughput is constrained by inference costs: diffusion-based models require hundreds of network evaluations per sample, and folding models rely on computationally expensive trunk networks. We introduce *Proteina-Pronto*, a framework for accelerating both design and folding models in protein design. We use diffusion distillation to reduce sampling costs in design and folding models, and replace the folding model's expensive trunk with a more efficient architecture trained through representation distillation.

For generation, we build on La-Proteina (Geffner et al., 2026), an atomistic protein generative model based on flow matching that represents proteins through explicit α-carbon coordinates and continuous per-residue latent variables capturing sequence and side-chain detail. Proteina-Complexa extends this formulation to target-conditioned binder design (Didi et al., 2026b). Both models rely on low-temperature stochastic sampling for high sample quality, achieved by downscaling the diffusion noise of the underlying SDE (Geffner et al., 2025; Lin et al., 2024). This reliance on stochasticity imposes two penalties on inference throughput. First, SDE-based samplers often require more integration steps than their deterministic counterparts based on ODEs (Karras et al., 2022). Second, trajectory distillation methods, which facilitate few-step inference, presuppose a deterministic sampler (Boffi et al., 2025; Chen et al., 2025a; Geng et al., 2025; Song et al., 2023).

We accelerate these models with a two-stage approach (Fig. 1a). First, we replace the stochastic sampler with a deterministic ODE using Temporal Score Rescaling (TSR) (Xu et al., 2025), which modifies the flow-matching velocity to emulate the high-fidelity characteristics of low-temperature SDE samplers within an ODE. Second, we adapt diffusion distillation to the product-space formulation shared by La-Proteina and Proteina-Complexa. These methods define the generative process as a system of two coupled modalities that denoise α-carbon coordinates and per-residue latents under

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.

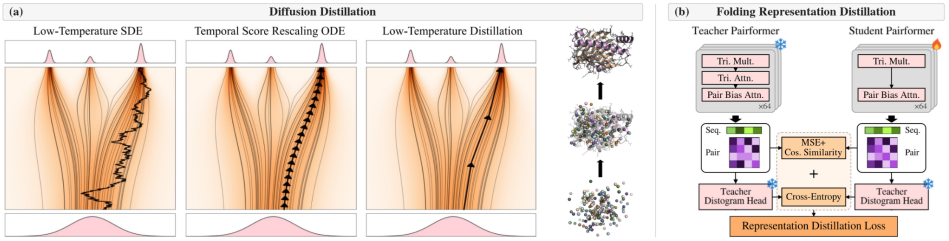

Figure 1: ***Proteina-Pronto* accelerates protein generation and folding. (a) Diffusion acceleration:** We combine Temporal Score Rescaling (TSR) and diffusion distillation to accelerate sampling in both protein design and diffusion-based folding models. For the low-temperature setting shown, TSR replaces stochastic sampling with a deterministic ODE, whose trajectories are distilled into a few-step student. **(b) Folding trunk acceleration:** Representation distillation trains a more efficient student trunk to reproduce the teacher's representations.

separate time schedules. This asymmetric pacing is essential for quality, yet diffusion distillation is often formulated for a single global time. We address this mismatch by coupling the per-modality schedules to a shared global time. We then show that the resulting velocity becomes an explicit per-modality rescaling of the original one, enabling the direct distillation of the coupled TSR-ODE.

For unconditional generation, *Proteina-Pronto* achieves performance comparable to the original 256-step La-Proteina SDE sampler (Geffner et al., 2026) with 16 steps, a 16× reduction in network evaluations. At 8 steps (32× fewer evaluations), *Proteina-Pronto* retains approximately 80% of the baseline's co-designability and diversity metrics. Single-step generation achieves 77.4% backbone designability when sequences are redesigned with ProteinMPNN (Dauparas et al., 2022). For binder design, *Proteina-Pronto* reduces sampling from 256 to 16 steps while retaining Proteina-Complexa performance. Our accelerated models enable more efficient inference-time scaling for binder design, including best-of-N sampling and beam search (Didi et al., 2026b). Depending on the evaluation model, our best configurations yield approximately 2-6× as many unique successful designs on hard targets as the corresponding original pipelines, within the same GPU budget for generation and evaluation.

For folding models, we accelerate Boltz-2 (Passaro et al., 2025), whose inference cost is dominated by the Pairformer trunk and diffusion sampling. We distill its diffusion module into a 4-step model using π-Flow (Chen et al., 2025a), and train a more efficient Pairformer without triangle-attention layers through representation distillation (see Fig. 1b). Together, these changes yield a 3.9× end-to-end inference speedup while retaining ~99% of the original model's lDDT on the PDB test set. We further use the accelerated model for faster reward computation during inference-time scaling and for hallucination-based binder design, achieving end-to-end acceleration of ~2.3× for the latter.

Contributions: (i) We propose replacing low-temperature SDE sampling with deterministic TSR-ODEs, reducing sampling steps by 4× for unconditional generation and binder design while retaining competitive performance; (ii) We show how diffusion distillation can be applied to multimodal product spaces through schedule coupling, and distill low-temperature TSR trajectories into students that achieve comparable performance in unconditional generation and binder design with 16× fewer sampling steps than the original SDE samplers; (iii) We accelerate Boltz-2 through diffusion and representation distillation, achieving a 3.9× wall-clock speedup while retaining approximately 99% of the original model's lDDT. The accelerated model enables faster reward evaluation during inference-time search and more efficient hallucination-based binder design; (iv) *Proteina-Pronto* enables more efficient inference-time scaling for binder design, yielding up to 6× as many unique successful designs per GPU-hour on hard targets by combining accelerated generation and folding-based evaluation.

2 Preliminaries and Related Work

Flow matching (Albergo et al., 2023; Lipman et al., 2023) trains a velocity network $\mathbf{v}_\theta(\mathbf{x}_t, t)$ to match the conditional expectation $\mathbb{E}[\alpha_t'\,(\mathbf{x}_1 - \mathbf{x}_0) \mid \mathbf{x}_t]$ of the interpolant $\mathbf{x}_t = \alpha_t\,\mathbf{x}_1 + (1-\alpha_t)\,\mathbf{x}_0$ between clean data $\mathbf{x}_1$ and Gaussian noise $\mathbf{x}_0 \sim \mathcal{N}(0, I)$. Sampling integrates the ODE $\mathrm{d}\mathbf{x}_t = \mathbf{v}_\theta(\mathbf{x}_t, t)\,\mathrm{d}t$ from $t = 0$ to $t = 1$. For Gaussian noise, the score and velocity are related by $\nabla_{x_t} \log p_t(\mathbf{x}_t) \approx \zeta_\theta(\mathbf{x}_t, t) = ((\alpha_t/\alpha_t')\mathbf{v}_\theta(\mathbf{x}_t, t) - \mathbf{x}_t)/(1-\alpha_t)$ (Zheng et al., 2023), enabling SDE sampling via $\mathrm{d}\mathbf{x}_t = \big[\,\mathbf{v}_\theta(\mathbf{x}_t, t) + g(t)\,\zeta_\theta(\mathbf{x}_t, t)\,\big]\,\mathrm{d}t + \sqrt{2g(t)\eta}\,\mathrm{d}w_t$. Protein design often relies on "low-temperature" sampling (Lin et al., 2024), achieved by setting $\eta < 1$. TSR (Xu et al., 2025) provides a deterministic alternative by rescaling the ODE velocity to emulate this low-temperature behavior (Sec. 3.1.1).

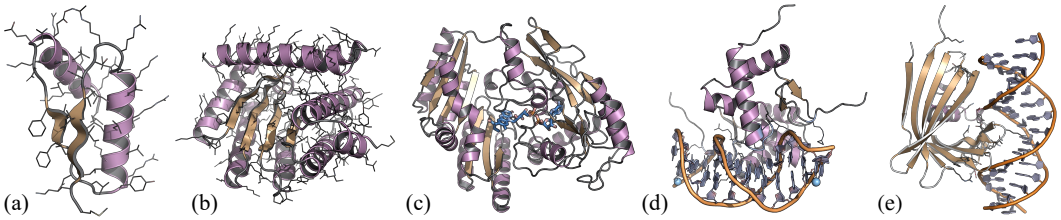

Figure 2: **Proteins produced by *Proteina-Pronto*. (a)** Co-designable 1-step protein generated by *Proteina-Pronto*$_{\pi\text{-}Flow}$. **(b)** Co-designable 4-step protein generated by *Proteina-Pronto*$_{TVM}$. **(c)** Protein-ligand complex predicted by 4-step Boltz-2$_{\text{distill}}$ (lDDT=0.882). **(d)** Protein-DNA complex predicted by 4-step Boltz-2$_{\text{distill}}$ (lDDT=0.961). **(e)** DNA-binding protein designed via hallucination with BoltzDesign using 4-step Boltz-2$_{\text{distill}}$.

Distillation methods for generative models range from distribution matching (Yin et al., 2024a;b) to trajectory-based optimization (Boffi et al., 2025; Chen et al., 2025a). We explore both approaches. Flow Map methods aim to learn the average velocity of the teacher's ODE between times $t < r$

$$\mathbf{u}(\mathbf{x}_t, t, r) = \tfrac{1}{r-t} \int_t^r \mathbf{v}_\theta(\mathbf{x}_\tau, \tau)\, \mathrm{d}\tau \tag{1}$$

with a student network $\mathbf{u}_\phi$. Sampling applies $\mathbf{x}_r = \mathbf{x}_t + (r-t)\mathbf{u}_\phi(\mathbf{x}_t, t, r)$ over a chosen time discretization, allowing the number of steps to be selected at inference to balance speed and accuracy.

π-Flow (Chen et al., 2025a), another trajectory-distillation method, uses a network to predict a velocity policy for an entire sampling segment, which can then be integrated without further network evaluations. On-policy imitation matches this policy to the teacher velocity along student trajectories.

Distribution Matching Distillation (DMD) trains a student to match the teacher's distribution through an approximate reverse-KL objective over noise-perturbed samples. Its gradient uses the difference between velocity predictions from the teacher and an auxiliary model trained on student samples. The original DMD includes a regression regularizer to mitigate mode collapse (Yin et al., 2024b).

La-Proteina is an atomistic protein generative model based on flow matching over a dual-modality product space. It represents a protein as $[\mathbf{x}, \mathbf{z}]$, where $\mathbf{x} \in \mathbb{R}^{L\times 3}$ are α-carbon coordinates and $\mathbf{z} \in \mathbb{R}^{L\times 8}$ latent variables capturing sequence and side-chain detail (a decoder network is used to obtain atomistic proteins from $[\mathbf{x}, \mathbf{z}]$). Training uses independent rectified interpolants for each modality

$$\mathbf{x}_{t_\mathbf{x}} = t_\mathbf{x}\, \mathbf{x}_1 + (1 - t_\mathbf{x})\, \mathbf{x}_0, \qquad \mathbf{z}_{t_\mathbf{z}} = t_\mathbf{z}\, \mathbf{z}_1 + (1 - t_\mathbf{z})\, \mathbf{z}_0. \tag{2}$$

A joint denoiser network $\mathbf{v}_\theta(\mathbf{x}_{t_x}, \mathbf{z}_{t_z}, t_x, t_z) = [\mathbf{v}^\mathbf{x}_\theta, \mathbf{v}^\mathbf{z}_\theta]$ is trained to predict the target velocities

$$\mathbf{v}^\mathbf{x}(\mathbf{x}_{t_\mathbf{x}}, \mathbf{z}_{t_\mathbf{z}}, t_\mathbf{x}, t_\mathbf{z}) = \mathbb{E}[\mathbf{x}_1 - \mathbf{x}_0 \mid \mathbf{x}_{t_\mathbf{x}}, \mathbf{z}_{t_\mathbf{z}}, t_\mathbf{x}, t_\mathbf{z}], \quad \mathbf{v}^\mathbf{z}(\mathbf{x}_{t_\mathbf{x}}, \mathbf{z}_{t_\mathbf{z}}, t_\mathbf{x}, t_\mathbf{z}) = \mathbb{E}[\mathbf{z}_1 - \mathbf{z}_0 \mid \mathbf{x}_{t_\mathbf{x}}, \mathbf{z}_{t_\mathbf{z}}, t_\mathbf{x}, t_\mathbf{z}] \tag{3}$$

across all $(t_\mathbf{x}, t_\mathbf{z}) \in [0, 1]^2$. Inference uses coupled low-temperature SDEs with asymmetric schedules that denoise the backbone faster than the latents (Sec. B). Proteina-Complexa extends this formulation to target-conditioned binder design and supports inference-time scaling with folding-based rewards.

Diffusion-based folding models such as AlphaFold3 (Abramson et al., 2024) and Boltz-2 (Passaro et al., 2025) combine a trunk that constructs sequence and pair representations with a diffusion module that generates atomic coordinates conditioned on these representations. These models are widely used in protein design pipelines for evaluating candidates based on predicted structures and confidence scores, as well as for hallucination-based iterative design. Inference costs arise from both diffusion sampling and expensive trunk computation, which includes triangle operations and recycling. Because the trunk operates outside the diffusion module, reducing denoising steps leaves its cost unchanged.

Related work. Diffusion-based protein design went from backbone generation (Ingraham et al., 2023; Watson et al., 2023; Geffner et al., 2025; Yim et al., 2023; Lin & Alquraishi, 2023; Lin et al., 2024; Bose et al., 2024; Huguet et al., 2024) to joint sequence-structure generation (Ren et al., 2024; Campbell et al., 2024) and fully atomistic models (Qu et al., 2024; Chu et al., 2024; Geffner et al., 2026; Stark et al., 2025). Most of these models are trained on combinations of the PDB (Berman et al., 2000) and AFDB (Varadi et al., 2021). Closely related to our work, Xie et al. (2025) accelerate the backbone model Proteina (Geffner et al., 2025) using score identity distillation (Zhou et al., 2024). For folding, PairMixer (Ouyang-Zhang et al., 2026) explored removing the expensive triangle attention layers. DCFold (Zhang et al., 2026) and Protenix-Mini variants (Gong et al., 2025; Qiang et al., 2025) explore the acceleration of diffusion-based structure prediction and reductions in trunk recycling or depth.

Concurrent work DeCAF (Scarpellini et al., 2026) explores flow map distillation of a protein-ligand co-folding model. Its few-step results rely on inference-time search over reward-steered particles; without search its sampler needs 20 steps to match the 200-step teacher, whereas Boltz-2_{distill} reaches a higher lDDT than the 200-step regular-temperature Boltz-2 with only 4 steps and no search. To the best of our knowledge, Boltz-2_{distill} is the first to successfully distill from a low-temperature teacher model.

3 *Proteina-Pronto*: Accelerated Generation and Folding

Proteina-Pronto accelerates atomistic protein generation and structure prediction. For generation, we replace low-temperature SDEs with deterministic TSR-ODEs, couple the modality schedules, and distill the resulting trajectories. For structure prediction, we distill Boltz-2's diffusion module into a few-step sampler and train a more efficient Pairformer trunk through representation distillation.

3.1 Accelerated Atomistic Protein Generation

Notation. We use $\mathbf{y} \in \{\mathbf{x}, \mathbf{z}\}$ to denote either modality, where $\mathbf{x} \in \mathbb{R}^{L\times 3}$ represents α-carbon coordinates and $\mathbf{z} \in \mathbb{R}^{L\times 8}$ represents per-residue latents. Parameter subscripts identify learned networks (e.g., $\mathbf{v}_\theta$ or $\mathbf{u}_\phi$); quantities without these subscripts denote their targets. Superscripts identify modality-specific components (e.g., $\mathbf{v}^{\mathbf{x}}$), while their absence denotes the joint quantity (e.g., $\mathbf{v} = [\mathbf{v}^{\mathbf{x}}, \mathbf{v}^{\mathbf{z}}]$).

3.1.1 From Low-Temperature SDEs to Deterministic TSR-ODEs

For few-step sampling, we replace the low-temperature SDEs of La-Proteina and Proteina-Complexa (Eq. (17)) with deterministic ODEs using Temporal Score Rescaling (TSR), which rescales the score component of the velocity field. We extend TSR to independent per-modality schedules through anisotropic rescaling. Given the teacher velocity $\mathbf{v}_\theta = [\mathbf{v}^{\mathbf{x}}_\theta, \mathbf{v}^{\mathbf{z}}_\theta]$, the TSR-rescaled components are

$$\mathbf{v}^{\mathbf{y}}_\theta(\mathbf{x}_{t_\mathbf{x}}, \mathbf{z}_{t_\mathbf{z}}, t_\mathbf{x}, t_\mathbf{z}; \gamma_\mathbf{y}) = (1-\gamma_\mathbf{y}(t_\mathbf{y}))\tfrac{\mathbf{y}_{t_\mathbf{y}}}{t_\mathbf{y}} + \gamma_\mathbf{y}(t_\mathbf{y})\,\mathbf{v}^{\mathbf{y}}_\theta(\mathbf{x}_{t_\mathbf{x}}, \mathbf{z}_{t_\mathbf{z}}, t_\mathbf{x}, t_\mathbf{z}), \qquad \mathbf{y} \in \{\mathbf{x}, \mathbf{z}\}, \tag{4}$$

where the scaling factor $\gamma_\mathbf{y}(t_\mathbf{y})$ is set using the signal-to-noise ratio $\xi_\mathbf{y}$ and parameters $(k_\mathbf{y}, \sigma_\mathbf{y})$

$$\gamma_\mathbf{y}(t_\mathbf{y}) = \frac{\xi_\mathbf{y}(t_\mathbf{y})\sigma^2_\mathbf{y} + 1}{\xi_\mathbf{y}(t_\mathbf{y})\sigma^2_\mathbf{y}/k_\mathbf{y} + 1}, \qquad \xi_\mathbf{y}(t_\mathbf{y}) = \frac{t^2_\mathbf{y}}{(1-t_\mathbf{y})^2}. \tag{5}$$

The rescaled velocities from Eq. (4) define the coupled ODEs

$$\mathrm{d}\mathbf{y}_{t_\mathbf{y}} = \mathbf{v}^{\mathbf{y}}_\theta(\mathbf{x}_{t_\mathbf{x}}, \mathbf{z}_{t_\mathbf{z}}, t_\mathbf{x}, t_\mathbf{z}; \gamma_\mathbf{y})\,\mathrm{d}t_\mathbf{y}, \qquad \mathbf{y} \in \{\mathbf{x}, \mathbf{z}\}. \tag{6}$$

These ODEs can be integrated along any monotonic path from $[0, 0]$ to $[1, 1]$ in the $[t_\mathbf{x}, t_\mathbf{z}]$ unit square. We follow the asymmetric schedules of Geffner et al. (2026), which denoise the α-carbon coordinates faster than the latents (Sec. B). This sampler change requires no retraining. As shown in Sec. 4, the TSR sampler reduces sampling steps by $8\times$ for unconditional generation while preserving co-designable structural diversity, and by $4\times$ for binder design while retaining competitive performance.

3.1.2 Coupling Modality Schedules

While La-Proteina and Proteina-Complexa are trained to handle any times $(t_\mathbf{x}, t_\mathbf{z}) \in [0, 1]^2$, distillation often targets dynamics parameterized by a single time variable. We therefore restrict the process to a path parameterized by a single time $t \in [0, 1]$, defining the per-modality coupled schedules as $t_\mathbf{x} = \alpha_\mathbf{x}(t)$ and $t_\mathbf{z} = \alpha_\mathbf{z}(t)$, where each $\alpha_\mathbf{y} : [0, 1] \to [0, 1]$ is a monotonically increasing satisfying $\alpha_\mathbf{y}(0) = 0$ and $\alpha_\mathbf{y}(1) = 1$. This couples the schedules for both modalities through the single time t.

Mapping Teacher Velocity to the Coupled Schedules From a Flow Matching lens, coupling the schedules for both modalities induces a coupled interpolant governed by the global time variable t

$$\mathbf{y}_t = \alpha_\mathbf{y}(t)\mathbf{y}_1 + (1-\alpha_\mathbf{y}(t))\mathbf{y}_0, \qquad \mathbf{y} \in \{\mathbf{x}, \mathbf{z}\}, \tag{7}$$

where $\mathbf{y}_1$ is clean data and $\mathbf{y}_0$ is Gaussian noise. Unlike uni-modal formulations, these interpolants are coupled by a global time t, while remaining at distinct, modality-specific noise levels determined by $\alpha_\mathbf{x}(t)$ and $\alpha_\mathbf{z}(t)$. Differentiating Eq. (7) w.r.t t, we recover the velocity of the coupled system

$$\tilde{\mathbf{v}}^{\mathbf{y}}(\mathbf{x}_t, \mathbf{z}_t, t) = \mathbb{E}\left[\alpha'_\mathbf{y}(t)(\mathbf{y}_1 - \mathbf{y}_0) \,|\, \mathbf{x}_t, \mathbf{z}_t, t\right] = \alpha'_\mathbf{y}(t)\mathbf{v}^{\mathbf{y}}(\mathbf{x}_t, \mathbf{z}_t, \alpha_\mathbf{x}(t), \alpha_\mathbf{z}(t)), \quad \mathbf{y} \in \{\mathbf{x}, \mathbf{z}\}, \tag{8}$$

where the second equality follows from Eq. (3). The pretrained teacher can therefore be reused without retraining: we evaluate it at the coupled modality times and multiply each velocity component

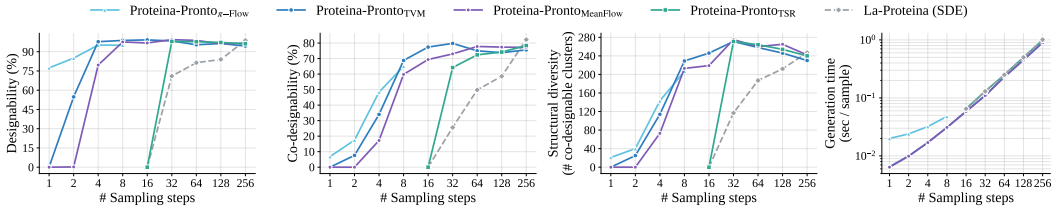

Figure 3: **Faster generation of diverse atomistic proteins.** *From left to right:* backbone designability after sequence redesign; all-atom co-designability, measuring consistency between generated sequences and structures; structural diversity, measured by number of co-designable structural clusters; and per-sample wall-clock generation time. Our evaluations use 500 proteins. For the backbone-only method of Xie et al. (2025), we only report designability. Timings are unavailable, but its Proteina-based architecture suggests per-step cost comparable to La-Proteina SDE. Generation times include decoding. Full numerical results are provided in Tab. 5.

by its schedule derivative $\alpha'_{\mathbf{y}}(t)$. We retain the asymmetric schedules that denoise the α-carbon coordinates $\mathbf{x}$ at a faster rate than the latents $\mathbf{z}$ from Geffner et al. (2026), specified in Sec. B.

Combining schedule coupling with the per-modality TSR rescaling gives the following global-time teacher velocity, with the TSR factors evaluated at each modality's local time $t_{\mathbf{y}} = \alpha_{\mathbf{y}}(t)$

$$\tilde{\mathbf{v}}^{\mathbf{y}}_\theta(\mathbf{x}_t, \mathbf{z}_t, t; \gamma_{\mathbf{y}}) = \alpha'_{\mathbf{y}}(t)\, \mathbf{v}^{\mathbf{y}}_\theta(\mathbf{x}_t, \mathbf{z}_t, \alpha_{\mathbf{x}}(t), \alpha_{\mathbf{z}}(t); \gamma_{\mathbf{y}}), \qquad \mathbf{y} \in \{\mathbf{x}, \mathbf{z}\}. \tag{9}$$

3.1.3 Distilling the TSR-Coupled ODE

We distill the generative dynamics from Eq. (9) with multiple diffusion distillation methods.

Terminal Velocity Matching. We optimize a student $\tilde{\mathbf{u}}_\phi = [\tilde{\mathbf{u}}^{\mathbf{x}}_\phi, \tilde{\mathbf{u}}^{\mathbf{z}}_\phi]$ to approximate the average velocity of the teacher's flow (Eq. (9)), minimizing the multi-modal TVM objective given by

$$\mathcal{L}(\phi) = \mathbb{E}\left[\sum_{\mathbf{y}\in\{\mathbf{x},\mathbf{z}\}} \left\| \tilde{\mathbf{u}}^{\mathbf{y}}_\phi(\mathbf{x}_t, \mathbf{z}_t, t, r) + \underbrace{(r-t)\frac{\mathrm{d}}{\mathrm{d}r}\tilde{\mathbf{u}}^{\mathbf{y}}_\phi(\mathbf{x}_t, \mathbf{z}_t, t, r) - \tilde{\mathbf{v}}^{\mathbf{y}}_\theta(\hat{\mathbf{x}}_r, \hat{\mathbf{z}}_r, r; \gamma_{\mathbf{y}})}_{\text{stop_grad}} \right\|^2 \right] \tag{10}$$

where $\hat{\mathbf{y}}_r = \mathbf{y}_t + (r-t)\tilde{\mathbf{u}}^{\mathbf{y}}_\phi$ is the prediction at r, and $\tilde{\mathbf{v}}^{\mathbf{y}}_\theta(\cdot; \gamma_{\mathbf{y}})$ is the teacher velocity. Intuitively, this objective matches the student's gradient at time r against the teacher velocity. To ensure training stability and account for the schedule derivatives in Eq. (8), we parameterize the student as

$$\tilde{\mathbf{u}}^{\mathbf{y}}_\phi(\mathbf{x}_t, \mathbf{z}_t, t, r) = \tfrac{\alpha_{\mathbf{y}}(r) - \alpha_{\mathbf{y}}(t)}{r-t}\, \mathbf{u}^{\mathbf{y}}_\phi(\mathbf{x}_t, \mathbf{z}_t, t, r), \tag{11}$$

which implies that $\mathbf{u}^{\mathbf{y}}_\phi$ effectively learns the teacher's average velocity normalized by the schedule displacement rather than global time, $\mathbf{u}^{\mathbf{y}}_\phi(\mathbf{x}_t, \mathbf{z}_t, t, r) \approx \frac{1}{\alpha_{\mathbf{y}}(r) - \alpha_{\mathbf{y}}(t)} \int_t^r \tilde{\mathbf{v}}^{\mathbf{y}}_\theta(\mathbf{x}_\tau, \mathbf{z}_\tau, \tau; \gamma_{\mathbf{y}})\, \mathrm{d}\tau$, ensuring robustness to the asymmetric denoising speeds of the backbone and latents. By initializing the student weights with the teacher parameters ($\phi \leftarrow \theta$), this parameterization satisfies $\lim_{r\to t} \tilde{\mathbf{u}}^{\mathbf{y}}_\phi = \tilde{\mathbf{v}}^{\mathbf{y}}_\theta$ without rescaling (i.e., $(k_{\mathbf{y}}, \sigma_{\mathbf{y}}) = (1, 1)$), ensuring the student begins distillation as a first-order accurate approximation of the teacher's flow, improving convergence. We transition to the target low-temperature regime by linearly annealing $(k_{\mathbf{y}}, \sigma_{\mathbf{y}})$ from $(1, 1)$ to their target values over the first 20k training steps.

π-Flow. *Proteina-Pronto$_{\pi\text{-Flow}}$* partitions the sampling interval $t \in [0, 1]$ into M segments of length $1/M$, with M fixed before training. Each segment spans $[t_{\text{src}}, t_{\text{dst}}]$, where $t_{\text{dst}} = t_{\text{src}} + 1/M$. Given the state at the segment start, a single evaluation of $G_\phi(\mathbf{x}_{t_{\text{src}}}, \mathbf{z}_{t_{\text{src}}}, t_{\text{src}})$ predicts velocities for both modalities at N equally spaced times within the segment. Linear interpolation between these predictions defines the policy $\pi^{\mathbf{y}}(\tau; \mathbf{x}_{t_{\text{src}}}, \mathbf{z}_{t_{\text{src}}}, t_{\text{src}})$, which supplies a velocity at any time $\tau \in [t_{\text{src}}, t_{\text{dst}}]$ without further network evaluations. This adapts the DX policy of π-Flow to predict coupled velocities instead of clean data. During sampling, we advance both modalities across a segment using Euler substeps, taking each substep's velocity from the predicted policy. At each subsequent segment start, we reevaluate G_ϕ on the resulting state, giving M network calls in total.

Training uses on-policy imitation within segments. We get segment-start states from the flow matching interpolant (Eq. (7)), predict the policy, and roll out its detached version π_D to a random $t \in [t_{\text{src}}, t_{\text{dst}}]$. At the resulting state, we match the policy prediction to the TSR-rescaled coupled teacher velocity:

$$\mathbf{y}_t = \mathbf{y}_{t_{\text{src}}} + \int_{t_{\text{src}}}^t \pi^{\mathbf{y}}_D(\tau)\, \mathrm{d}\tau, \qquad \mathcal{L}(\phi) = \tfrac{1}{2} \sum_{\mathbf{y}\in\{\mathbf{x},\mathbf{z}\}} \left\| \tilde{\mathbf{v}}^{\mathbf{y}}_\theta(\mathbf{x}_t, \mathbf{z}_t, t; \gamma_{\mathbf{y}}) - \pi^{\mathbf{y}}(t) \right\|_2^2. \tag{12}$$

Gradients flow through $\pi^{\mathbf{y}}(t)$ only, with the rollout and teacher targets detached from the computation graph. We initialize G_ϕ from the teacher and replicate its output channels across the N grid points.

We also experiment with MeanFlow (Geng et al., 2025), an alternative flow-map formulation, and DMD (Yin et al., 2024b). Both underperform TVM and π-Flow in unconditional atomistic generation (results in Sec. 4.1). We defer details of their adaptation to our coupled multi-modal setting to Sec. A.

Architecture. All student models retain the teacher's architecture, with minor additions: a second-time conditioning for TVM and MeanFlow, and an expanded linear output heads for π-Flow. As shown in Sec. 4, these step reductions yield substantial savings in wall-clock generation time.

Self-conditioning. Self-conditioning provides a clean-sample prediction as an additional input, and is critical for Proteina-Complexa performance (Sec. C.1). Naive student training obtains this prediction through an extra model call at the current noisy state, whereas inference uses the preceding step's prediction. In traditional flow matching, consecutive steps typically operate at similar noise levels. We hypothesize that this training-inference mismatch becomes more consequential for few-step students, whose steps span larger changes in noise level. We therefore propose *self-forcing* self-conditioning, a simple modification for few-step students that better aligns training with inference. During training, we simulate a preceding student step and use its resulting state and clean-sample prediction as the state and conditioning inputs for the current step. The detailed formulation is provided in Sec. C.2.

3.1.4 Accelerated Inference-Time Scaling

Didi et al. (2026b) demonstrated the benefits of inference-time scaling for binder design, exploring both best-of-N sampling and guided search. We use *Proteina-Pronto* models based on TSR and flow-map distillation to reduce sampling costs across these strategies, targeting more unique successful designs within a given compute budget. For best-of-N, which generates independent candidates and ranks them using folding-based rewards, our accelerated samplers directly reduce the cost of candidate generation. Beam search maintains a set of partially denoised candidates. At each round, *trajectory propagation* advances several copies of each candidate to the next search time, while *look-ahead rollouts* complete these continuations for folding-based reward evaluation. The highest-scoring candidates are retained at their intermediate noise level for the next round. We accelerate look-ahead using *Proteina-Pronto*$_{TSR}$ or *Proteina-Pronto*$_{TVM}$, while retaining SDE propagation to generate distinct continuations through stochastic sampling (Sabour et al., 2025). To accelerate propagation as well, we adopt *denoise-re-noise sampling* (Xu et al., 2023; Kim et al., 2024). Fig. 4 compares generation time per sample across these strategies, separating propagation and look-ahead costs. Complete details are provided in Sec. D.

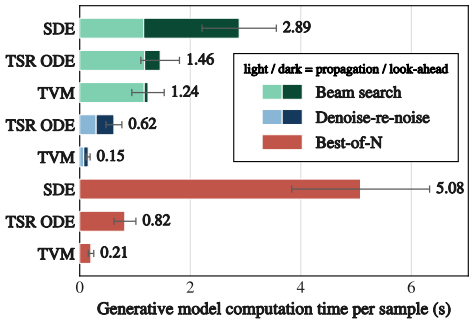

Figure 4: **Efficiency of inference-time scaling methods.** Y-axis: look-ahead method for beam search; propagation & look-ahead method for denoise-re-noise; generation method for best-of-N.

3.2 Folding Model Acceleration

Our profiling of Boltz-2 (Passaro et al., 2025), presented in Sec. G.6, shows that the diffusion module and the Pairformer trunk account for approximately 41% and 32% of inference time, respectively, with the trunk's share increasing with input size. Here we discuss how we distill these two modules.

Few-step diffusion module. We accelerate the diffusion module of Boltz2 (Passaro et al., 2025) using π-Flow. As observed by Passaro et al. (2025), low-temperature sampling improves Boltz2's prediction accuracy, motivating distillation from a low-temperature teacher. However, in our experiments, applying the original sampler's constant step scaling to the probability-flow ODE led to unstable optimization during distillation. We instead apply TSR, which scales up the velocity field less aggressively near pure noise than constant step scaling. The resulting deterministic low-temperature teacher matches the original sampler's accuracy and is more stable to distill. Using π-Flow, we train a four-step student whose policy predicts clean atomic coordinates, matching Boltz2's output parameterization. Training uses on-policy imitation of the TSR-rescaled teacher predictions, with the loss computed in clean-coordinate space. Full architectural and training details are provided in Sec. F.2.

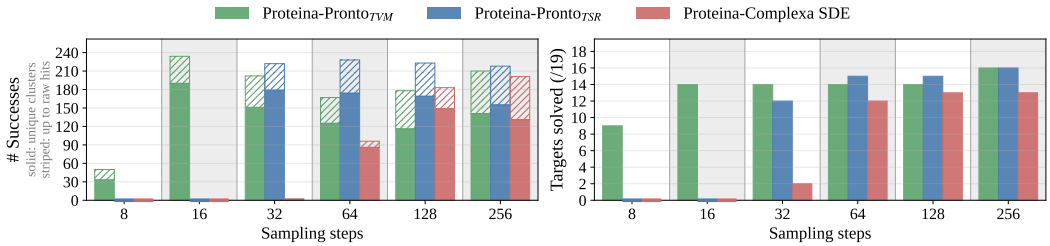

Figure 5: **Faster binder design.** Each method generates 100 samples per target, across 19 targets from prior work. *(left)* Solid bars count structural clusters among successful designs; total bar heights, including striped extensions, count all successful samples. *(right)* Number of targets solved (i.e. at least one successful design).

Pairformer representation distillation. To reduce the Pairformer's cost, we distill it into a more efficient student architecture. Our profiling shows that triangle attention accounts for approximately 59% of the Pairformer's cost. We therefore construct the student by removing triangle-attention layers from every block of the teacher's Pairformer, leaving the remaining architecture unchanged (Fig. 1b). We initialize the retained student layers from the teacher's parameters and keep the MSA module frozen. To train this student trunk, we align its sequence and pair representations at every recycling iteration with the frozen teacher's final representations. We also use an auxiliary distogram loss, which provides structural supervision through the inter-token distance distributions predicted from the final pair representations. The architecture, distillation objectives, and training details are provided in Sec. F.3.

We distill the diffusion module and Pairformer trunk independently and combine them at inference time, without joint training, to obtain Boltz-2$_{\text{distill}}$. This yields a 3.9× inference speedup over Boltz-2.

4 Experiments

We evaluate *Proteina-Pronto* for unconditional atomistic generation, target-conditioned binder design, and structure prediction. Training and sampling details in Sec. F, metric definitions in Sec. E.

4.1 Unconditional Atomistic Generation

We compare La-Proteina with the original SDE sampler (Geffner et al., 2026) with *Proteina-Pronto$_{TSR}$* and students distilled with TVM, MeanFlow, π-Flow and DMD for unconditional monomer generation. We generate 100 samples for each length in $\{50, 100, 150, 200, 250\}$, and report designability, measuring whether a sequence redesigned with ProteinMPNN folds into the generated structure; all-atom co-designability, measuring consistency between the generated structure and its paired sequence; diversity, which counts the unique structural or sequence clusters among designable or co-designable samples; and secondary structure content. We also compare with the backbone-only baseline of Xie et al. (2025), based on Score Identity Distillation (Zhou et al., 2024), using backbone-level metrics.

Fig. 3 compares designability, all-atom co-designability, structural diversity, and generation time across methods; Tab. 5 provides full results, including sequence diversity and secondary-structure content. Compared with the 256-step La-Proteina SDE baseline, *Proteina-Pronto$_{TSR}$* maintains comparable diversity at 32 steps (8× reduction), while *Proteina-Pronto$_{TVM}$* approximately matches baseline performance at 16 steps (16× reduction). *Proteina-Pronto$_{\pi\text{-}Flow}$* retains approximately 80% of baseline all-atom co-designability and diversity at 8 steps (32× reduction), and 60% at 4 steps (64× reduction). *Proteina-Pronto$_{MeanFlow}$* remains competitive at larger budgets but trails *Proteina-Pronto$_{TVM}$* and *Proteina-Pronto$_{\pi\text{-}Flow}$* in co-designability and diversity at low budgets. With ProteinMPNN sequence redesign, *Proteina-Pronto$_{\pi\text{-}Flow}$* achieves 77.4% backbone designability in one generation step, versus 0.4% for Xie et al. (2025). Step reductions yield approximately proportional wall-clock speedups for TSR, TVM, and MeanFlow; despite overhead, *Proteina-Pronto$_{\pi\text{-}Flow}$* achieves approximately 21× and 32× speedups at eight and four steps, respectively. DMD did not yield competitive atomistic co-design: its co-designable samples exhibit severe secondary-structure collapse (84-90% α-helical content, no

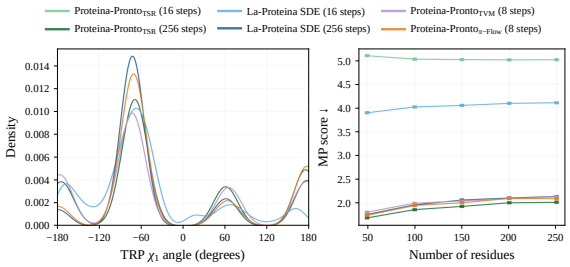

Figure 6: ***Proteina-Pronto* generates high quality structures with fewer steps.** *(left)* Distribution of TRP χ_1 angle generated by different methods. *(right)* Molprobity scores.

β-sheets) and systematically shortened consecutive alpha-carbon distances (3.49-3.63 Å versus approximately 3.8 Å expected). We present all tested variants and complete results in Sec. A.2 and Tab. 2.

We additionally assess atomistic local atomic geometry using side-chain angle distributions and MolProbity (Chen et al., 2015), a validation tool for protein structures that assigns a score combining steric clashes with backbone and side-chain conformational statistics; lower scores indicate better geometry. At eight steps, *Proteina-Pronto*$_{TVM}$ and *Proteina-Pronto*$_{\pi\text{-}Flow}$ achieve MP scores comparable to the original 256-step SDE sampler, whereas SDE and TSR samplers degrade substantially at 16 steps (Fig. 6, right). The students also closely reproduce the 256-step teachers' Tryptophan χ_1 distributions (Fig. 6, left). We include additional structural validation analyses in Secs. G.3 and G.4.

4.2 Target-Conditioned Binder Design

For binder design, we build on TVM, motivated by its unconditional generation performance and its sampling flexibility. Unlike π-Flow, TVM allows the number of sampling steps to be selected at inference time, and models transitions between arbitrary starting and ending times, useful for inference-time search. We design binders for 19 targets from prior work (Pacesa et al., 2025; Zambaldi et al., 2024). We generate 100 samples per target, and adopt the success criteria from Didi et al. (2026b), detailed in Sec. E. Fig. 5 shows that both *Proteina-Pronto*$_{TSR}$ and *Proteina-Pronto*$_{TVM}$ accelerate sampling significantly while yielding competitive binder-design performance over the original 256-step Proteina-Complexa SDE baseline. For instance, *Proteina-Pronto*$_{TVM}$ at 16 and 32 steps solves more tasks and produces more unique successes than the SDE 256-step baseline, providing 8-16$\times$ speedups. Additionally, *Proteina-Pronto*$_{TSR}$ is competitive at 32 steps and surpasses the 256-step SDE baseline at 64 steps, producing more unique successful designs and solving more tasks with 4$\times$ fewer steps.

4.3 Folding Model Acceleration

We compare Boltz-2$_{\text{distill}}$ with the 200-step Boltz-2 sampler using low-temperature (step scale 1.5) and regular-temperature (step scale 1.0). The diffusion module of Boltz-2$_{\text{distill}}$ is distilled from a TSR-based low-temperature teacher (Sec. 3.2). We evaluate all models on the PDB test set curated by Boltz-2 using lDDT, DockQ, protein-ligand lDDT (lDDT-PLI), and ligand RMSD (L-RMSD); metric definitions included in Sec. E.2. All models use three trunk recycling iterations, and the confidence module selects the highest-ranked of five structures per input. As shown in Fig. 7, the four-step Boltz-2$_{\text{distill}}$ achieves an lDDT of 0.811, retaining approximately 99% of the low-temperature baseline's 0.82 and exceeding the regular-temperature baseline's 0.8. Previous approaches reduce sampling steps (Gong et al., 2025) or distill the model (Zhang et al., 2026) using regular-temperature ODE dynamics. While this avoids unstable step scaling during few-step sampling, it does not explicitly preserve the accuracy benefits of low-temperature sampling.

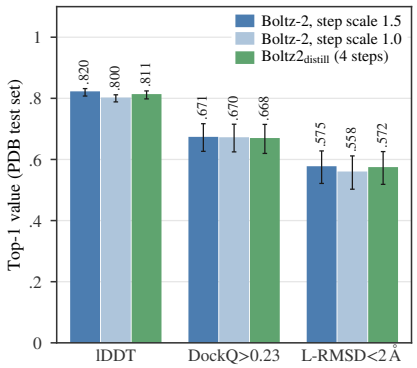

Figure 7: **Accurate folding in four sampling steps.** We compare Boltz-2$_{\text{distill}}$ with 200-step Boltz-2 at low (step scale 1.5) and regular (step scale 1.0) temperature on the PDB test set. The error bars show 95% bootstrap confidence intervals.

Fig. 21 presents results across all metrics on the PDB test set and CASP15. On the PDB, Boltz-2$_{\text{distill}}$ approximately matches the low-temperature Boltz-2 baseline and outperforms the regular-temperature variant in both fraction of ligand predictions with L-RMSD < 2 Å and lDDT-PLI. For protein interfaces, Boltz-2$_{\text{distill}}$ retains more than 99% and 97% of the low-temperature baseline's success rates at DockQ > 0.23 and DockQ > 0.49, respectively. These results are achieved with $3.9\times$ end-to-end and $6\times$ GPU-time speedups on the PDB test set, and $3.3\times$ and $4.9\times$ on CASP15. Additional analysis and ablations isolating the contributions of the distilled trunk and diffusion module are provided in Sec. G.7. We also explore one- and two-step students using π-Flow and parallel decoding distillation (PDD) (Shaul et al., 2026), trading some accuracy (our best models retain 94% and 98% of the low-temperature baseline's lDDT on the PDB test set, respectively) for further acceleration (Sec. F.2.6).

Hallucination-based protein design with Boltz-2$_{\text{distill}}$ We evaluate Boltz-2$_{\text{distill}}$ in BoltzDesign (Cho et al., 2025), a hallucination-based framework that iteratively optimizes binder sequences

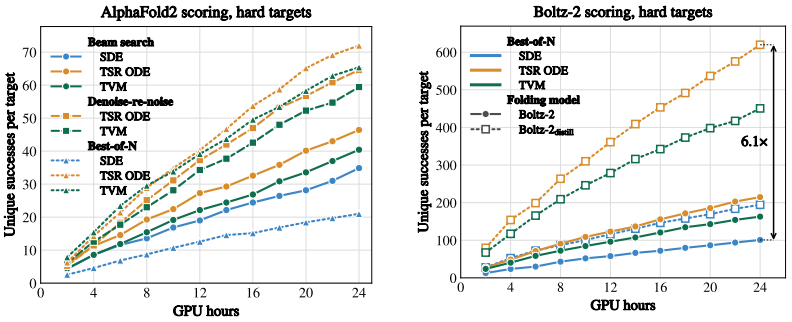

Figure 8: **Inference-time scaling with *Proteina-Pronto*.** *(left)* Using AlphaFold2-Multimer for scoring and different search algorithms. *(right)* Best-of-N with different sampling methods using both Boltz-2 and Boltz-2_{distill} for scoring.

by backpropagating a structural objective through a frozen folding model. We compare Boltz-2 with Boltz-2_{distill} in BoltzDesign's Pairformer+Confidence mode, generating 30 designs per target for five protein, four ligand, and one DNA targets. Boltz-2_{distill} combines the distilled Pairformer with four-step sampling. Replacing Boltz-2 with Boltz-2_{distill} accelerates the full iterative design loop by approximately $3.3\times$ on average. Both models' designs undergo the same evaluation pipeline: we redesign eight sequences per design, refold them, and count the design as successful if at least one passes the success criteria (details in Sec. G.8). Including the GPU cost of evaluation, Boltz-2_{distill} yields 2.1-2.6$\times$ more successful designs per GPU-hour than Boltz-2 (Tab. 1). Fig. 22 visualizes two successful DNA binders by Boltz-2_{distill}.

Table 1: **More successful designs per GPU-hour with Boltz-2_{distill}.** For each target category, we report successful designs per GPU-hour (counting design and evaluation). Both success filters require complex pLDDT > 0.7; the strict and relaxed filters additionally require ipAE < 10 Å and ipAE < 15 Å, respectively.

	Boltz2		Boltz-2_{distill}		Success Ratio	
Targets	strict	relax.	strict	relax.	strict	relax.
Protein (5)	3.7	4.1	**7.9**	**10.8**	2.2×	2.6×
Ligand (4)	5.2	5.3	**12.0**	**12.7**	2.3×	2.4×
DNA (1)	4.6	5.3	**9.5**	**11.8**	2.1×	2.2×

4.4 Accelerated Inference-Time Search

We evaluate inference-time scaling strategies on the 19-target benchmark of Proteina-Complexa, following their easy/hard target split. We compare best-of-N, beam search, and denoise-re-noise search, generating with the Proteina-Complexa SDE sampler (400 steps, following Didi et al. (2026b)), *Proteina-Pronto*$_{TSR}$ (64 steps), or *Proteina-Pronto*$_{TVM}$ (16 steps), and scoring with AlphaFold2-Multimer or Boltz-2. We report unique successful designs per target against GPU-hours. Unlike Fig. 4, which isolates generation, we measure the total time covering generation and folding-based evaluation. Sampling discretizations, search parameters, and the denoise-re-noise scheme are given in Sec. D.

Fig. 8 (left) shows hard targets scored with AlphaFold2-Multimer, where evaluation costs are identical across methods and gains come from cheaper generation. Accelerated generation more than doubles the number of unique successful designs at a fixed budget: the best SDE configuration, beam search, reaches an average of 18 unique successes per target in 12 GPU-hours, whereas *Proteina-Pronto*$_{TSR}$ and *Proteina-Pronto*$_{TVM}$ reach 40 at the same budget. On aggregate, best-of-N with accelerated samplers is the strongest option, although this is not universal, and on some hard targets, such as HER2, accelerated beam search and denoise-re-noise do better. Accelerating evaluation compounds these gains (Fig. 8, right): replacing the 200-step Boltz-2 scorer with our 4-step Boltz-2_{distill}, *Proteina-Pronto*$_{TSR}$ generation yields $6.1\times$ as many unique successful designs per GPU-hour as the SDE sampler scored with regular Boltz-2. Results on easy targets and per-target breakdowns are reported in Sec. G.9.

5 Conclusions

We introduce *Proteina-Pronto*, a framework for accelerating protein generation and structure prediction. Our distilled protein design models achieve performance comparable to the 256-step La-Proteina and Proteina-Complexa SDE samplers in 16 steps, a $16\times$ speedup. We also accelerate Boltz-2 through diffusion and representation distillation while retaining approximately 99% of its lDDT on the PDB test set. Together, the accelerated generators and folding models enable more efficient inference-time scaling, while Boltz-2_{distill} also accelerates hallucination-based protein design. Further reducing sampling steps in protein design models remains challenging. Asymmetric schedules and low-temperature sampling increase curvature and distribute it throughout the teacher's trajectories (Sec. G.5), making these dynamics difficult to approximate with very few student steps. Future work could explore ways to reduce this curvature or develop distillation methods that better account for it.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods (coding). We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, support qualitative and thematic data analysis, interpret results. The following tasks are not applicable to this work: generate synthetic data sets, assist with translation, clean and reformat dataset. Additionally, we used generative AI tools to create or modify scientific figures or images, create or edit software code (coding), edit a research paper to improve readability. We have reviewed all AI-assisted work. We checked the code produced using AI coding agents, and went over all the sections that were edited with LLMs to improve readability. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Protein design models can support the development of new therapeutics and enzymes. Our work accelerates existing protein generation and structure prediction models, making these tools more accessible under limited compute budgets and enabling larger design campaigns. These benefits should be accompanied by responsible use and appropriate biosafety and biosecurity safeguards to address potential misuse.

REPRODUCIBILITY STATEMENT

We provide training details in the appendix, including datasets, computational resources, batch sizes, learning rates, and training step counts, along with sampling settings and evaluation protocols. We also plan to release our code alongside the paper.

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, 2024.

Gustaf Ahdritz, Nazim Bouatta, Christina Floristean, Sachin Kadyan, Qinghui Xia, William Gerecke, Timothy J O'Donnell, Daniel Berenberg, Ian Fisk, Niccolò Zanichelli, et al. Openfold: retraining alphafold2 yields new insights into its learning mechanisms and capacity for generalization. *Nature methods*, 21(8):1514–1524, 2024.

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.

Sankar Basu and Björn Wallner. Dockq: a quality measure for protein-protein docking models. *PloS one*, 11(8):e0161879, 2016.

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):235–42, 2000.

Marco Biasini, Tobias Schmidt, Stefan Bienert, Valerio Mariani, Gabriel Studer, Jürgen Haas, Niklaus Johner, Andreas Daniel Schenk, Ansgar Philippsen, and Torsten Schwede. Openstructure: an integrated software framework for computational structural biology. *Biological crystallography*, 69(5):701–709, 2013.

Nicholas M Boffi, Michael S Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation. *arXiv preprint arXiv:2505.18825*, 2025.

Joey Bose, Tara Akhound-Sadegh, Guillaume Huguet, Kilian Fatras, Jarrid Rector-Brooks, Cheng-Hao Liu, Andrei Cristian Nica, Maksym Korablyov, Michael M. Bronstein, and Alexander Tong. SE(3)-stochastic flow matching for protein backbone generation. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Salvatore Candido, Thomas Hayes, Alexander Derry, Roshan Rao, Zeming Lin, Robert Verkuil, Bryan Z Wu, Jin Sub Lee, Elise S Bruguera, Jehan A Keval, et al. Language modeling materializes a world model of protein biology. *bioRxiv*, pp. 2026–06, 2026.

Longxing Cao, Brian Coventry, Inna Goreshnik, Buwei Huang, William Sheffler, Joon Sung Park, Kevin M Jude, Iva Marković, Rameshwar U Kadam, Koen HG Verschueren, et al. Design of protein-binding proteins from the target structure alone. *Nature*, 605(7910):551–560, 2022.

Hansheng Chen, Kai Zhang, Hao Tan, Leonidas Guibas, Gordon Wetzstein, and Sai Bi. pi-flow: Policy-based few-step generation via imitation distillation. *arXiv preprint arXiv:2510.14974*, 2025a.

Vincent B Chen, Jonathan R Wedell, R Kent Wenger, Eldon L Ulrich, and John L Markley. Molprobity for the masses–of data. *Journal of biomolecular NMR*, 63:77–83, 2015.

X Chen, Y Zhang, C Lu, W Ma, J Guan, C Gong, J Yang, H Zhang, K Zhang, and AML ByteDance. Protenix: Advancing structure prediction through a comprehensive alphafold3 reproduction. *bioRxiv*, 2025b.

Yehlin Cho, Martin Pacesa, Zhidian Zhang, Bruno E Correia, and Sergey Ovchinnikov. Boltzdesign1: Inverting all-atom structure prediction model for generalized biomolecular binder design. *BioRxiv*, pp. 2025–04, 2025.

Alexander E. Chu, Jinho Kim, Lucy Cheng, Gina El Nesr, Minkai Xu, Richard W. Shuai, and Po-Ssu Huang. An all-atom protein generative model. *Proceedings of the National Academy of Sciences*, 121(27):e2311500121, 2024.

Nathaniel Corley, Simon Mathis, Rohith Krishna, Magnus S Bauer, Tuscan R Thompson, Woody Ahern, Maxwell W Kazman, Rafael I Brent, Kieran Didi, Andrew Kubaney, et al. Accelerating biomolecular modeling with atomworks and rf3. *BioRxiv*, 2025.

Justas Dauparas, Ivan Anishchenko, Nathaniel Bennett, Hua Bai, Robert J Ragotte, Lukas F Milles, Basile IM Wicky, Alexis Courbet, Rob J de Haas, Neville Bethel, et al. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56, 2022.

Kieran Didi, Danny Reidenbach, Matthew Penner, Supriya Ravichandran, Marshall Case, Mike Nichols, Erik Swanson, Alex Reis, Maggie Prescott, Yue Qian, Dongming Qian, Jingjing Yang, Weiji Li, Le Li, Daichi Shonai, Sean Gay, Bhoomika Basu Mallik, Ho Yeung Chim, Liurong Chen, Miguel Atienza Juantay, Hubert Klein, Anna U. Macintyre, Maxim Secor, Daniele Granata, Zhonglin Cao, Guoqing Zhou, Tomas Geffner, Xi Chen, Micha Livne, Zuobai Zhang, Tianjing Zhang, Kyle Gion, Michael M. Bronstein, Martin Steinegger, Kristine Deibler, Scott Soderling, Alena Khmelinskaia, Florian Hollfelder, Christian Dallago, Emine Kucukbenli, Arash Vahdat, Pierce Ogden, and Karsten Kreis. Latent generative search unlocks de novo design of untapped biomolecular interactions at scale. *bioRxiv*, 2026a. doi: 10.64898/2026.09.12.751118. Preprint.

Kieran Didi, Zuobai Zhang, Guoqing Zhou, Danny Reidenbach, Zhonglin Cao, Sooyoung Cha, Tomas Geffner, Christian Dallago, Jian Tang, Michael M Bronstein, et al. Scaling atomistic protein binder design with generative pretraining and test-time compute. *The International Conference on Learning Representations (ICLR)*, 2026b.

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.

Tomas Geffner, Kieran Didi, Zuobai Zhang, Danny Reidenbach, Zhonglin Cao, Jason Yim, Mario Geiger, Christian Dallago, Emine Kucukbenli, Arash Vahdat, and Karsten Kreis. Proteina: Scaling flow-based protein structure generative models. In *International Conference on Learning Representations (ICLR)*, 2025.

Tomas Geffner, Kieran Didi, Zhonglin Cao, Danny Reidenbach, Zuobai Zhang, Christian Dallago, Emine Kucukbenli, Karsten Kreis, and Arash Vahdat. La-proteina: Atomistic protein generation via partially latent flow matching. *The International Conference on Learning Representations (ICLR)*, 2026.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

Chengyue Gong, Xinshi Chen, Yuxuan Zhang, Yuxuan Song, Hao Zhou, and Wenzhi Xiao. Protenix-mini: Efficient structure predictor via compact architecture, few-step diffusion and switchable plm. *arXiv preprint arXiv:2507.11839*, 2025.

Tiankai Hang, Shuyang Gu, Chen Li, Jianmin Bao, Dong Chen, Han Hu, Xin Geng, and Baining Guo. Efficient diffusion training via min-snr weighting strategy. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 7441–7451, 2023.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Guillaume Huguet, James Vuckovic, Kilian Fatras, Eric Thibodeau-Laufer, Pablo Lemos, Riashat Islam, Cheng-Hao Liu, Jarrid Rector-Brooks, Tara Akhound-Sadegh, Michael Bronstein, Alexander Tong, and Avishek Joey Bose. Sequence-augmented se(3)-flow matching for conditional protein backbone generation. *arXiv preprint arXiv:2405.20313*, 2024.

John Ingraham, Max Baranov, Zak Costello, Vincent Frappier, Ahmed Ismail, Shan Tie, Wujie Wang, Vincent Xue, Fritz Obermeyer, Andrew Beam, and Gevorg Grigoryan. Illuminating protein space with a programmable generative model. *Nature*, 623:1070–1078, 2023.

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 Zidek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596:583–589, 2021.

Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Foundations of Crystallography*, 32(5):922–923, 1976.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 24174–24184, 2024.

Dongjun Kim, Chieh-Hsin Lai, WeiHsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. In *International Conference on Learning Representations*, volume 2024, pp. 44493–44525, 2024.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Rohith Krishna, Jue Wang, Woody Ahern, Pascal Sturmfels, Preetham Venkatesh, Indrek Kalvet, Gyu Rie Lee, Felix S. Morey-Burrows, Ivan Anishchenko, Ian R. Humphreys, Ryan McHugh, Dionne Vafeados, Xinting Li, George A. Sutherland, Andrew Hitchcock, C. Neil Hunter, Alex Kang, Evans Brackenbrough, Asim K. Bera, Minkyung Baek, Frank DiMaio, and David Baker. Generalized biomolecular modeling and design with rosettafold all-atom. *Science*, 384(6693): eadl2528, 2024. doi: 10.1126/science.adl2528. URL https://www.science.org/doi/abs/10.1126/science.adl2528.

Yeqing Lin and Mohammed Alquraishi. Generating novel, designable, and diverse protein structures by equivariantly diffusing oriented residue clouds. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

Yeqing Lin, Minji Lee, Zhao Zhang, and Mohammed AlQuraishi. Out of many, one: Designing and scaffolding proteins at the scale of the structural universe with genie 2. *arXiv preprint arXiv:2405.15489*, 2024.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Valerio Mariani, Marco Biasini, Alessandro Barbato, and Torsten Schwede. lddt: a local superposition-free score for comparing protein structures and models using distance difference tests. *Bioinformatics*, 29(21):2722–2728, 2013.

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.

Martin Pacesa, Lennart Nickel, Christian Schellhaas, Joseph Schmidt, Ekaterina Pyatova, Lucas Kissling, Patrick Barendse, Jagrity Choudhury, Srajan Kapoor, Ana Alcaraz-Serna, et al. One-shot design of functional protein binders with bindcraft. *Nature*, 646(8084):483–492, 2025.

Saro Passaro, Gabriele Corso, Jeremy Wohlwend, Mateo Reveiz, Stephan Thaler, Vignesh Ram Somnath, Noah Getz, Tally Portnoi, Julien Roy, Hannes Stark, et al. Boltz-2: Towards accurate and efficient binding affinity prediction. *BioRxiv*, 2025.

Bo Qiang, Chengyue Gong, Xinshi Chen, Yuxuan Zhang, and Wenzhi Xiao. Protenix-mini+: efficient structure prediction model with scalable pairformer. *arXiv preprint arXiv:2510.12842*, 2025.

Wei Qu, Jiawei Guan, Rui Ma, Ke Zhai, Weikun Wu, and Haobo Wang. P(all-atom) is unlocking new path for protein design. *bioRxiv*, 2024.

Milong Ren, Tian Zhu, and Haicang Zhang. Carbonnovo: Joint design of protein structure and sequence using a unified energy-based model. In *Forty-first International Conference on Machine Learning*, 2024.

Amirmojtaba Sabour, Michael S Albergo, Carles Domingo-Enrich, Nicholas M Boffi, Sanja Fidler, Karsten Kreis, and Eric Vanden-Eijnden. Test-time scaling of diffusions with flow maps. *arXiv preprint arXiv:2511.22688*, 2025.

Gianluca Scarpellini, Ron Shprints, Peter Holderrieth, Juno Nam, Pranav Murugan, Rafael Gómez-Bombarelli, Tommi Jaakkola, Maruan Al-Shedivat, Nicholas Matthew Boffi, and Avishek Joey Bose. Few-step cofolding with all-atom flow maps. *arXiv preprint arXiv:2606.08375*, 2026.

Neta Shaul, Chao Liu, Arash Vahdat, and Julius Berner. Parallel decoding distillation for fast image and video generation. *arXiv preprint arXiv:2607.26004*, 2026.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. 2023.

Hannes Stark, Felix Faltings, MinGyu Choi, Yuxin Xie, Eunsu Hur, Timothy O'Donnell, Anton Bushuiev, Talip Uçar, Saro Passaro, Weian Mao, et al. Boltzgen: Toward universal binder design. *bioRxiv*, pp. 2025–11, 2025.

Martin Steinegger and Johannes Söding. Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat Biotechnol.*, 35:1026–1028, 2017.

Michel van Kempen, Stephanie S. Kim, Charlotte Tumescheit, Milot Mirdita, Jeongjae Lee, Cameron L. M. Gilchrist, Johannes Söding, and Martin Steinegger. Fast and accurate protein structure search with foldseek. *Nat Biotechnol.*, 42:243–246, 2024.

Mihaly Varadi, Stephen Anyango, Mandar Deshpande, Sreenath Nair, Cindy Natassia, Galabina Yordanova, David Yuan, Oana Stroe, Gemma Wood, Agata Laydon, Augustin Žídek, Tim Green, Kathryn Tunyasuvunakool, Stig Petersen, John Jumper, Ellen Clancy, Richard Green, Ankur Vora, Mira Lutfi, and Sameer Velankar. Alphafold protein structure database: Massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucleic Acids Research*, 50:D439–D444, 2021.

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, Basile I. M. Wicky, Nikita Hanikel, Samuel J. Pellock, Alexis Courbet, William Sheffler, Jue Wang, Preetham Venkatesh, Isaac Sappington, Susana Vázquez Torres, Anna Lauko, Valentin De Bortoli, Emile Mathieu, Regina Barzilay, Tommi S. Jaakkola, Frank DiMaio, Minkyung Baek, and David Baker. De novo design of protein structure and function with rfdiffusion. *Nature*, 620:1089–1100, 2023.

Jeremy Wohlwend, Gabriele Corso, Saro Passaro, Noah Getz, Mateo Reveiz, Ken Leidal, Wojtek Swiderski, Liam Atkinson, Tally Portnoi, Itamar Chinn, et al. Boltz-1 democratizing biomolecular interaction modeling. *BioRxiv*, pp. 2024–11, 2025.

Liyang Xie, Haoran Zhang, Zhendong Wang, Wesley Tansey, and Mingyuan Zhou. Distilled protein backbone generation. *arXiv preprint arXiv:2510.03095*, 2025.

Yanbo Xu, Yu Wu, Sungjae Park, Zhizhuo Zhou, and Shubham Tulsiani. Temporal score rescaling for temperature sampling in diffusion and flow models. *arXiv preprint arXiv:2510.01184*, 2025.

Yilun Xu, Mingyang Deng, Xiang Cheng, Yonglong Tian, Ziming Liu, and Tommi Jaakkola. Restart sampling for improving generative processes. *Advances in Neural Information Processing Systems*, 36:76806–76838, 2023.

Jason Yim, Andrew Campbell, Andrew Y. K. Foong, Michael Gastegger, José Jiménez-Luna, Sarah Lewis, Victor Garcia Satorras, Bastiaan S. Veeling, Regina Barzilay, Tommi Jaakkola, and Frank Noé. Fast protein backbone generation with se(3) flow matching. *arXiv preprint arXiv:2310.05297*, 2023.

Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T Freeman. Improved distribution matching distillation for fast image synthesis. *Advances in neural information processing systems*, 37:47455–47487, 2024a.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6613–6623, 2024b.

Jaehoon Yoo, Wonjung Kim, Floor Eijkelboom, Chanhyuk Lee, Nicholas M Boffi, Seunghoon Hong, and Jinwoo Kim. Self-conditioned flow map language models via fixed-point flows. *arXiv preprint arXiv:2607.00714*, 2026.

Vinicius Zambaldi, David La, Alexander E Chu, Harshnira Patani, Amy E Danson, Tristan OC Kwan, Thomas Frerix, Rosalia G Schneider, David Saxton, Ashok Thillaisundaram, et al. De novo design of high-affinity protein binders with alphaproteo. *arXiv preprint arXiv:2409.08022*, 2024.

Yang Zhang and Jeffrey Skolnick. Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function, and Bioinformatics*, 57(4):702–710, 2004.

Zhe Zhang, Yuanning Feng, Yuxuan Song, Keyue Qiu, Hao Zhou, and Wei-Ying Ma. Dcfold: Efficient protein structure generation with single forward pass. In *The Fourteenth International Conference on Learning Representations*, 2026.

Qinqing Zheng, Matt Le, Neta Shaul, Yaron Lipman, Aditya Grover, and Ricky TQ Chen. Guided flows for generative modeling and decision making. *arXiv preprint arXiv:2311.13443*, 2023.

Mingyuan Zhou, Huangjie Zheng, Zhendong Wang, Mingzhang Yin, and Hai Huang. Score identity distillation: Exponentially fast distillation of pretrained diffusion models for one-step generation. In *Forty-first International Conference on Machine Learning*, 2024.

Appendix

A Details for *Proteina-Pronto* with MeanFlow and DMD **16**
A.1 MeanFlow 16
A.2 Distribution Matching Distillation 17

B La-Proteina Sampling Details **18**

C Self-Conditioning **19**
C.1 Self-Conditioning for La-Proteina and Proteina-Complexa 19
C.2 Adapting Self-Conditioning for Distilled Models 19

D Inference-Time Scaling with Accelerated Models **21**

E Metrics **22**
E.1 Metrics for Generative Design Model 22
E.2 Metrics for Folding Model 23

F Details for Experiments **23**
F.1 Protein Design Generative Models 23
F.1.1 Unconditoinal Monomer Generative Models 23
F.1.2 Binder Design Generative Models 24
F.2 Distilling Boltz2 diffusion module with π-Flow 24
F.2.1 Low-Temperature ODE Teacher 25
F.2.2 Adaption of student architecture 25
F.2.3 Noise scale sampling 26
F.2.4 Loss parameterization and weighting 26
F.2.5 Data and training configurations 27
F.2.6 1- and 2-step generator distillation 27
F.3 Distilling Boltz2 Pairformer with representation distillation 28
F.3.1 Motivation and student architecture 28
F.3.2 Representation distillation loss 29
F.3.3 Training 30

G Additional Results **31**
G.1 Unconditional Monomer Generation 31
G.2 TSR Score-Scaling: Parameters and Quality–Diversity Trade-Off 32
G.3 Extended MolProbity Atomistic Structure Analysis 33
G.4 Extended Rotamer Distribution Analysis 33
G.5 Curvature analysis for *Proteina-Pronto* Coupled Velocity 33
G.6 Inference Time Profiling of Boltz2 34
G.7 Additional evaluation of Boltz-2_{distill} 34
G.8 Hallucination-based binder design with Boltz-2_{distill} 39
G.9 Extended Results for Accelerated Inference-Time Search 40

A Details for *Proteina-Pronto* with MeanFlow and DMD

A.1 MeanFlow

We adapt MeanFlow (Geng et al., 2025) to distill the coupled TSR-ODE (Eq. (9)). As in TVM, the two-time student $\tilde{\mathbf{u}}_\phi = [\tilde{\mathbf{u}}^{\mathbf{x}}_\phi, \tilde{\mathbf{u}}^{\mathbf{z}}_\phi]$ predicts the teacher's average velocity between a start time t and an

end time $r > t$. We use the same schedule-aware parameterization (Eq. (11)), teacher initialization ($\phi \leftarrow \theta$), and annealing of $(k_{\mathbf{y}}, \sigma_{\mathbf{y}})$ over the first 20k training steps, but supervise the student with a different objective.

To derive this objective, consider the exact average velocity $\tilde{\mathbf{u}}$ along a teacher trajectory. For each modality, it satisfies $\mathbf{y}_r = \mathbf{y}_t + (r-t)\tilde{\mathbf{u}}^{\mathbf{y}}$. Advancing the starting state along this trajectory while holding r fixed leaves the endpoint $\mathbf{y}_r$ unchanged. Differentiating therefore gives the MeanFlow identity

$$\tilde{\mathbf{u}}^{\mathbf{y}}(\mathbf{x}_t, \mathbf{z}_t, t, r) = \tilde{\mathbf{v}}_\theta^{\mathbf{y}}(\mathbf{x}_t, \mathbf{z}_t, t; \gamma_{\mathbf{y}}) + (r-t)\tfrac{\mathrm{d}}{\mathrm{d}t}\tilde{\mathbf{u}}^{\mathbf{y}}(\mathbf{x}_t, \mathbf{z}_t, t, r). \tag{13}$$

Here, $\mathrm{d}/\mathrm{d}t$ differentiates both the starting state and its time along the coupled teacher flow, with the endpoint time r held fixed. Applied to the student, this derivative is

$$\frac{\mathrm{d}}{\mathrm{d}t}\tilde{\mathbf{u}}_\phi^{\mathbf{y}} = \partial_t \tilde{\mathbf{u}}_\phi^{\mathbf{y}} + \sum_{\mathbf{y}' \in \{\mathbf{x},\mathbf{z}\}} \left(\tilde{\mathbf{v}}_\theta^{\mathbf{y}'} \cdot \nabla_{\mathbf{y}'}\right) \tilde{\mathbf{u}}_\phi^{\mathbf{y}}. \tag{14}$$

For brevity, student quantities in this expression and below are evaluated at $(\mathbf{x}_t, \mathbf{z}_t, t, r)$, and each teacher velocity $\tilde{\mathbf{v}}_\theta^{\mathbf{y}}$ at $(\mathbf{x}_t, \mathbf{z}_t, t; \gamma_{\mathbf{y}})$. The sum includes both modalities because each student output depends on their joint state. We train the student to satisfy Eq. (13) using the objective

$$\mathcal{L}(\phi) = \mathbb{E}\left[\sum_{\mathbf{y}\in\{\mathbf{x},\mathbf{z}\}} \left\| \tilde{\mathbf{u}}_\phi^{\mathbf{y}} - \underbrace{\left(\tilde{\mathbf{v}}_\theta^{\mathbf{y}} + (r-t)\frac{\mathrm{d}}{\mathrm{d}t}\tilde{\mathbf{u}}_\phi^{\mathbf{y}}\right)}_{\text{stop_grad}} \right\|^2\right]. \tag{15}$$

The total derivative is computed by a single Jacobian-vector product. This derivative includes the time dependence of the schedule prefactor in Eq. (11). The entire regression target is detached, so optimization gradients pass only through the leading student prediction.

Intuitively, MeanFlow encourages the student to predict the same endpoint as the starting state and time advance together along the teacher flow. TVM imposes a complementary condition: as the ending time r changes, the time derivative of the student's predicted endpoint should match the teacher velocity at that endpoint (Eq. (10)). Thus, MeanFlow constrains changes in the starting point, whereas TVM constrains changes in the endpoint.

This distinction also determines where teacher supervision is obtained. TVM evaluates the teacher at the student-predicted endpoint, introducing an on-policy component: the teacher provides feedback at a state reached by the student's own prediction. MeanFlow instead evaluates the teacher at the current training state, without querying it at the predicted endpoint. When training states are drawn directly from the flow-matching interpolant, these teacher queries are independent of the student. However, we also apply the same *self-forcing* self-conditioning procedure as for TVM (Sec. C.2). In this case, the current state is itself produced by a preceding student step, so MeanFlow also receives supervision at student-generated states. We report the experimental comparison in Sec. 4.1.

A.2 Distribution Matching Distillation

Distribution Matching Distillation (DMD) (Yin et al., 2024b;a) aims to match the student's output distribution to a target distribution, without requiring correspondence between individual sampling trajectories. Let p_ϕ denote the student distribution and p_θ the target distribution, with p_ϕ^t and p_θ^t their perturbed versions under the coupled interpolants of Eq. (7). The standard formulation is motivated by the reverse-KL objective

$$\mathcal{L}_{\mathrm{DM}}(\phi) = \mathbb{E}_t\left[D_{\mathrm{KL}}(p_\phi^t \| p_\theta^t)\right]. \tag{16}$$

Its gradient depends on the difference between the scores of the two perturbed distributions. DMD obtains these predictions from a frozen teacher and an auxiliary *fake-score* network, trained to denoise student-generated samples and track the student's changing distribution.

For our low-temperature adaptation, we use teacher predictions obtained by applying TSR to the coupled velocity field (Eq. (9)). We express the update through the corresponding clean-sample predictions, with inverse-error normalization. The fake-score network uses the same velocity-to-clean conversion but receives no additional TSR scaling, since it is trained directly on the student's outputs (which mimics the low temperature teacher). Training alternates between student updates and multiple denoising updates of the fake-score network.

Table 2: Unconditional generation with *Proteina-Pronto$_{DMD}$*, trained for 20k steps with paired regression ($\lambda_{\text{reg}} = 0.25$). We evaluate 500 proteins per sampling budget. Secondary-structure statistics are computed over all-atom co-designable samples. Metric definitions follow Tab. 5.

	Co-des (%)		Div. (co-des)		Sec. str. (%)		Backbone metrics	
Steps	C_α	All-atom	Str.	Seq.	α	β	Des (%)	Div. (des)
32	65.4	58.2	122	201	90	0	99.6	245
16	47.6	38.8	101	156	88	0	96.6	308
8	31.2	21.0	70	94	85	0	86.2	318
4	14.2	9.8	30	46	84	0	67.6	268

Additional objectives. Reverse-KL-based distribution matching can suffer from mode dropping and loss of diversity in practice. To address the collapse observed in our experiments, we follow Yin et al. (2024b) and add paired regression. We precompute pairs of Gaussian noise and clean samples obtained by integrating the teacher TSR-ODE. The student is trained to reproduce each teacher sample in a single evaluation from the same initial noise, using squared-error regression with $\lambda_{\text{reg}} = 0.25$. We also test replacing Cartesian regression on the C_α coordinates with a smooth lDDT loss (Mariani et al., 2013; Abramson et al., 2024), which penalizes errors in local pairwise distances.

Motivated by DMD2 (Yin et al., 2024a), we additionally test an adversarial objective. A lightweight classifier operates on intermediate features from the frozen teacher to distinguish student-generated samples from precomputed teacher samples, after perturbing both at matched noise levels. The student receives an adversarial loss encouraging its samples to be classified as teacher samples.

Empirical observations. We were unable to obtain competitive DMD students in the configurations tested. Distribution matching alone exhibits severe mode collapse: in an inspected batch of 20 samples, all proteins were near-identical, elongated single α-helices. Paired regression replaces these helical rods with more compact, globular proteins, but does not prevent secondary-structure collapse. Even with the paired regression, DMD exhibits severe secondary-structure collapse: α-helical content ranges from 84-90%, and none of these samples contain β-sheets at any of the tested sampling budgets (Tab. 2).

Generated samples also exhibit a systematic error in local backbone geometry. Mean distances between consecutive C_α atoms range from 3.49 to 3.63 Å across 4-32 sampling steps, below the expected spacing of approximately 3.8 Å. Increasing the sampling budget does not resolve this error: at 32 steps, the mean distance is 3.49 Å. Neither adding the adversarial objective nor replacing Cartesian regression with smooth lDDT resolves the combined secondary-structure and local-geometry failures.

Tab. 2 reports results for DMD with paired Cartesian regression after 20k training steps. Compared with the other distillation methods at matching sampling budgets, DMD yields lower co-designability and co-designable structural and sequence diversity (Tab. 5). At eight steps, it produces 70 co-designable structural clusters, compared with 176 for MeanFlow, 202 for TVM, and 207 for π-Flow. Thus, although increasing the sampling budget improves co-designability, substantial deficiencies in diversity and local geometry remain. Unlike trajectory distillation, which operates directly on a specified low-temperature ODE, DMD uses score-based distributional supervision; we leave a more systematic treatment of this low-temperature setting to future work.

B La-Proteina Sampling Details

As mentioned in Sec. 2, La-Proteina trains a velocity network $\mathbf{v}(\mathbf{x}_{t_\mathbf{x}}, \mathbf{z}_{t_\mathbf{z}}, t_\mathbf{x}, t_\mathbf{z})$ to denoise alpha-carbon coordinates and latent variables for any pair $(t_\mathbf{x}, t_\mathbf{z})$. Building on the fact that the score of the intermediate noisy densities can be obtained as a linear transformation of the flow matching velocity, La-Proteina generates samples by simulating the system of SDEs given by

$$\begin{aligned} \mathrm{d}\mathbf{x}_{t_\mathbf{x}} &= \mathbf{v}^{\mathbf{x}}_\theta \,\mathrm{d}t_\mathbf{x} + \beta_\mathbf{x}(t_\mathbf{x})\zeta^{\mathbf{x}}_\theta \,\mathrm{d}t_\mathbf{x} + \sqrt{2\beta_\mathbf{x}(t_\mathbf{x})\eta_\mathbf{x}}\,\mathrm{d}\mathcal{W}_{t_\mathbf{x}} \\ \mathrm{d}\mathbf{z}_{t_\mathbf{z}} &= \mathbf{v}^{\mathbf{z}}_\theta \,\mathrm{d}t_\mathbf{z} + \beta_\mathbf{z}(t_\mathbf{z})\zeta^{\mathbf{z}}_\theta \,\mathrm{d}t_\mathbf{z} + \sqrt{2\beta_\mathbf{z}(t_\mathbf{z})\eta_\mathbf{z}}\,\mathrm{d}\mathcal{W}_{t_\mathbf{z}}, \end{aligned} \tag{17}$$

from $(t_\mathbf{x}, t_\mathbf{z}) = (0, 0)$ to $(t_\mathbf{x}, t_\mathbf{z}) = (1, 1)$. While in practice this integration could be done for any monotonic trajectory, Geffner et al. (2026) observed that using schedules that denoise backbones

faster than the latent variables was critical for good performance. Specifically, given $t \in [0, 1]$, they found that the "exponential" and "quadratic" schedules, given by the scheduling functions

$$f_{\mathbf{x}}(t) = \frac{1 - 10^{-2t}}{1 - 10^{-2}} \qquad \text{and} \qquad f_{\mathbf{z}}(t) = t^2, \tag{18}$$

performed best. The corresponding discretizations for each modality are then obtained by partitioning the unit interval uniformly and evaluating the corresponding scheduling functions at those points.

Finally, for high-quality generations, Geffner et al. (2026) observed that low temperature sampling is required, and found that the parameters $(\eta_{\mathbf{x}}, \eta_{\mathbf{z}}) = (0.1, 0.1)$ led to good performance.

Coupling functions used by *Proteina-Pronto*. Following the choice of the sampling scheme from Eq. (18), we use the coupling functions $\alpha_{\mathbf{x}}(t) = f_{\mathbf{x}}(t)$ and $\alpha_{\mathbf{z}}(t) = f_{\mathbf{z}}(t)$ for *Proteina-Pronto*. This effectively ties the modalities in a path that denoises alpha-carbon coordinates faster than the latents, following the precise rates from Eq. (18).

C Self-Conditioning

This section ablates self-conditioning for the teacher models La-Proteina and Proteina-Complexa, and introduces *self-forcing* self-conditioning for distilled models.

C.1 Self-Conditioning for La-Proteina and Proteina-Complexa

We examine the importance of self-conditioning by enabling or disabling it during sampling from the pretrained La-Proteina and Proteina-Complexa models, using both the original SDE sampler and TSR-ODE sampling. For unconditional monomer generation, self-conditioning increases the number of co-designable structural clusters across all tested sampling budgets for both samplers (Fig. 9). It also generally improves all-atom co-designability, although this effect depends on the sampler and sampling budget. The dependence on self-conditioning is substantially stronger for binder design (Fig. 10): disabling it sharply reduces both the total number of successful samples and the number of structurally distinct successful designs, while also reducing target coverage. For example, at 64 steps, disabling self-conditioning reduces the number of solved targets from 15 to 9 for TSR and from 12 to 4 for SDE. Increasing the sampling budget does not eliminate these deficits, with substantial gaps remaining at 256 steps. These results show that the benefits of self-conditioning persist when replacing stochastic SDE sampling with deterministic TSR sampling, and that preserving this capability is particularly important when accelerating binder design.

C.2 Adapting Self-Conditioning for Distilled Models

Self-conditioning in flow matching. Self-conditioning supplies the model with its own clean-sample prediction as an additional input. For training examples that use self-conditioning, a preliminary forward pass is performed on the noisy input without self-conditioning. Its predicted velocity is converted into a clean-sample estimate, which conditions a second forward pass on the same noisy input and noise level. The training objective is applied to this second prediction. For the rectified interpolants used by La-Proteina and Proteina-Complexa, the clean-sample estimate for modality $\mathbf{y}$ is

$$\hat{\mathbf{y}}_1 = \mathbf{y}_{t_{\mathbf{y}}} + (1 - t_{\mathbf{y}})\, \mathbf{v}^{\mathbf{y}}_\theta(\mathbf{x}_{t_{\mathbf{x}}}, \mathbf{z}_{t_{\mathbf{z}}}, t_{\mathbf{x}}, t_{\mathbf{z}}), \qquad \mathbf{y} \in \{\mathbf{x}, \mathbf{z}\}. \tag{19}$$

At inference, an additional forward pass is avoided by reusing the clean-sample prediction from the preceding sampling step. The first sampling step proceeds without a previous prediction.

Naive adaptation to distilled models. The same procedure can be applied to a distilled student: during training, we evaluate it twice on the same state and time interval, using the first call's clean-sample estimate to condition the second. For TVM and MeanFlow, we obtain this estimate by extrapolating the predicted average velocity to the clean endpoint. Accounting for the modality schedules

$$\hat{\mathbf{y}}_1 = \mathbf{y}_t + \big(1 - \alpha_{\mathbf{y}}(t)\big)\, \mathbf{u}^{\mathbf{y}}_\phi(\mathbf{x}_t, \mathbf{z}_t, t, r), \qquad \mathbf{y} \in \{\mathbf{x}, \mathbf{z}\}. \tag{20}$$

Here, $\mathbf{u}^{\mathbf{y}}_\phi$ is the velocity normalized by the modality's schedule displacement, rather than the global-time velocity $\tilde{\mathbf{u}}^{\mathbf{y}}_\phi$. The resulting clean-sample prediction extrapolates the estimated motion over the

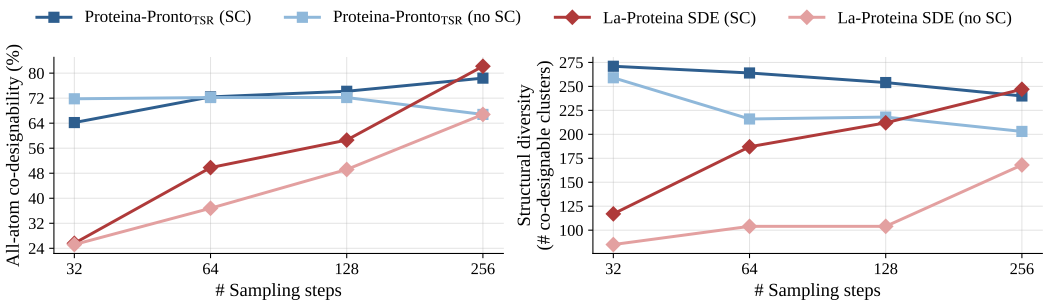

Figure 9: Self-conditioning ablation for La-Proteina SDE and *Proteina-Pronto$_{TSR}$* for unconditional monomer generation. Same evaluation pipeline as used in the main text (Sec. 4.)

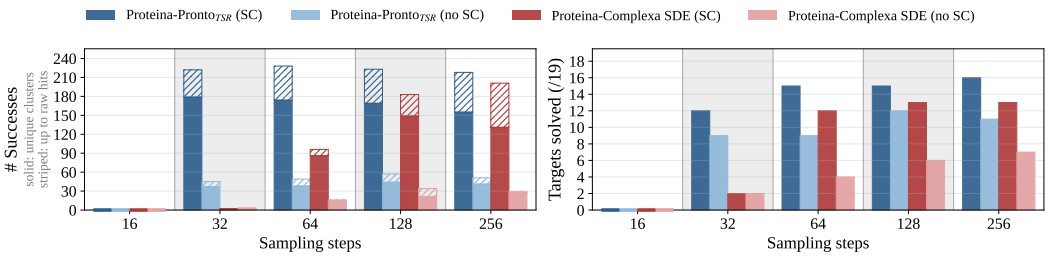

Figure 10: Self-conditioning ablation for Proteina-Complexa SDE and *Proteina-Pronto$_{TSR}$* for binder design. Same targets and evaluation pipeline as used in the main text (Sec. 4.)

interval. During inference, we instead reuse the clean-sample estimate from the preceding student step. This introduces a mismatch: training uses a conditioning prediction obtained at the current noise level, whereas inference uses one obtained at an earlier, noisier state. We hypothesize that this mismatch becomes more consequential for distilled models because their larger sampling steps separate consecutive predictions by substantially different noise levels.

***Self-forcing* self-conditioning.** To better align training with inference, we construct the conditioning input through a preceding student step during training. For a transition from t to r, we use the preceding interval $[\max\{0,\, t-(r-t)\},\, t]$. This interval has the same global-time duration as the current transition unless its starting time is clipped to zero, in which case it is shorter. We construct its starting state from the coupled flow-matching interpolant in Eq. (7), then evaluate the student to advance this state to t. This call provides both a predicted state at t and a clean-sample estimate obtained using Eq. (20). We use the predicted state as the input to the current transition from t to r, and the clean-sample estimate as its self-conditioning input. We then apply the corresponding distillation objective to this current transition. At $t = 0$, no preceding interval is available, so the current call proceeds without self-conditioning, as at the start of sampling. Thus, both the current state and its conditioning prediction come from the same preceding student step, as they do during inference. This addresses the mismatch in how self-conditioning inputs are constructed, although the preceding prediction may still be less informative when sampling steps are large.

For π-Flow, we apply the same construction to consecutive sampling segments. We predict and integrate the policy for the preceding segment, using its endpoint as the starting state of the current segment. The velocity at the preceding policy's final knot is converted into a clean-sample estimate, accounting for the modality schedules, and used to condition the current policy prediction. The naive variant instead obtains its conditioning prediction through an additional policy evaluation on the same segment.

Ablation. We compare TVM students trained with naive and *self-forcing* self-conditioning on the 19-target binder-design benchmark. We focus on 16-step sampling, where the distinction between the two training procedures is particularly relevant to accelerated generation. Both students use self-conditioning at inference, reusing the preceding step's clean-sample prediction. The student trained with *self-forcing* solves 14 of the 19 targets and produces 180 unique successful designs,

compared with 8 targets and 60 unique successes for naive self-conditioning. *self-forcing* produces more unique sucesses and improves task coverage in this regime.

Related work on self-conditioning and flow-map distillation. Recent work by Yoo et al. (2026) also studies self-conditioning in the context of flow-map distillation, in the context of language models They interpreting repeated self-conditioned denoising at a fixed noisy state and time as iterations toward a fixed point. Their two-stage approach first distills these iterations into a single forward pass, producing a denoiser that approximates the converged prediction without carrying self-conditioning between sampling steps. The resulting flow model is then distilled into a few-step flow map. We explored a similar approach for our protein models, but were unable to preserve the teacher's generation quality in the first stage: compressing multiple self-conditioned teacher evaluations at the same noisy state into a single forward pass proved challenging in our experiments. We therefore retain self-conditioning in our distilled students and train them using the *self-forcing* scheme described above.

D Inference-Time Scaling with Accelerated Models

Didi et al. (2026b) demonstrated the benefits of inference-time scaling for binder design through best-of-N sampling and guided search. These strategies allocate additional compute to generating and evaluating candidates. We use *Proteina-Pronto* to reduce their sampling costs, targeting more unique successful designs within a given compute budget.

Sampling configurations. We compare the original Proteina-Complexa SDE sampler with *Proteina-Pronto$_{TSR}$* and *Proteina-Pronto$_{TVM}$*, using 400, 64, and 16 steps, respectively, for a full sampling trajectory. Throughout this section, the shared time t increases from noise ($t = 0$) to clean data ($t = 1$). Candidate structures are decoded and scored using the folding model selected for each experiment: AlphaFold2-Multimer, the original 200-step Boltz-2, or the four-step Boltz-2$_{\text{distill}}$.

Best-of-N sampling. We generate independent candidates from independent Gaussian initial states and rank them using folding-based rewards. Using *Proteina-Pronto$_{TSR}$* or *Proteina-Pronto$_{TVM}$* directly accelerates candidate generation, without requiring intermediate search or selection.

Beam search with accelerated look-ahead. Beam search allocates compute to promising partially denoised candidates. We use a beam width of four, a branching factor of four, and four search rounds. At each round, *trajectory propagation* advances four copies of each retained candidate using independent SDE noise to obtain distinct continuations. *Look-ahead rollouts* complete and decode these continuations for folding-based reward evaluation. The four highest-scoring intermediate states are retained for the next round. Propagation always uses the 400-step SDE sampler. For look-ahead, we compare the same SDE sampler with *Proteina-Pronto$_{TSR}$* and *Proteina-Pronto$_{TVM}$*, following Sabour et al. (2025). This reduces rollout costs while preserving stochastic branching.

Search checkpoints occur at $t = 0.25, 0.5, 0.75, 1$. Propagation and look-ahead use separate grids, uniform in the shared time and mapped through the same per-modality schedules. These checkpoints align exactly with indices $(100, 200, 300, 400)$ on the SDE grid, $(16, 32, 48, 64)$ on the TSR grid, and $(4, 8, 12, 16)$ on the TVM grid. Each look-ahead rollout uses only the remaining portion of its sampler's grid.

Denoise-re-noise search. To accelerate trajectory propagation as well, we must preserve the stochasticity that allows copies of the same candidate to diverge. We therefore adopt denoise-re-noise sampling (Xu et al., 2023; Kim et al., 2024), using either *Proteina-Pronto$_{TSR}$* or *Proteina-Pronto$_{TVM}$* for both propagation and look-ahead. We retain the same beam width, branching factor, and number of search rounds as beam search. For each non-terminal round starting at time t, we first denoise deterministically to a cleaner state at s, then re-noise to the next search time r, where $t < r < s \leq 1$. We set

$$s = t + \tfrac{1}{2}(1 - t), \qquad r = t + \tfrac{1}{2}(s - t) = t + \tfrac{1}{4}(1 - t). \tag{21}$$

Independent re-noising noise for each copy produces distinct partially denoised candidates. Accelerated look-ahead rollouts complete and score these candidates, and the four highest-scoring states at r are retained. Starting from $t = 0$, the successive retained times are 0.25, 0.4375, and 0.578125. In the fourth and final round, we denoise directly to $t = 1$ without re-noising. Thus, the two guided-search strategies use the same branching settings but different search times.

Self-conditioning. TVM uses recurrent self-conditioning: each sampling step receives the preceding step's clean-sample prediction, without an additional forward pass to recompute it. After re-noising, we retain the clean-sample prediction from the preceding deterministic jump unchanged. The subsequent step therefore conditions on a prediction made before the re-noising operation.

Computational cost. Generation timings in Fig. 4 include propagation and look-ahead computation, but exclude folding-based reward evaluation. Campaign costs include both generation and folding-based evaluation and are reported in A100 GPU-hours. We compare strategies by the number of unique successful designs obtained within these budgets in Fig. 8 and Sec. G.9.

E METRICS

E.1 METRICS FOR GENERATIVE DESIGN MODEL

We evaluate generative models through their designability, all-atom co-designability, and diversity.

- **Designability.** This metric assesses whether a generated backbone is physically plausible and capable of being realized by an amino acid sequence. Following the standard protocol in the field, we use ProteinMPNN (Dauparas et al., 2022) with temperature 0.1 to design 8 candidate sequences for each generated backbone. Each sequence is then folded using ESMFold (Lin et al., 2023) to produce a predicted structure. A backbone is considered designable if the minimum C_α-RMSD between the generated coordinates and any of the 8 ESMFold predictions is < 2.0 Å. This serves as a validation generated structures independently of the generated sequence.
- **All-atom co-designability.** Unlike designability, which uses an external sequence designer, co-designability evaluates the consistency of the joint sequence-structure generation. We directly fold the sequence generated by the models using ESMFold. A sample is successful only if the all-atom RMSD between this folded structure and the model-generated structure is ≤ 2.0 Å. High co-designability indicates that the model has successfully captured the relationship between latent variables and 3D structure.
- **Structural diversity.** We cluster the subset of designable or all co-designable samples at a 0.5 TM-score threshold using FoldSeek (van Kempen et al., 2024), and report the total number of unique clusters.
- **Sequence diversity.** We extract the amino-acid sequence from each of the same subset of designable or all co-designable samples and cluster them with MMseqs2 (Steinegger & Söding, 2017) at a 10% minimum sequence identity and 70% coverage threshold (target coverage mode), reporting the total number of unique clusters.

For the the generated atomistic structure evaluation, we use the MolProbity (MP) score and the Clash score.

- **Clash Score.** The Clash Score quantifies the number of serious steric overlaps (typically defined as an atomic overlap > 0.4 Å) per 1,000 atoms within a molecular model. This evaluation relies on the explicit computational placement of hydrogen atoms to accurately assess all-atom contacts. A lower Clash Score indicates fewer unphysical atomic collisions, reflecting superior local packing and stereochemical realism.
- **MolProbity Score.** The MolProbity Score is an empirical, global metric that aggregates three distinct evaluations: the Clash Score, the percentage of unfavored Ramachandran backbone conformations, and the percentage of poor side-chain rotamers. These are combined into a single log-weighted value normalized to reflect the expected crystallographic resolution (in Å) at which the model's observed geometric quality would be considered typical. A lower MolProbity Score signifies a more physically realistic, high-quality structural prediction.

For binder design tasks, we follow the success criteria from Didi et al. (2026b). We evaluate designs by refolding each binder sequence together with its target using AlphaFold2-Multimer (Jumper et al., 2021; Evans et al., 2021). A design is considered successful if it satisfies all three criteria: inter-chain

predicted aligned error (ipAE) < 7 Å, complex pLDDT > 90, and binder self-consistency RMSD (scRMSD) < 1.5 Å.

- **Inter-chain predicted aligned error (ipAE).** Measures confidence in the relative positioning of the binder and target, with lower values indicating greater confidence in the predicted interface, produced by the folding model.
- **Complex pLDDT.** Folding model confidence score in the local structure across the binder-target complex, with higher values indicating more confident structural predictions.
- **Binder self-consistency RMSD (scRMSD).** Measures RMSD between the generated binder and its refolded structure (from the generated sequence).

E.2 Metrics for Folding Model

We evaluate folding model with the metrics described below, following Boltz' evaluation pipeline (Wohlwend et al., 2025; Passaro et al., 2025). The evaluation pipeline is built upon OpenStructure (Biasini et al., 2013). Stereochemistry check is enabled during the lDDT calculation for more rigorous local structure evaluation.

- **Local Distance Difference Test (lDDT).** lDDT (Mariani et al., 2013) is a superposition-free metric that evaluates the local structural accuracy of a predicted model against a reference structure. lDDT effectively assesses local backbone and side-chain accuracy without the confounding penalties of global alignment.
- **DockQ.** A continuous quality metric, ranging from 0 to 1, developed to evaluate the accuracy of multimeric and protein-protein docking interfaces. DockQ (Basu & Wallner, 2016) is a combination of three classic structural criteria: the fraction of native interfacial contacts (F_{nat}), the interface Root Mean Square Deviation (iRMSD), and the ligand Root Mean Square Deviation (LRMSD). Higher DockQ score generally indicates better predicted interfaces. Scores ≥ 0.23 denote an acceptable model, while scores ≥ 0.49 indicate medium to high accuracy.
- **Protein-Ligand Interaction lDDT (lDDT-PLI).** A specialized adaptation of the lDDT metric tailored specifically for protein-small molecule interfaces. lDDT-PLI measures the fraction of native intermolecular contacts (distances between ligand heavy atoms and surrounding protein atoms) that are correctly preserved in the predicted pose within specified tolerance thresholds.
- **Template Modeling Score (TM-Score).** TM-Score (Zhang & Skolnick, 2004) assesses the global topological similarity of two protein structures. The metric is length-independent and scales from 0 to 1, with a score >0.5 strictly representing a correctly predicted global fold.
- **L-RMSD<2Å.** Ligand Root Mean Square Deviation (L-RMSD) measures the average distance between superimposed atoms of a predicted ligand structure and its reference. The success rate of predictions achieving an L-RMSD<2.0Å serves as a standard rigorous binary benchmark for success.

F Details for Experiments

F.1 Protein Design Generative Models

As mentioned in the main text, *Proteina-Pronto$_{TSR}$* requires no re-training, as it only involves a change of the sampling algorithm at inference time.

F.1.1 Unconditoinal Monomer Generative Models

Training The distilled *Proteina-Pronto* models (π-Flow, TVM, MeanFlow, and DMD) for unconditional monomer generation were trained on the subset of the AFDB (Varadi et al., 2021) consisting of proteins between 32 and 256 residues with an average plddt ≥ 80. All these models were initialized from the La-Proteina teacher (Geffner et al., 2026), using the Adam optimizer (Kingma, 2014) with a learning rate of 10^{-5}. All models were trained to distill *Proteina-Pronto$_{TSR}$* with $(k_{\mathbf{x}}, \sigma_{\mathbf{x}}, k_{\mathbf{z}}, \sigma_{\mathbf{z}}) = (12, 2, 3, 3)$.

- *Proteina-Pronto$_{TVM}$*: Trained for 95k steps with Adam (Kingma, 2014) on 24 A100-80GB NVIDIA GPUs, with a batch size of 6 per GPU, and a learning rate of 10^{-5}. The model was trained with *self-forcing* self-conditioning in half of the training steps (following the usual choice in flow matching and diffusion models).
- *Proteina-Pronto$_{MeanFlow}$*: Trained for 130k steps with Adam on 24 A100-80GB NVIDIA GPUs, with a batch size of 6 per GPU, and a learning rate of 10^{-5}. The model was trained with *self-forcing* self-conditioning in half of the training steps.
- *Proteina-Pronto$_{\pi\text{-}Flow}$*: We trained separate students for 1, 2, 4, and 8 sampling steps, using the Adam optimizer, with approximately 750k training updates per model on average. The final training configuration used 32 A100-80GB NVIDIA GPUs with a batch size of 16 per GPU. The policy uses 64 grid points for the 1-, 2-, and 4-step students and 32 for the 8-step student. Training uses four teacher queries, with times sampled uniformly from $[0.001, 0.999]$. We integrate the predicted velocity policies using 256 Euler substeps during both training and inference.
- *Proteina-Pronto$_{DMD}$*: Trained for 40k steps with Adaom, using alternating optimization, with four fake-score network updates for each student update. We used 24 A100-80GB NVIDIA GPUs, and a batch size of 16 per GPU. The reported model combines distribution matching with paired regression weighted by $\lambda_{\text{reg}} = 0.25$. The objectives and additional experiments with adversarial and smooth lDDT losses are described in Sec. A.2.

Sampling Following Geffner et al. (2026) we fix the SDE noise-scaling factors to $(0.1, 0.1)$. For *Proteina-Pronto$_{TSR}$* we use $(k_{\mathbf{x}}, \sigma_{\mathbf{x}}, k_{\mathbf{z}}, \sigma_{\mathbf{z}}) = (12, 2, 3, 3)$, which, as explained above, is also used for the distillation teacher. We explore other TSR scaling parameters and show results for them in Sec. G.2. Both La-Proteina with the SDE sampler and *Proteina-Pronto$_{TSR}$* use self-conditioning, which improves co-designability and diversity (ablation in Sec. C.1).

F.1.2 Binder Design Generative Models

Training The distilled model *Proteina-Pronto$_{TVM}$* for binder design was trained on the same data as Proteina-Complexa (Didi et al., 2026b), an 80:20 mixture of Teddymer (510,454 cluster representatives) and PDB multimers (45,856 experimental complexes). (Teddymer comprises synthetic binder–target pairs obtained by splitting predicted multidomain proteins into interacting domains, followed by structural clustering and interface-quality filtering.) We trained on 32 A100-80GB NVIDIA GPUs with an effective batch size of 2 per GPU (batch size of 1 with gradient accumulation set to two). We set the learning rate to 2×10^{-5} for the first 80k steps, a cosine decay to 2×10^{-6} between steps 80k and 130k, and 2×10^{-6} after that. We trained for 150k steps. This model was trained to distill *Proteina-Pronto$_{TSR}$* with $(k_{\mathbf{x}}, \sigma_{\mathbf{x}}, k_{\mathbf{z}}, \sigma_{\mathbf{z}}) = (12, 3, 4, 8)$.

Sampling Following Didi et al. (2026b) we fix the SDE noise-scaling factors to $(0.1, 0.1)$. For *Proteina-Pronto$_{TSR}$* we use $(k_{\mathbf{x}}, \sigma_{\mathbf{x}}, k_{\mathbf{z}}, \sigma_{\mathbf{z}}) = (12, 3, 4, 8)$, which, as explained above, are also used for the distillation teacher. Both La-Proteina with the SDE sampler and *Proteina-Pronto$_{TSR}$* use self-conditioning, which improves performance (ablation in Sec. C.1).

F.2 Distilling Boltz2 diffusion module with π-Flow

Boltz2 predicts structures with low-temperature sampling of its diffusion module. Its sampler multiplies every denoising update by a constant step scale parameter larger than one (1.5 by default), such that each step moves the structure further towards the clean-data prediction of the diffusion module than the probability flow ODE or SDE prescribes. Low-temperature sampling is the default of Boltz2 and noticeably improves the prediction accuracy: on the PDB test set, it raises the lDDT from 0.800 with a step scale of 1.0 to 0.820. Therefore, our goal is to distill a few-step student from a low-temperature teacher rather than from the regular-temperature one for better accuracy. This goal motivates the choice of the teacher (Sec. F.2.1) and of the loss weighting (Sec. F.2.4), which keep the distillation training of the low-temperature teacher stable.

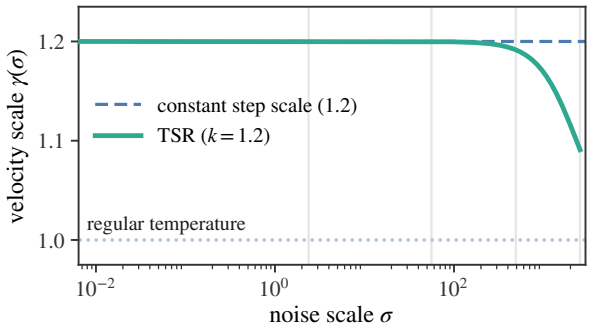

Figure 11: **Velocity scale of the low-temperature teacher** as a function of the noise scale, for TSR ($k = 1.2$, $\sigma_{\mathrm{TSR}} = 2560$) and for a constant step scale. Vertical lines mark the segment starts of the 4-step student.

F.2.1 Low-Temperature ODE Teacher

From SDE to ODE. The default sampler of Boltz2 is an EDM-style stochastic sampler (Abramson et al., 2024; Karras et al., 2022; Wohlwend et al., 2025). Trajectory distillation methods such as π-Flow imitate the deterministic trajectories of an ODE and cannot be applied to such a stochastic sampler. Therefore, we first switch to the probability flow ODE of the EDM formulation as the teacher, as also proposed by ProteinX-mini (Gong et al., 2025). We use $\mathbf{X}_t \in \mathbb{R}^{A\times 3}$ to denote the coordinates of a biomolecular complex with A atoms at diffusion time t and noise scale σ_t (Eq. (24)), and D_θ to denote the clean-data prediction of Boltz2's diffusion module.

Low temperature via TSR. Applying a constant step scale[1] to the ODE velocity, as the Boltz2 sampler does by default, made distillation unstable in our experiments. The student is initialized from the teacher, so its initial clean-data prediction D_θ differs from the low-temperature target from Eq. (23) by $(\gamma(\sigma_t) - 1)(\mathbf{X}_t - D_\theta)$, with the magnitude of this difference growing linearly with the noise scale. With a constant step scale and the variance-exploding noise schedule of EDM, this residual produces large losses at large noise scales and gradient norms an order of magnitude larger than for the regular-temperature teacher. We instead rescale the velocity with TSR (Xu et al., 2025),

$$\frac{\mathrm{d}\mathbf{X}_t}{\mathrm{d}\sigma_t} = \gamma(\sigma_t)\,\frac{\mathbf{X}_t - D_\theta(\mathbf{X}_t;\sigma_t)}{\sigma_t}, \qquad \gamma(\sigma) = \frac{\sigma_{\mathrm{TSR}}^2/\sigma^2 + 1}{\sigma_{\mathrm{TSR}}^2/(k\,\sigma^2) + 1}, \tag{22}$$

which is Eq. (5) with $\xi(\sigma) = 1/\sigma^2$ and $(k, \sigma_{\mathrm{TSR}})$ in place of $(k_{\mathbf{y}}, \sigma_{\mathbf{y}})$. Unlike a constant step scale, γ approaches one for $\sigma \gg \sigma_{\mathrm{TSR}}$, where the noisy distribution is dominated by the Gaussian noise irrespective of the temperature, and approaches k for $\sigma \ll \sigma_{\mathrm{TSR}}$. We use $k = 1.2$ and set $\sigma_{\mathrm{TSR}} = 2560$, the maximum noise scale, which reduces the low-temperature scaling in the noise-dominant region while keeping it at 1.2 elsewhere (see visualization of vector field scaling in Fig. 11). With the 200-step stochastic sampler, this rescaling reaches the lDDT of the default step scale of 1.5 (0.822 vs. 0.820). In clean-data space, the rescaled velocity corresponds to the regression target of the student,

$$\tilde{D}_\theta(\mathbf{X}_t;\sigma_t) = \big(1 - \gamma(\sigma_t)\big)\,\mathbf{X}_t + \gamma(\sigma_t)\,D_\theta(\mathbf{X}_t;\sigma_t), \tag{23}$$

so the student learns the low-temperature ODE directly and no rescaling is applied at its inference time.

Temperature annealing. Similar to Sec. 3.1.3, we linearly anneal k from 1 to 1.2 over the first 5k training steps, so that the target changes gradually from the regular-temperature ODE that the student is initialized with. We found this to stabilize the early phase of training.

F.2.2 Adaption of Student Architecture

We finetune the Boltz2 diffusion module to be a dynamic-$\hat{\mathbf{X}}_1^{(t)}$ policy (DX policy) predictor using the π-Flow framework. DX policy requires the student model to output a grid of clean data predictions,

[1]The step scale of the sampler is usually denoted by η in the folding model literature (Abramson et al., 2024). We denote it by $\gamma(\sigma)$ since it becomes a function of the noise scale and follows the TSR notation.

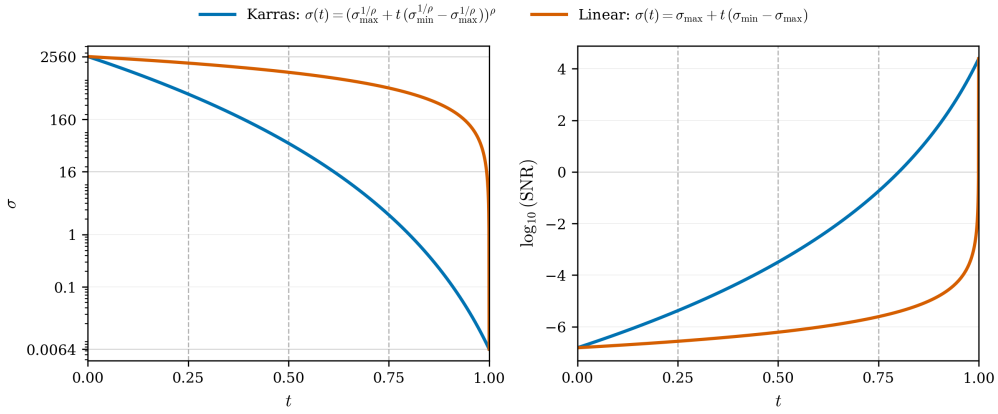

Figure 12: Visualization of different mappings of noise scale (σ_t) or signal-to-noise (SNR) ratio and time.

$\hat{\mathbf{X}}_1^{(t)}$, at N evenly spaced times $t_1, \ldots, t_N \in [t_{\text{src}}, t_{\text{dst}}]$, where t_{src} and t_{dst} are the source and destination time of a segment of the ODE trajectory. Given that the Boltz2 diffusion module is a variant of the EDM (Karras et al., 2022) model that directly predicts the clean data, we only need minimal change to convert the diffusion module into a DX policy predictor by changing the final linear layer output dimension from $[A, 3]$ to $[N \times A, 3]$. Student weights are initialized from teacher, and the expanded final layer replicates that of the teacher N times. The policy velocity with respect to the noise scale is $(\mathbf{X}_t - \hat{\mathbf{X}}_1^{\pi}(t))/\sigma_t$, where $\hat{\mathbf{X}}_1^{\pi}(t)$ linearly interpolates the grid and σ_t is the interpolated noise scale of each grid cell.

F.2.3 Noise scale sampling

In Boltz2, the diffusion module follows the EDM formulation to build the interpolant between clean data ($\mathbf{X}_1$) and noise by:

$$\mathbf{X}_t = \mathbf{X}_1 + \sigma_t \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{24}$$

where

$$\sigma_t = \left(\sigma_{\max}^{1/\rho} + t\left(\sigma_{\min}^{1/\rho} - \sigma_{\max}^{1/\rho}\right)\right)^{\rho} \tag{25}$$

The training configuration sets $\sigma_{\min} = 0.0004$ and $\sigma_{\max} = 160$ and further scales them by $\sigma_{\text{data}} = 16$. Therefore the effective noise scale ranges from $\sigma \in [\sigma_{\min}, \sigma_{\max}] = [0.0064, 2560]$. In Boltz2 EDM setting, $\rho = 7$.

During DX policy π-Flow distillation, the segment starting point t_{src} is sampled uniformly from the $M = 4$ segment boundaries and the rollout intermediate time t uniformly within the segment, and both are mapped to noise scale σ_t using Eq. (25) (Karras mapping) as input to the model. In our experiments, the Karras mapping prevents catastrophic lDDT collapse associated with other mapping choice such as linear mapping $\sigma_t = \sigma_{\max} + t(\sigma_{\min} - \sigma_{\max})$ because it focuses the training much more on medium to low noise region (Fig. 12).

F.2.4 Loss parameterization and weighting

Given that the output of our student policy network and the teacher network are both the clean data $\mathbf{X}_1$, the policy is parameterized to output in $\mathbf{X}_1$-space and converted to velocity during inference. The loss is also computed directly in the $\mathbf{X}_1$-space instead of the velocity space as originally proposed in (Chen et al., 2025a). Weighting the loss based on the noise scale is found to be essential during π-Flow distillation of Boltz2 diffusion module. With the modifications above, the loss function used in the π-Flow policy-based imitation distillation is:

$$\mathcal{L}(\phi, \psi) = \frac{\lambda(\sigma_t)}{e^{W_\psi(\sigma_t)}} \left\| \hat{\mathbf{X}}_1^{\pi} - \text{sg}\left[\mathcal{A}(\tilde{D}_\theta)\right] \right\|^2 + W_\psi(\sigma_t) \tag{26}$$

where $\lambda(\sigma)$ is a non-learnable weighting based on noise scale, $\hat{\mathbf{X}}_1^{\pi}$ is the clean data prediction of the student policy at $\mathbf{X}_t$, W_ψ is a learnable uncertainty function used to apply additional weighting, $\mathcal{A}$ is the Kabsch alignment (Kabsch, 1976) operation that rigidly aligns the target onto the student

prediction, and sg denotes stop-gradient. The implementation of W_ψ follows exactly the learnable loss weighting MLP in the EDM2 (Karras et al., 2024) model.

Three options of $\lambda(\sigma)$ are tested, including:

- **Uniform:** $\lambda(\sigma) = 1$ is a uniform weighting of all noise scale.
- **EDM:** $\lambda(\sigma) = \frac{\sigma^2+\sigma_{\text{data}}^2}{(\sigma\cdot\sigma_{\text{data}})^2}$. The weighting used in EDM.
- **Min-SNR:** $\lambda(\sigma) = \min(\frac{\sigma_{\text{data}}^2}{\sigma^2}, \lambda_{\max})$. The weighting proposed by Hang et al. (2023) (γ in their notation).

We find that the direct application of EDM weighting in π-Flow distillation leads to collapse of lDDT at early stage of training, as it grows as $1/\sigma^2$ at low noise. Both uniform and Min-SNR weighting stabilize training, but Min-SNR leads to better lDDT (Tab. 3). The learnable loss weighting further improves the lDDT on top of the Min-SNR, thus becoming the final choices. We note that this ablation result comes from our early π-Flow distillation experiments, which use the regular-temperature teacher (step scale 1.0). Its absolute values are therefore lower than those of the final low-temperature model reported in Sec. 4. With the low-temperature teacher, $\lambda_{\max}$ between 5 and 50 performs similarly, while $\lambda_{\max} = 100$ diverged. Our final checkpoint is trained using $\lambda_{\max} = 20$. The rigid alignment $\mathcal{A}$ uses the weighted Kabsch algorithm. It does not change the final lDDT, but prevents a transient lDDT drop early in training and speeds up convergence.

Table 3: lDDT of 4-step π-Flow students with different loss weighting on the PDB test set. lDDT values are from checkpoints in early experiments with the regular-temperature teacher, evaluated at checkpoints of ∼20k steps.

Loss weighting	lDDT↑
Uniform	0.731
Min-SNR	0.785
Min-SNR+W_ψ	0.792

F.2.5 Data and training configurations

We train the π-Flow student with a mixture of the PDB and AFDB (Varadi et al., 2021) with a sampling weight of 0.55 and 0.45, respectively. The sampling and clustering process of the training data follows the original Boltz2 exactly (Passaro et al., 2025). We train only on the 384-token/3,456-atom crop size, which is equivalent to the first training stage of Boltz2.

We train the π-Flow student with 32 NVIDIA H100 GPUs for 60k steps. The effective batch size is 32 and the diffusion head multiplicity is 16. The nucleotide is upweighted $5\times$ and ligand $10\times$ in loss calculation and rigid alignment, following Boltz2 training setting. The number of grid $N = 80$ for the DX policy. In π-Flow training, the number of intermediate states is 2 and total number of substeps is 128. We follow the π-Flow paper to use the micro-window loss with window size of 3. We use the Muon optimizer (Jordan et al., 2024), a 10^{-4} learning rate with 1k steps linear warm-up, halved every 15k steps after 30k steps, no weight decay, and gradient clipping at a norm of 1,000. We evaluate the exponential moving average (EMA) of the weights with a decay of 0.999. At inference, we use 128 substeps and, following Chen et al. (2025a), shorten the last of the 4 segments to half the length of the others.

F.2.6 1- and 2-step generator distillation

The policy network of the most accurate Boltz-2_{distill} uses 4 sampling steps. To explore even lower sampling costs, we also distill 1- and 2-step generators from the same low-temperature teacher, with π-Flow and with Parallel Decoding Distillation (PDD) (Shaul et al., 2026). For π-Flow, we train one model per number-of-step with the recipe above, changing only the number of segments and clipping gradients at a norm of 50 to further stabilize the training.

PDD is closely related to π-Flow. Its student also replicates the final layer of the teacher to predict a grid of outputs from a single network evaluation, integrates these outputs without further network evaluations, and is trained by on-policy imitation, i.e., by matching the teacher at states obtained by rolling out the detached student. The main difference is that PDD discretizes the whole trajectory into a fixed grid of small intervals and predicts one output per interval, supervised with the teacher's ODE step over that interval, instead of a policy over a segment of fixed length. A single network evaluation covers a block of consecutive intervals. Training with blocks of varying length allows the number of steps to be chosen at inference. We implement PDD within our π-Flow setup: the grid has

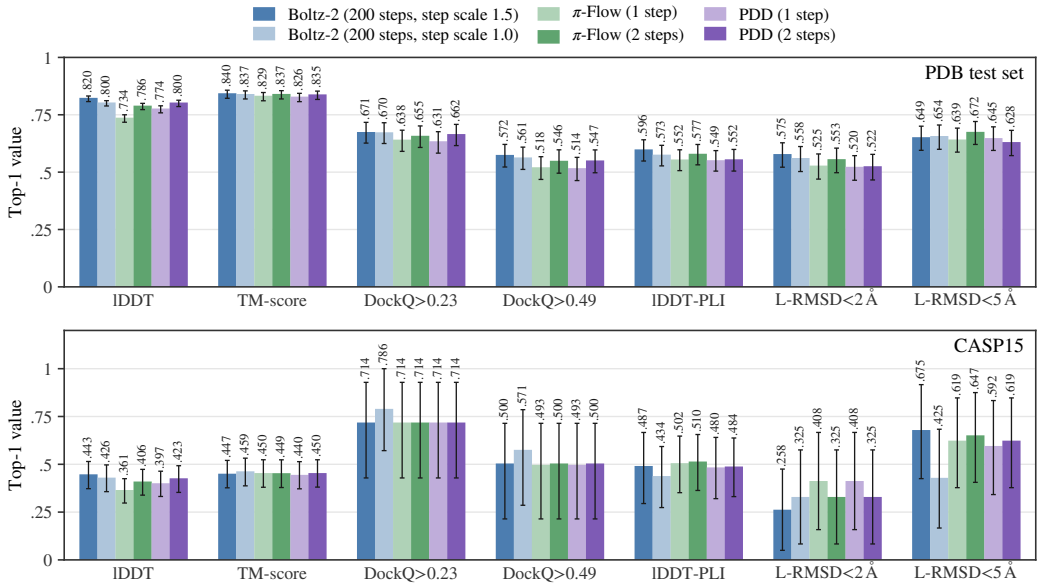

Figure 13: **1- and 2-step π-Flow and PDD students** with our distilled trunk, compared with the 200-step Boltz2 sampler at step scale 1.5 and 1.0 on the PDB test set and CASP15. Top-1 of 5 samples by confidence module; error bars show 95% bootstrap confidence intervals.

256 intervals on the Karras schedule (Eq. (25)), the student predicts the clean coordinates of every interval as for the DX policy, the rollout within a block recomputes the velocity from the current state at every interval, and each target is a single 2nd-order Heun step of the TSR-rescaled teacher. For M sampling steps, we use a model trained with a maximum block length of $256/M$ intervals, $\lambda_{\max} = 50$ and gradient clipping at a norm of 100, and keep all other settings the same as π-Flow.

Fig. 13 compares the 1- and 2-step students of both methods, combined with our distilled trunk, to the 200-step Boltz2 sampler. Both methods reach a similar lDDT on the PDB test set (0.811 for π-Flow, i.e., Boltz-2_{distill}, and 0.810 for PDD) at 4-steps, but π-Flow surpass PDD on some other metrics like DockQ and TM-score at 4-step, making π-Flow our default model. However, with fewer steps, PDD suffers noticeably less accuracy degradation in lDDT: it reaches 0.800 and 0.774 lDDT with 2 and 1 steps, compared with 0.786 and 0.734 for π-Flow. The 2-step PDD model thus matches the 200-step Boltz2 sampler at regular temperature (0.800 lDDT). On CASP15, PDD likewise reaches 0.423 and 0.397 lDDT with 2 and 1 steps respectively, compared with 0.406 and 0.361 for π-Flow.

F.3 Distilling Boltz2 Pairformer with representation distillation

F.3.1 Motivation and student architecture

Removing triangle attention from the Pairformer is a known path to a more efficient trunk: Ouyang-Zhang et al. (2026) showed that a Boltz1-based model without triangle attention in Pairformer (called PairMixer), trained from scratch, matches the accuracy of the original model at a substantially lower cost. More recent folding model such as ESMFold2 (Candido et al., 2026) also trained without triangle attention. Triangle attention is also the most expensive component of the Boltz- Pairformer. In a profiling session in which the teacher and our student trunk were run on the same 531 samples randomly selected from the PDB test set and CASP15, removing it from all 64 blocks reduces the time of a Pairformer pass by 59% on average, from 1.29 s to 0.53 s. Its share in pairformer inference also grows with complex size, from 41% below 256 tokens to 62% above 1,024 tokens (Fig. 14). Therefore, we choose to remove triangle attention rather than search over trunk architectures, and focus on how to obtain such a trunk from the trained Boltz2 model using representation distillation instead of training it from scratch.

The student removes the two triangle attention layers from each of the 64 Pairformer blocks and keeps all other layers untouched. The student can thus directly replace the teacher trunk and be initialized from the teacher weights: each student block is initialized from the corresponding teacher

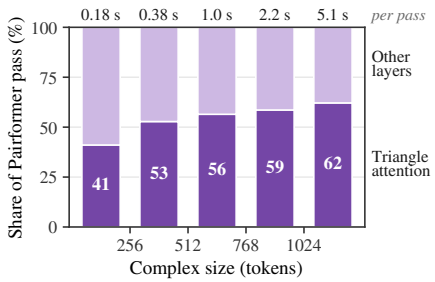

Figure 14: **Share of a single pass of Boltz2 Pairformer inference spent in triangle attention** by complex size. Numbers on top: mean time of a teacher pass. n represents number of samples in that complex size bin.

block, skipping the removed triangle attention layers. The MSA module and the layers that embed the representations of the previous recycling iteration are copied from the teacher. The MSA module keeps its triangle attention and is frozen during training, since it accounts for a small fraction of the inference time (Fig. 20); only the Pairformer and the recycling layers are trained.

F.3.2 Representation distillation loss

We denote the sequence and pair representations by s and z, and mark the student and the teacher with the subscripts $\mathcal{S}$ and $\mathcal{T}$. Boltz2 runs its trunk for C iterations, where each iteration receives the sequence and pair representations of the previous one as additional input. We use $C = 4$ iterations (3 recycle steps after 1 initial pass) for the student, both in training and inference. We detach the representations passed between iterations, such that the loss of each iteration is only backpropagated through that iteration. Let $\mathsf{s}^{(c)}_{\mathcal{S},i} \in \mathbb{R}^{384}$ and $\mathsf{z}^{(c)}_{\mathcal{S},ij} \in \mathbb{R}^{128}$ be the sequence representation of token i and the pair representation of tokens i and j produced by the student after iteration c, and $\mathsf{s}_{\mathcal{T},i}$ and $\mathsf{z}_{\mathcal{T},ij}$ those of the frozen teacher after $C + 1 = 5$ iterations. All student iterations are aligned to the same teacher representations, which teaches the student to refine its own recycled outputs. Running the teacher for one more iteration than the student provides a target slightly ahead of the final student iteration, while remaining affordable to compute at every training step.

The representation distillation loss is $\mathcal{L}_{\mathrm{RD}} = \lambda_{\mathrm{align}}\mathcal{L}_{\mathrm{align}} + \lambda_{\mathrm{KD}}\mathcal{L}_{\mathrm{KD}}$ (Eqs. (29) and (30)). We detail both parts below and list all hyperparameters in Tab. 4. The diffusion module is not used during representation distillation training.

Alignment loss. For a student feature $\mathbf{a} \in \mathbb{R}^d$ of a token or token pair and the corresponding teacher feature $\mathbf{b}$, we measure the discrepancy with

$$\delta(\mathbf{a}, \mathbf{b}) = \tfrac{1}{d}\|\mathbf{a} - \mathbf{b}\|^2 + \lambda_{\cos}\big(1 - \cos(\mathbf{a}, \mathbf{b})\big), \tag{27}$$

where the mean squared error constrains the scale of the features and the cosine distance their direction, and $\lambda_{\cos} = 1000$ brings both terms to a similar magnitude at initialization. The alignment losses of iteration c average δ over tokens and token pairs with modality weights (Zhang et al., 2026),

$$\ell^{(c)}_{\mathsf{s}} = \frac{\sum_i \omega_i\, \delta\big(\mathsf{s}^{(c)}_{\mathcal{S},i}, \mathsf{s}_{\mathcal{T},i}\big)}{\sum_i \omega_i}, \qquad \ell^{(c)}_{\mathsf{z}} = \frac{\sum_{i,j} \sqrt{\omega_i \omega_j}\, \delta\big(\mathsf{z}^{(c)}_{\mathcal{S},ij}, \mathsf{z}_{\mathcal{T},ij}\big)}{\sum_{i,j} \sqrt{\omega_i \omega_j}}, \tag{28}$$

such that the final representation alignment loss is

$$\mathcal{L}_{\mathrm{align}} = \sum_{c=1}^{C} \nu_c \big(\lambda_{\mathsf{z}}\, \ell^{(c)}_{\mathsf{z}} + \lambda_{\mathsf{s}}\, \ell^{(c)}_{\mathsf{s}}\big). \tag{29}$$

The token weights $\omega_i = 1 + 5 \cdot \mathbb{I}[i \in \text{nucleic acid}] + 10 \cdot \mathbb{I}[i \in \text{ligand}]$ for valid tokens and $\omega_i = 0$ for padding follow the atom weights of the diffusion loss of AlphaFold3 (Abramson et al., 2024) and Boltz2. This weighting prevents the numerous protein tokens from dominating the loss and upweight the relatively rare ligand and nucleic acid tokens in the dataset. The iteration weights $\nu_c \propto (0.4, 0.8, 1.0, 1.5)$ are normalized to sum to one, so that the later iterations (more recycles) receive larger weight. We use $\lambda_{\mathsf{z}} = 2$ and $\lambda_{\mathsf{s}} = 1$, to ensure pair representation quality after distillation.

Distogram-based knowledge distillation (KD) loss. The frozen distogram head $\mathcal{H}$ of the teacher is a classifier that maps a pair representation to a distribution over 64 distance bins between 2 and 22 Å for every token pair. Let $\mathbf{p}_{\mathcal{S}} = \mathcal{H}\big(\mathsf{z}_{\mathcal{S}}^{(C)}\big)$ and $\mathbf{p}_{\mathcal{T}} = \mathcal{H}(\mathsf{z}_{\mathcal{T}})$ denote the distributions predicted from the student and teacher pair representations, and $\mathbf{p}^{\mathrm{gt}}$ the distribution of the observed distances over the same bins. The KD loss is

$$\mathcal{L}_{\mathrm{KD}} = \mathrm{CE}\big((1-\lambda_{\mathrm{gt}})\,\mathbf{p}_{\mathcal{T}} + \lambda_{\mathrm{gt}}\,\mathbf{p}^{\mathrm{gt}},\ \mathbf{p}_{\mathcal{S}}\big), \tag{30}$$

where the cross-entropy is computed for every token pair and averaged over all valid token pairs, excluding the diagonal. This is the sum of soft-target knowledge distillation (Hinton et al., 2015), weighted by $1-\lambda_{\mathrm{gt}} = 0.9$, and a light ground-truth term, weighted by $\lambda_{\mathrm{gt}} = 0.1$. We use $\lambda_{\mathrm{align}} = 1$ and $\lambda_{\mathrm{KD}} = 80$, which makes the KD loss $\sim$10% of the total loss. Since $\mathcal{H}$ is frozen and $\mathbf{p}_{\mathcal{T}}$ is fixed, the KD loss only trains the final pair representation of the student, and its teacher part is minimized when the student reproduces the teacher's distogram. The distogram-based KD loss serves as an auxiliary loss to provide further supervision on the pair representation.

Table 4: **Hyperparameters of the representation distillation loss** $\mathcal{L}_{\mathrm{RD}} = \lambda_{\mathrm{align}}\mathcal{L}_{\mathrm{align}} + \lambda_{\mathrm{KD}}\mathcal{L}_{\mathrm{KD}}$ (Eqs. (28) and (30)). The same values are used in both training stages.

Symbol	Description	Value
Recycle		
C	iterations of the student, in training and inference	4 (3 recycle steps)
–	iterations of the teacher target	$C+1=5$
Alignment loss $\mathcal{L}_{\mathrm{align}}$		
ν_c	iteration weights, normalized to sum to one	$\propto (0.4, 0.8, 1.0, 1.5)$
λ_{z}, λ_{s}	weights of the pair and sequence representations	2, 1
$\lambda_{\cos}$	weight of the cosine distance in δ	1000
ω_i	token weights of protein, nucleic acid and ligand	1, 6, 11
KD loss $\mathcal{L}_{\mathrm{KD}}$		
λ_{gt}	fraction of the ground-truth distribution in the target	0.1
Total loss $\mathcal{L}_{\mathrm{RD}}$		
λ_{align}, λ_{KD}	weights of $\mathcal{L}_{\mathrm{align}}$ and $\mathcal{L}_{\mathrm{KD}}$	1, 80

F.3.3 TRAINING

Data and optimization. We compute the teacher representations on-the-fly at every training step with the frozen teacher trunk, which provides fresh crops and MSA subsamples with up to 8,192 sequences at every step instead of a fixed cache. We use a data mixture containing PDB and AFDB with sampling weights of 0.55 and 0.45, respectively. We use 384-token/3,456-atom crop size, pocket and contact conditioning with probability 0.15 each, and single-sequence inputs with probability 0.05 to retain the prediction capability under single-sequence mode. We train with 64 NVIDIA H100 GPUs, one crop per GPU, bf16 mixed precision. We use the cuEquivariance kernel for the remaining triangle multiplication layers and triangle attention layers in the MSA module. We use Muon (Jordan et al., 2024) for the two-dimensional weight matrices of the hidden layers, and AdamW (Loshchilov & Hutter, 2017) for all other parameters. The learning rate is warmed up linearly to 10^{-3} over 1k steps, held constant until 20k steps, and decayed by a factor of 0.95 every 2.5k steps thereafter, with a weight decay of 0.003 and gradient clipping at a norm of 10. We evaluate the EMA of the weights with a decay of 0.999.

Learning rates of the two tracks. Removing triangle attention only changes the pair track of the Pairformer. The sequence representation updating track layers, i.e., the attention with pair bias, its layer normalization and the sequence transition, keep their architecture and are exact copies of the teacher layers at initialization. However, the pair sequence representation updating track (triangle multiplication and pair transition) has to compensate for the removed layers. We therefore train the sequence-track layers with a $0.1\times$ learning rate than the pair-track and the recycle layers. With the full learning rate, we observe early overfitting of sequence representation during representation distillation training.

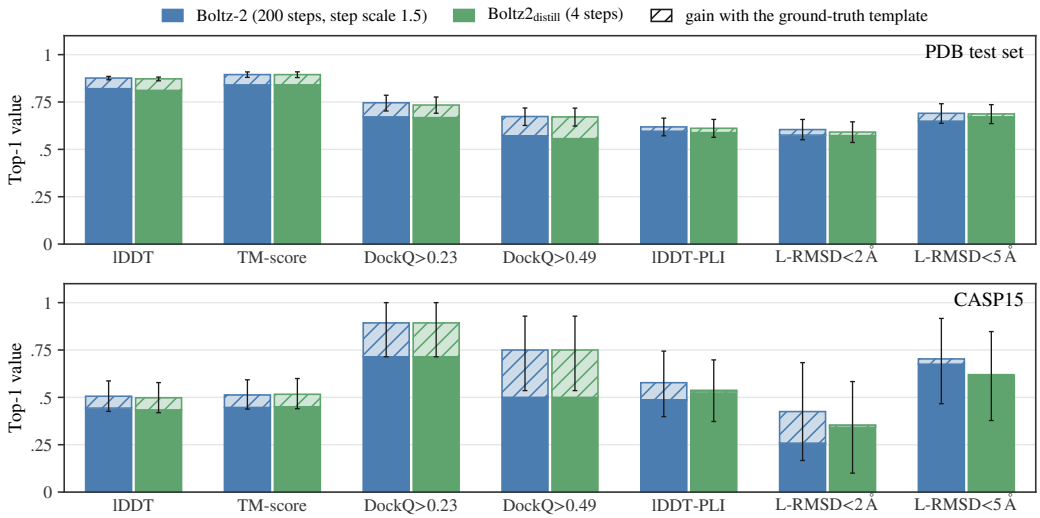

Figure 15: **Template following.** Solid bars show the accuracy without templates, and striped extensions the gain when the experimental structure of each target is provided as template, for the 200-step Boltz2 sampler at step scale 1.5 and Boltz-2_{distill}. Top-1 of 5 samples by confidence; error bars show 95% bootstrap confidence intervals with templates.

Template-following finetuning. Boltz2 can condition on structural templates embedded by its template module and added to the pair representation at every recycling iteration. Such function is critical for the Boltz2 model to be used as a reward model for protein generative model (Didi et al., 2026b). Since we distill without templates first, the student trunk initially never processes template features. We therefore train in two stages with the same objective. In the first stage, we distill without templates and take the EMA weights after 85k steps. In the second stage, we continue from these weights for another 64k steps with templates enabled. The teacher and the student both receive the templates through the frozen template module of the teacher, and the recycling layers and the Pairformer of the student are finetuned to process the template information. We drop the templates of each protein chain with probability 0.5, so that the student retains its accuracy without templates, and use PDB and AFDB sampling weights of 0.7 and 0.3. In this stage, the learning rate is 3×10^{-4}, held constant for 10k steps and decayed by a factor of 0.9 every 2.5k steps.

We evaluate template following by providing the experimental structure of each sample as its template, an oracle setting that measures whether a model makes use of the template information (Fig. 15). The ground-truth template raises the lDDT of Boltz2 on the PDB test set from 0.820 to 0.876 (+0.056), and that of Boltz-2_{distill} from 0.811 to 0.872 (+0.061). We see similar gains in TM-score and DockQ and on CASP15 (+0.063 and +0.064 lDDT). The π-Flow student was only trained on template-free representations, but can directly use the template-informed representations of the distilled trunk without further training.

Combining the distilled components. We replace the Pairformer and the recycling layers of Boltz2 with those of the distilled trunk, and its diffusion module with the π-Flow student, to get the final Boltz-2_{distill}. The other module including MSA, template, input embedder, diffusion conditioning, distogram, and confidence modules are frozen during training so they remain unchanged. Although the π-Flow student is trained on teacher representations, using it with the student representations only reduces the lDDT on the PDB test set from 0.814 to 0.811, while the percentage of DockQ > 0.49 decreases from 58.0% to 55.7% (Sec. G.7).

G Additional Results

G.1 Unconditional Monomer Generation

Tab. 5 shows full results for unconditional monomer design, using La-Proteina with the SDE sampler, *Proteina-Pronto*$_{TSR}$, and *Proteina-Pronto* distilled with TVM, MeanFlow and π-Flow. The table

Table 5: *Proteina-Pronto* performance for unconditional atomistic design. La-Proteina SDE (Geffner et al., 2026) and *Proteina-Pronto$_{TSR}$* rows below 16 steps are omitted, as both yield zero co-designability and designability at these budgets. Secondary-structure content is computed over all-atom co-designable samples, except for Xie et al. (2025), which generates backbones only and is evaluated over designable samples. Results for Xie et al. (2025) were extracted from their paper, which uses the same evaluation pipeline. Timing uses batches of 10 proteins of 150 residues and includes sampling and final decoding. We report the median batch runtime over 10 runs, divided by 10 to obtain seconds per sample. NA indicates unavailable timings.

# Steps	Method	Co-des (%)		Div. (co-des)		Sec. Str. %		Backbone metrics		Time
		C_α	All-atom	Str.	Seq.	α	β	Des (%)	Div. (des)	$\frac{\text{sec}}{\text{sample}}$
256	*Proteina-Pronto$_{TVM}$*	78.6	75.6	230	302	59	15	94.4	309	0.90
	Proteina-Pronto$_{MeanFlow}$	81.8	77.2	242	331	62	12	96.6	324	0.90
	Proteina-Pronto$_{TSR}$	81.6	78.4	240	325	61	13	96.2	305	0.99
	La-Proteina SDE	85.6	82.2	247	311	63	9	98.8	305	1.01
128	*Proteina-Pronto$_{TVM}$*	77.0	73.8	246	308	61	13	96.4	336	0.45
	Proteina-Pronto$_{MeanFlow}$	80.6	77.4	265	332	61	13	96.2	349	0.45
	Proteina-Pronto$_{TSR}$	77.4	74.2	254	324	61	13	97.2	357	0.50
	La-Proteina SDE	65.8	58.6	212	238	54	8	84.0	298	0.50
64	*Proteina-Pronto$_{TVM}$*	79.0	75.0	258	328	63	12	95.4	344	0.23
	Proteina-Pronto$_{MeanFlow}$	80.6	77.8	260	328	63	12	98.8	336	0.23
	Proteina-Pronto$_{TSR}$	76.4	72.4	264	328	63	11	97.8	366	0.25
	La-Proteina SDE	54.8	49.8	187	221	69	6	81.4	330	0.25
32	*Proteina-Pronto$_{TVM}$*	83.4	79.8	271	335	70	8	98.4	340	0.11
	Proteina-Pronto$_{MeanFlow}$	77.0	73.0	275	322	64	11	99.4	385	0.11
	Proteina-Pronto$_{TSR}$	70.2	64.2	271	302	62	9	98.0	424	0.13
	La-Proteina SDE	31.8	25.6	117	124	72	3	71.0	335	0.13
16	*Proteina-Pronto$_{TVM}$*	81.4	77.4	246	356	68	10	99.2	316	0.059
	Proteina-Pronto$_{MeanFlow}$	76.2	69.4	219	311	69	10	96.8	324	0.058
	Proteina-Pronto$_{TSR}$	0.0	0.0	–	–	–	–	0.0	–	0.065
	La-Proteina SDE	0.0	0.0	–	–	–	–	0.0	–	0.065
8	*Proteina-Pronto$_{\pi\text{-}Flow}$*	72.2	65.0	207	260	69	10	95.0	313	0.048
	Proteina-Pronto$_{TVM}$	75.4	68.8	229	292	76	5	98.8	347	0.031
	Proteina-Pronto$_{MeanFlow}$	67.4	59.8	213	267	74	5	97.6	358	0.031
4	*Proteina-Pronto$_{\pi\text{-}Flow}$*	60.6	48.4	143	194	72	9	95.2	301	0.032
	Proteina-Pronto$_{TVM}$	47.0	34.0	114	158	81	2	97.8	353	0.017
	Proteina-Pronto$_{MeanFlow}$	31.2	17.2	73	84	72	8	79.6	333	0.017
2	*Proteina-Pronto$_{\pi\text{-}Flow}$*	29.8	17.4	40	66	74	8	85.0	269	0.024
	Proteina-Pronto$_{TVM}$	12.0	7.6	25	38	82	1	54.8	207	0.0100
	Proteina-Pronto$_{MeanFlow}$	0.4	0.0	–	–	–	–	0.2	1	0.0099
1	*Proteina-Pronto$_{\pi\text{-}Flow}$*	14.4	6.8	21	36	75	3	77.4	293	0.020
	Proteina-Pronto$_{TVM}$	0.0	0.0	–	–	–	–	0.2	1	0.0065
	Proteina-Pronto$_{MeanFlow}$	0.0	0.0	–	–	–	–	0.0	–	0.0064
Backbone-only baseline										
16	Xie et al. (2025)	–	–	–	–	63	5	94.4	286	NA
8	Xie et al. (2025)	–	–	–	–	67	1	75.6	292	NA
5	Xie et al. (2025)	–	–	–	–	72	0	47.4	151	NA
1	Xie et al. (2025)	–	–	–	–	75	0	0.4	2	NA

shows designability, C_α co-designability, all-atom co-designability, structural and sequence diversity, and secondary structure content. All metrics are described in detail in Sec. E.

G.2 TSR Score-Scaling: Parameters and Quality–Diversity Trade-Off

We study how the Temporal Score Rescaling (TSR) parameters affect generation quality by sampling *Proteina-Pronto$_{TSR}$* across a range of TSR parameters and sampling budgets. TSR rescales the velocity of each modality as shown in Eq. (4) by a per-modality factor governed by two parameters, k and σ. These parameters are set independently for the backbone C_α, denoted $\mathbf{x}$ (with $k_\mathbf{x}, \sigma_\mathbf{x}$), and the latents, denoted $\mathbf{y}$ (with $k_\mathbf{y}, \sigma_\mathbf{y}$). We sweep the parameters $k_\mathbf{x} \in \{1, 6, 12\}$, $\sigma_\mathbf{x} \in \{2, 3\}$, $k_\mathbf{y} \in \{3, 6\}$ and $\sigma_\mathbf{y} \in \{3, 6\}$, Each combination is evaluated at three sampling budgets, 32, 64 and 128 steps. For every configuration we generate 100 samples for length in $\{50, 100, 150, 200, 250\}$.

Table 6: *Proteina-Pronto$_{TSR}$* performance across TSR scaling parameters and sampling budgets. TSR params are the score-scaling k and σ per channel, where $\mathbf{x}$ denotes the backbone (C_α) channel and $\mathbf{y}$ the local-latent channel. Secondary-structure content is over all-atom co-designable samples.

# Steps	TSR				Co-des (%)		Div. (co-des)		Sec. Str. %		Backbone metrics	
	$k_\mathbf{x}$	$\sigma_\mathbf{x}$	$k_\mathbf{y}$	$\sigma_\mathbf{y}$	C_α	All-atom	Str.	Seq.	α	β	Des (%)	Div. (des)
128	12	2	3	3	77.4	74.2	254	324	61	13	97.2	357
	12	2	3	6	78.4	76.2	264	292	61	12	97.4	352
	12	2	6	3	82.2	78.4	267	313	61	13	96.6	343
	12	2	6	6	80.0	77.4	263	274	61	13	97.6	341
	12	3	3	3	89.2	87.0	221	354	67	9	98.6	257
	12	3	6	3	87.6	84.6	238	297	67	9	99.2	290
	12	3	6	6	88.6	86.4	232	265	65	10	98.6	275
	6	2	3	3	78.2	72.8	268	332	59	13	95.8	358
	6	2	3	6	75.2	72.0	263	284	58	14	94.6	351
	6	2	6	3	77.0	71.6	274	301	59	14	96.0	361
	6	3	6	3	85.0	81.0	253	307	63	11	98.4	318
	1	2	3	3	9.8	5.0	21	25	29	29	19.8	88
64	12	2	3	3	76.4	72.4	264	328	63	11	97.8	366
	12	2	6	3	82.6	77.4	286	318	62	12	97.4	369
	12	3	3	6	86.4	83.8	256	281	69	8	99.4	299
	12	3	6	3	86.2	83.4	249	298	67	9	98.6	288
	12	3	6	6	89.4	86.6	251	261	67	9	99.6	301
	6	3	3	3	81.0	77.0	260	344	66	9	99.0	339
	6	3	3	6	85.8	81.8	264	283	67	9	98.4	322
	6	3	6	3	83.8	80.2	251	307	66	9	97.8	311
	6	3	6	6	84.8	81.2	261	264	66	9	99.0	329
	1	2	3	3	8.4	5.6	20	28	30	24	19.8	89
32	12	2	3	3	70.2	64.2	271	302	62	9	98.0	424
	12	2	3	6	75.0	69.2	293	277	62	10	98.8	429
	12	3	3	6	82.0	77.0	275	260	69	6	98.8	360
	6	3	3	6	76.6	73.0	273	289	69	7	96.8	362
	1	2	3	3	5.6	2.8	13	14	33	21	12.6	60

Results are shown in Tab. 6. We focus on two metrics: all-atom co-designability and the structural diversity of the all-atom co-designable samples (Div. Str.). For each sampling budget we retain only the parameter combinations that are Pareto-optimal in this two-metric space. To avoid discarding near-optimal settings that fall marginally short of the frontier, we relax the criterion to an ϵ-Pareto set with $\epsilon = 2\%$: a configuration is kept unless some other configuration (at the same budget) exceeds it by more than 2% (relative) on both metrics simultaneously. In addition, we always report the distillation-matched setting ($k_\mathbf{x}$=12, $\sigma_\mathbf{x}$=2, $k_\mathbf{y}$=3, $\sigma_\mathbf{y}$=3), the teacher configuration our distilled student is trained to reproduce, even when it is not (near-)Pareto-optimal, as it is the natural reference for the student. Finally, we also report the variant with ($k_\mathbf{x}$=1, $\sigma_\mathbf{x}$=2, $k_\mathbf{y}$=3, $\sigma_\mathbf{y}$=3). Although it performs markedly worse (its co-designability collapses), we include it to illustrate how strongly the TSR scaling parameters affect performance.

G.3 Extended MolProbity Atomistic Structure Analysis

Fig. 16 shows MolProbity and Clash scores for different variants of *Proteina-Pronto* and for La-Proteina with the SDE sampler, at various number of sampling steps.

G.4 Extended Rotamer Distribution Analysis

Figs. 17 and 18 show the rotamer distribution for the χ_1 angle produced by La-Proteina with the SDE sampler and different variants of *Proteina-Pronto* at a different number of sampling steps for all aminoacids.

G.5 Curvature analysis for *Proteina-Pronto* Coupled Velocity

As discussed in the main text, the coupling functions $\alpha_\mathbf{y}$ determine the velocity field to be distilled. One property of velocity fields that affects distillation is given by their curvature. In this section we

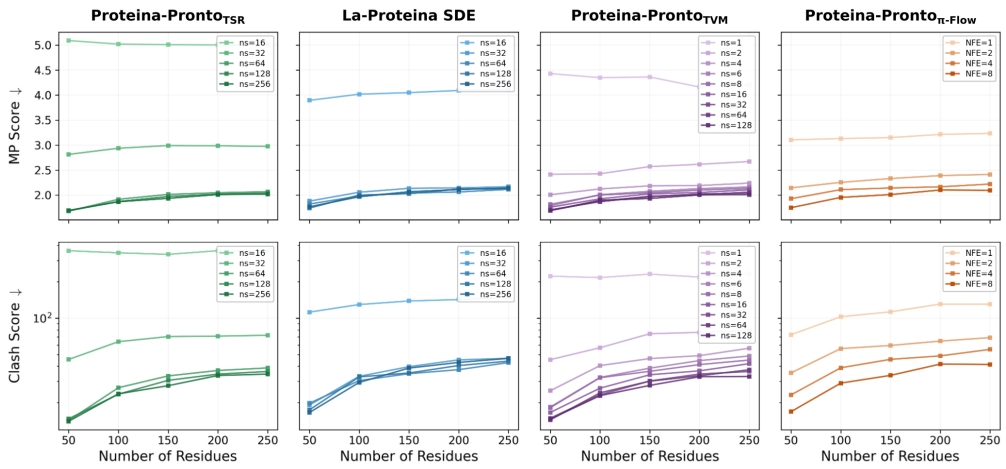

Figure 16: Additional MolProbity results for La-Proteina with the SDE sampler, and different *Proteina-Pronto* models.

analyze the curvature of the resulting velocity field for different coupling functions $\alpha_{\mathbf{y}}$, going from the uniform couplings gradually to the couplings used in practice, described in Sec. B.

We probe the geometry of the sampling trajectories generated by the tied-schedule TSR-rescaled ODE $\dot{\mathbf{y}}_t = \mathbf{v}_t^{\mathbf{y}}(\mathbf{x}_t, \mathbf{z}_t, t)$. We integrate this ODE with N Euler steps of size $\Delta t = 1/N$, producing velocities $\mathbf{v}_n := \mathbf{v}(\mathbf{x}_{t_n}, \mathbf{z}_{t_n}, t_n)$ at discrete times $t_n = n\Delta t$. At each step we measure two quantities. The first is the discrete curvature

$$\kappa_n \;=\; \frac{1 - \cos \measuredangle\big(\mathbf{v}_n, \mathbf{v}_{n+1}\big)}{\Delta t} \;=\; \frac{1}{\Delta t}\left(1 - \frac{\langle \mathbf{v}_n, \mathbf{v}_{n+1}\rangle}{\|\mathbf{v}_n\|\,\|\mathbf{v}_{n+1}\|}\right),$$

which captures how fast the direction of the velocity field rotates along the trajectory and is invariant to the local speed $\|\mathbf{v}_t\|$. The second is the discrete acceleration magnitude

$$a_n \;=\; \frac{\|\mathbf{v}_{n+1} - \mathbf{v}_n\|}{\Delta t},$$

a finite-difference proxy for $\|\mathrm{d}\mathbf{v}_t/\mathrm{d}t\|$ that captures the total rate of change of the velocity (combining changes in both direction and norm). Intuitively, κ diagnoses how curved the trajectory is in state space, while a diagnoses how stiff the ODE is in time. Fig. 19 reports both quantities averaged over $M = 15$ independent trajectories at proteins length $L = 100$. It can be observed that for the uniform couplings the curvature tends to be concentrated on the latter part of the trajectories. However, as we move towards to the coupling functions described in Sec. B, the curvature is distributed more broadly for all times t. We hypothesize this makes the distillation task harder.

G.6 Inference Time Profiling of Boltz2

Fig. 20 breaks down the inference time of Boltz2 on samples selected from PDB test set and CASP15, measured with CUDA events on NVIDIA H100 GPUs with bf16 mixed precision, 3 recycles, 200 sampling steps and a single structure per complex. The diffusion module and the Pairformer account for 41% and 32% of the wall-clock time. The CPU-side featurization and I/O, which distillation does not accelerate, accounts for 16%. A Pairformer pass takes 1.19 s, compared with 0.03 s for a denoising step, and its share of the wall-clock time grows with the biomolecular complex size from 12% below 256 tokens to 44% above 1,024 tokens. When 5 structures are predicted per complex in one batch, as in our evaluation, the share of the diffusion module increases to 65%.

G.7 Additional Evaluation of Boltz-2$_{\text{distill}}$

Here we provide full evaluation of Boltz-2$_{\text{distill}}$ with more metrics and additionally on the CASP15 set (Fig. 21). Note that we enabled stereocheck when evaluating the lDDT so the penalty is harsh

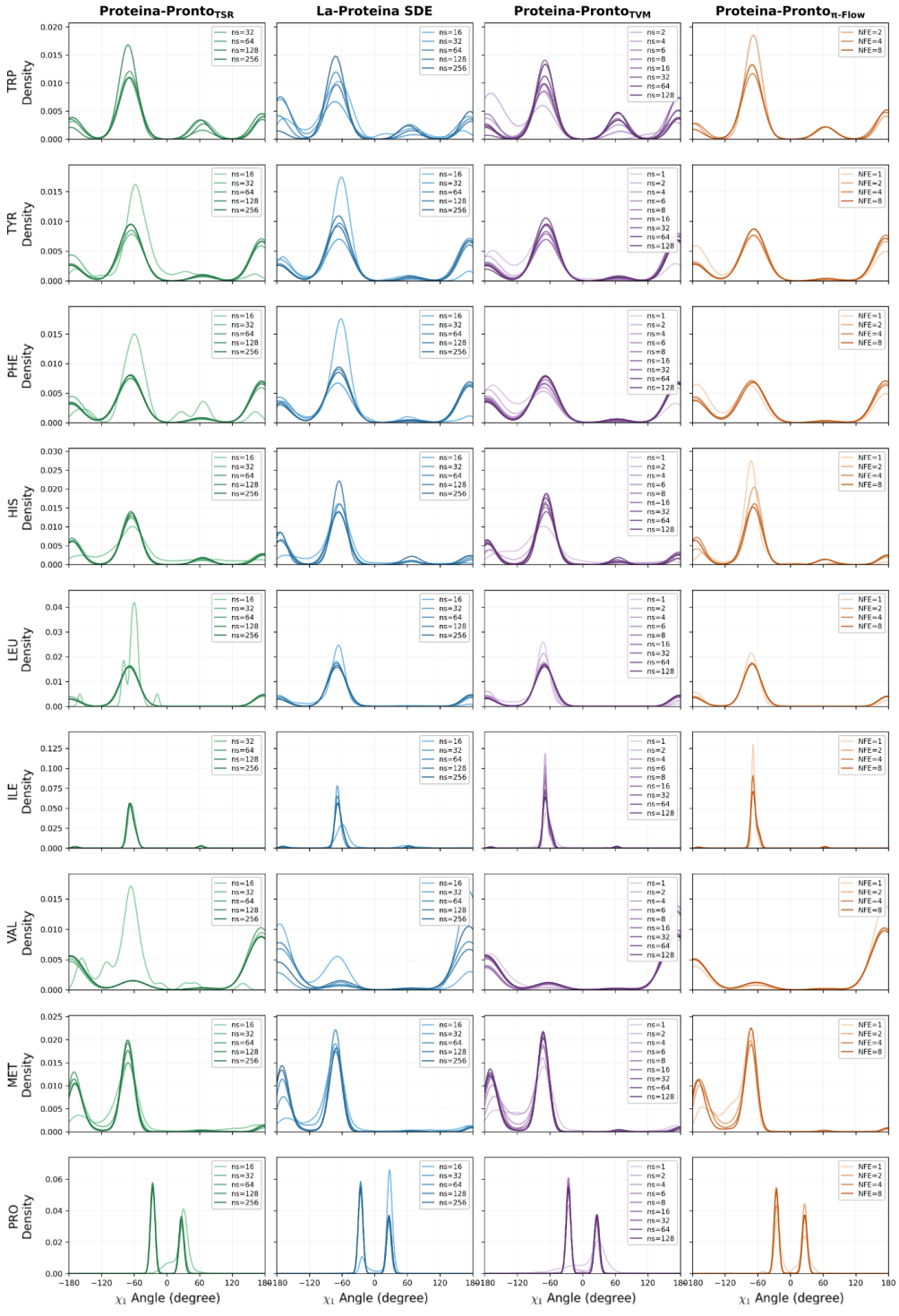

Figure 17: Distribution of χ_1 angle produced by La-Proteina SDE and different variants of *Proteina-Pronto* at different number of sampling steps for multiple aminoacids.

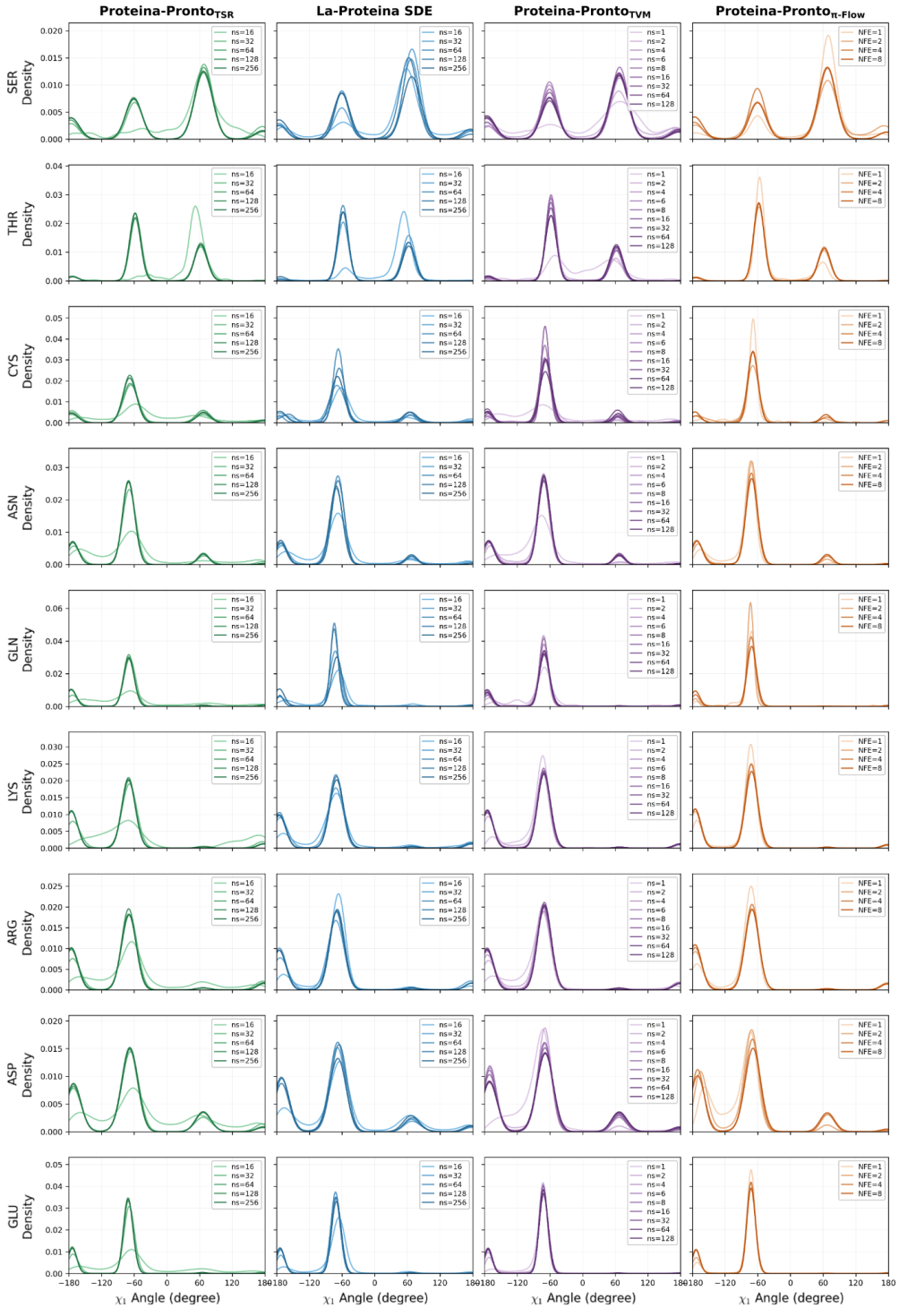

Figure 18: Distribution of χ_1 angle produced by La-Proteina SDE and different variants of *Proteina-Pronto* at different number of sampling steps for multiple aminoacids.

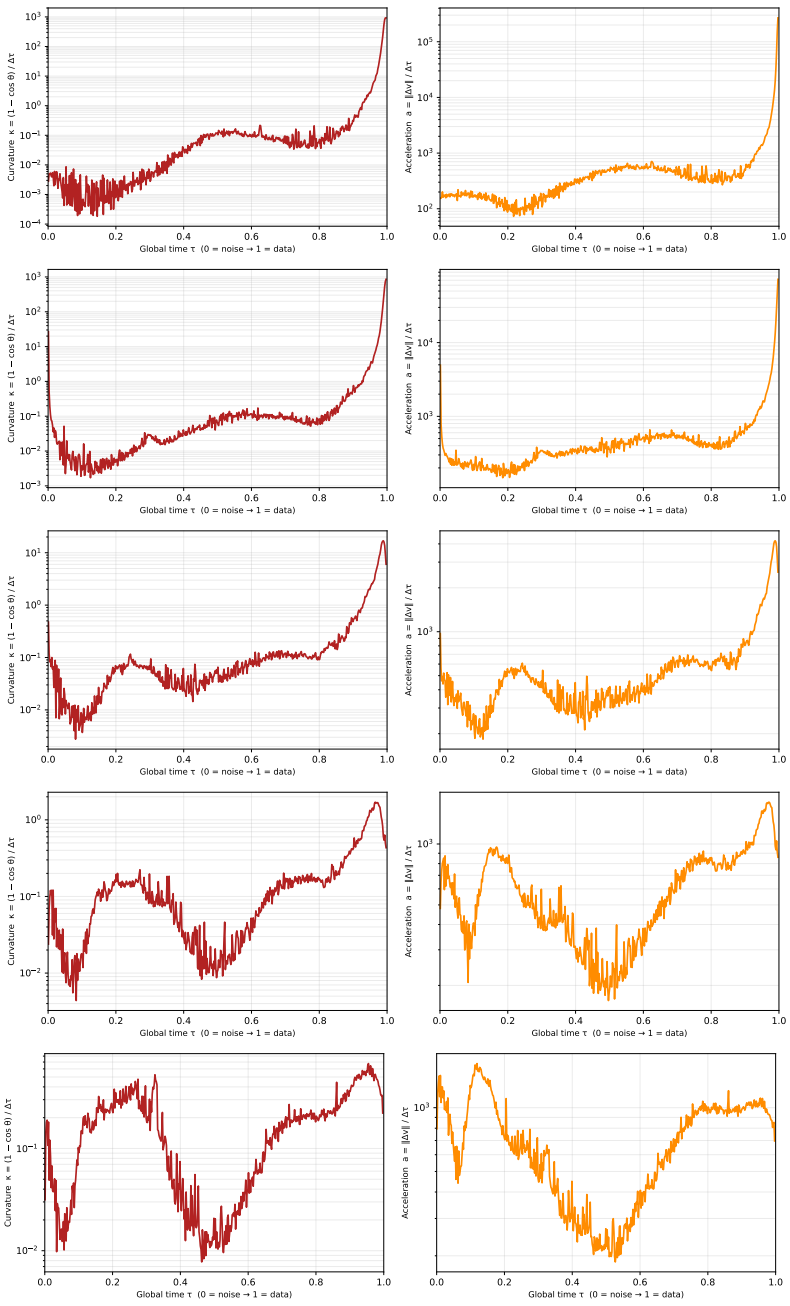

Figure 19: **Curvature analysis for the coupled velocity field for different schedule functions $\alpha_{\mathbf{y}}$.** From top to bottom, we gradually move from uniform coupling functions ($\alpha_{\mathbf{y}}(t) = t$) to the schedules used by *Proteina-Pronto*.

for side-chain structure with wrong stereochemistry. Besides the three configurations of Fig. 7, we evaluate two ablations that use only one of the two distilled components of Boltz-2_{distill}: the distilled trunk with the 200-step Boltz2 SDE sampler at step scale 1.5, and the original Boltz2 trunk with the 4-step π-Flow student. We also include Boltz2 with a 4-step ODE sampler at regular temperature, i.e., without noise injection and with a step scale of 1.0, which follows the few-step sampler configuration of Protenix-Mini (Gong et al., 2025).

On the PDB test set, the lDDT of Boltz-2_{distill} is higher than that of Boltz2 at regular temperature, and $\sim$1% lower than that of Boltz2 at low temperature. When used alone, the distilled trunk and

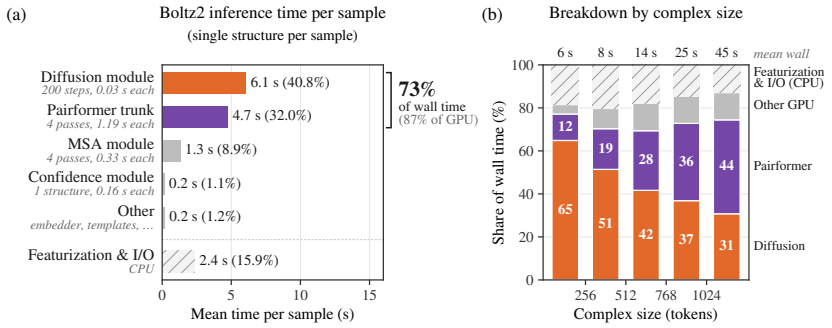

Figure 20: **Inference time of Boltz2** with 3 recycling iterations, 200 sampling steps and a single structure per complex. (a) Mean time per complex of each component. (b) Share of the wall-clock time by complex size, with the mean wall-clock time on top.

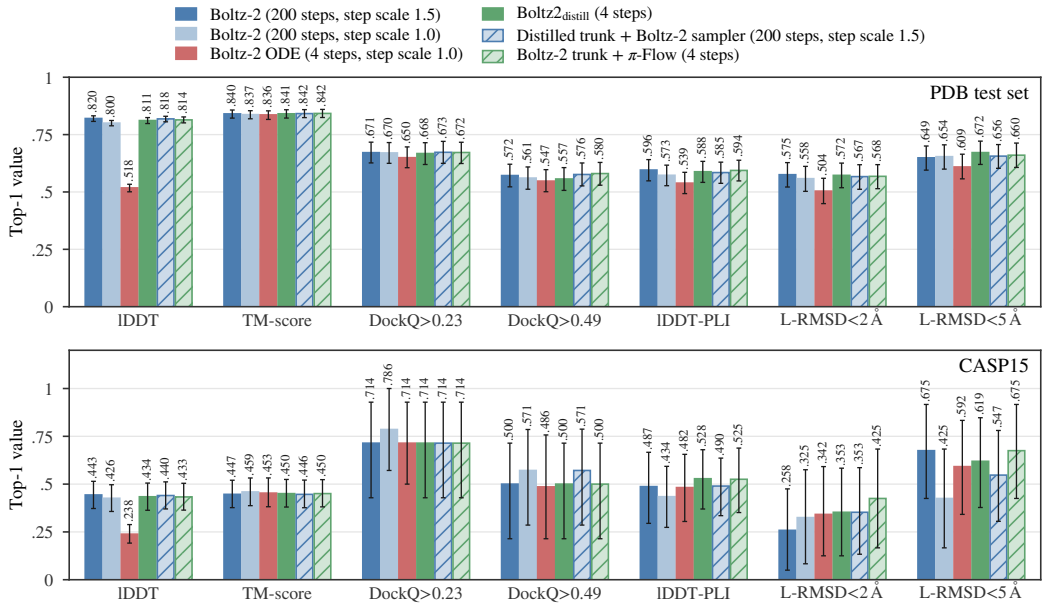

Figure 21: **Full evaluation on the PDB test set and CASP15.** Boltz2 with its 200-step sampler at step scale 1.5 and 1.0 and with a 4-step ODE sampler at step scale 1.0, Boltz-2_{distill}, and, striped, the two ablations that use only the distilled trunk or only the π-Flow student. Top-1 of 5 samples, without templates; error bars show 95% bootstrap confidence intervals. The 4-step ODE comes from an earlier evaluation that subsampled the MSA to 1,024 sequences, whereas all other configurations use the full MSA.

the π-Flow student give 0.818 and 0.814 lDDT, respectively, both exceeding the Boltz2 at regular temperature. In contrast, the 4-step ODE reaches only 0.518 lDDT, although its TM-score (0.836) is close to that of the 200-step samplers. 4-step ODE with original model preserves the global fold but not the local structure. On CASP15, the lDDT of Boltz-2_{distill} (0.434) again lies between those of Boltz2 at low (0.443) and regular (0.426) temperature, far above the 0.238 of the 4-step ODE. The interface and ligand metrics are only calculated on 12 to 14 targets in the CASP15 set so they have wide confidence intervals.

Tab. 7 reports the corresponding inference times. Distilling the diffusion module accounts for most of the speedup, as the 200-step sampler dominates the inference time of Boltz2 with batched 5-structures inference per complex (Sec. G.6). The trunk then becomes the most expensive part of the model, and distilling it further reduces the end-to-end time on the PDB test set from 11.3 to 8.0 seconds, bottleneck (42%) of which becomes featurization and I/O instead of any other GPU-running module.

Table 7: **Average inference time per complex** with 3 recycling steps and 5 structures per complex (batched inference to mimic real use case), measured with CUDA events on NVIDIA H100 GPUs with bf16 mixed precision. The total time includes featurization and I/O, and the GPU time the forward pass of the model only. Speedups over Boltz2 in parentheses. The last row is Boltz-2_{distill}; the times of the two ablations in the middle rows are composed from the measured times of their trunk and structure module.

		PDB test set		CASP15	
Trunk	Diffusion module	Total (s)	GPU (s)	Total (s)	GPU (s)
Boltz2	Boltz2, 200 steps	31.4	27.9	33.5	29.8
Distilled	Boltz2, 200 steps	28.1 (1.1×)	24.8 (1.1×)	30.5 (1.1×)	26.6 (1.1×)
Boltz2	π-Flow, 4 steps	11.3 (2.8×)	7.8 (3.6×)	13.0 (2.6×)	9.3 (3.2×)
Distilled	π-Flow, 4 steps	**8.0** (3.9×)	**4.7** (6.0×)	**10.0** (3.3×)	**6.1** (4.9×)

G.8 Hallucination-based binder design with Boltz-2_{distill}

BoltzDesign (Cho et al., 2025) designs a binder by backpropagating a structural objective through a frozen folding model into a continuous relaxation of the binder sequence. We use its default schedule of 125 optimization steps, of which 75 optimize softmax-relaxed logits, 45 anneal the temperature toward one-hot and 5 use straight-through one-hot logits, with 1 recycle step. BoltzDesign can compute its objective from the distogram head alone (Pairformer inference required for pair representation) or additionally from the confidence head , i.e., from the predicted lDDT and the predicted aligned error within the complex and at its interface. We use the latter mode (Pairformer+Confidence), which requires structure prediction at every optimization step. We compare two folding models that differ only in their trunk and sampler: the original Boltz2, and Boltz-2_{distill} with its distilled trunk and its 4-step π-Flow student.

Targets. We evaluate 10 targets across 3 different modalities. The 5 protein targets include PD-L1, PD-1, TrkA, DerF21, CD45. They are selected from the binder design benchmark of Didi et al. (2026b) and originally from target structures of the BindCraft (Pacesa et al., 2025) and AlphaProteo (Zambaldi et al., 2024) sets. The 4 small-molecule targets, IAI, FAD, SAM, and OQO, are adapted from RFdiffusion-AllAtom (Krishna et al., 2024) and BoltzDesign. The one nucleic-acid target (PDB ID: 5ZMC) is a DNA duplex included in BoltzDesign. We design 30 binders per target and model. We derive the length of each binder deterministically from its seed within the length window of its target, so that both models design binders of the same length from the same seed for fair comparison of computational efficiency.

Evaluation. We score every design with an inverse folding model followed by an independent folding model, using the evaluation pipeline of Didi et al. (2026b). Each BoltzDesign-generated binder is firstly redesigned by either SolubleMPNN for the protein targets or LigandMPNN for the ligand and DNA targets, keeping the residues within 4 Å of the target fixed, giving 8 sequences per design. We then refold each of these sequences together with its target (AlphaFold2-Multimer (Jumper et al., 2021; Evans et al., 2021) for the protein targets, RosettaFold3 (Corley et al., 2025) for the ligand targets, and Protenix-v2 (Chen et al., 2025b) for nucleic acid target). We finally apply the two success criteria of BoltzDesign to the refolded complexes: a complex pLDDT above 0.7 together with an interface PAE below 10 Å (strict) or below 15 Å (relaxed). A design counts as successful if at least one of its 8 redesigned sequences passes.

Results. Replacing Boltz2 with Boltz-2_{distill} reduces the time of an optimization step from 4.3–5.4 to 1.3–1.6 seconds, a speedup between 3.2 and 3.5× on all 10 targets (Tab. 8). The two models are on par on the ligand targets and the DNA target. On the protein targets, Boltz-2_{distill} generate ∼26% and ∼10% less successes based on strict and relaxed filter, respectively. Evaluating the designs is a real but model-independent cost, so we report this number both on the GPU-hours of the design and on those of the design+evaluation. Boltz-2_{distill} produces more successful designs per GPU-hour under every criterion, modality and cost basis, by 2.5 to 3.3× on the design alone and by 2.1 to 2.6× once the evaluation is included. Overall, BoltzDesign with Boltz-2_{distill} is a much more computation-efficient option. Fig. 22 shows two of the DNA binders designed by Boltz-2_{distill}.

Table 8: **Results of the BoltzDesign experiment.** A design counts as a success if at least one of its 8 redesigned sequences passes the criterion. Every success entry gives the number of successes and the successful designs per GPU-hour, counting the design alone (design-h) and the design+evaluation of its designs (total-h). The design and evaluation columns are the GPU-hours the 30 designs of a target take on NVIDIA H100 GPUs. The design time covers the optimization loop and the writing of the structures. The evaluation time contains the redesign of every binder and the refolding of its 8 sequences. The italic row of every modality pools its targets. The better of the two models is in bold.

	Boltz2				Boltz-2_{distill}			
Target	design (h)↓	eval. (h)↓	strict (count/per design-h/per total-h)↑	relaxed (count/per design-h/per total-h)↑	design (h)↓	eval. (h)↓	strict (count/per design-h/per total-h)↑	relaxed (count/per design-h/per total-h)↑
PDL1	4.79	0.31	**19**/4.0/3.7	**22**/4.6/4.3	1.43	0.30	13/**9.1/7.5**	19/**13.3/11.0**
PD1	5.09	0.34	**22**/4.3/4.1	24/4.7/4.4	1.50	0.33	**22/14.6/12.0**	**25/16.6/13.6**
TrkA	4.59	0.29	**16**/3.5/3.3	**19**/4.1/3.9	1.43	0.29	14/**9.8/8.1**	16/**11.2/9.3**
DerF21	5.65	0.37	**25**/4.4/4.2	**26**/4.6/4.3	1.70	0.35	17/**10.0/8.3**	25/**14.7/12.2**
CD45	5.33	0.32	**17**/3.2/3.0	**20**/3.8/3.5	1.57	0.32	7/**4.5/3.7**	15/**9.5/7.9**
Protein (5)	*25.45*	*1.62*	***99**/3.9/3.7*	***111**/4.4/4.1*	*7.64*	*1.59*	*73/**9.6/7.9***	*100/**13.1/10.8***
IAI	4.60	0.77	**27**/5.9/5.0	**30**/6.5/5.6	1.40	0.85	**27/19.3/12.0**	**30/21.4/13.3**
FAD	4.85	0.99	**30**/6.2/5.1	**30**/6.2/5.1	1.43	0.92	27/**18.8/11.5**	**30/20.9/12.8**
SAM	4.68	0.96	29/6.2/5.1	**30**/6.4/5.3	1.40	0.96	**30/21.4/12.7**	**30/21.4/12.7**
OQO	4.68	1.00	**30**/6.4/5.3	**30**/6.4/5.3	1.38	1.09	29/**21.0/11.7**	**30/21.7/12.1**
Ligand (4)	*18.80*	*3.72*	***116**/6.2/5.2*	***120**/6.4/5.3*	*5.62*	*3.82*	*113/**20.1/12.0***	***120/21.4/12.7***
DNA duplex (5ZMC)	4.55	1.14	**26**/5.7/4.6	**30**/6.6/5.3	1.38	1.15	24/**17.4/9.5**	**30/21.7/11.8**

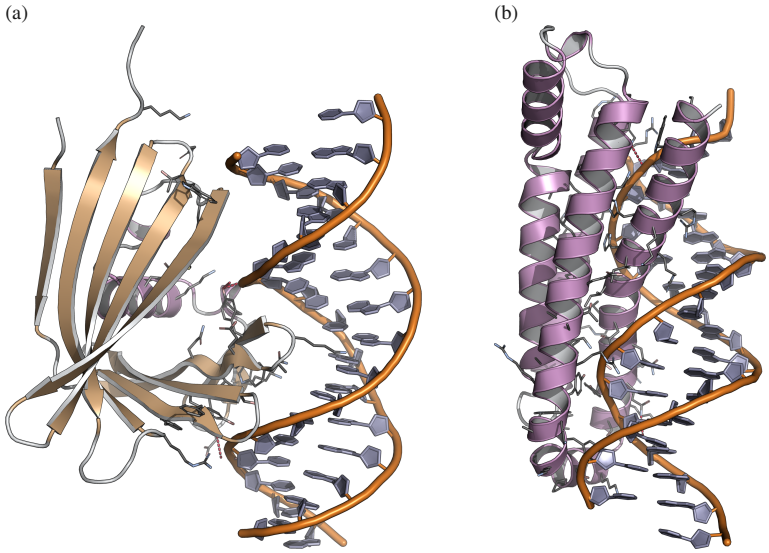

Figure 22: **DNA binders designed by Boltz-2_{distill} in BoltzDesign.** Two designs against the DNA duplex of PDB entry 5ZMC (orange): (a) a β-sheet binder and (b) a helical bundle, the lowest-ipAE design of this target.

G.9 Extended Results for Accelerated Inference-Time Search

This section complements Sec. 4.4, which reports aggregate results on hard targets. Fig. 23 reports the corresponding aggregates on easy targets, and Figs. 24 and 25 break both groups down per target, for AlphaFold2-Multimer and Boltz-2 scoring respectively. All curves report unique successful designs against GPU-hours covering both generation and folding-based evaluation, and follow the easy/hard target definitions of Didi et al. (2026b).

Easy targets. With AlphaFold2-Multimer scoring (Fig. 23, left), best-of-N with accelerated *Proteina-Pronto* models considerably outperforms the original Proteina-Complexa with beam search or best-of-N, yielding over twice as many unique successful binders for a given GPU-hour budget. As on hard targets (Fig. 8), accelerating evaluation compounds these gains: combining accelerated

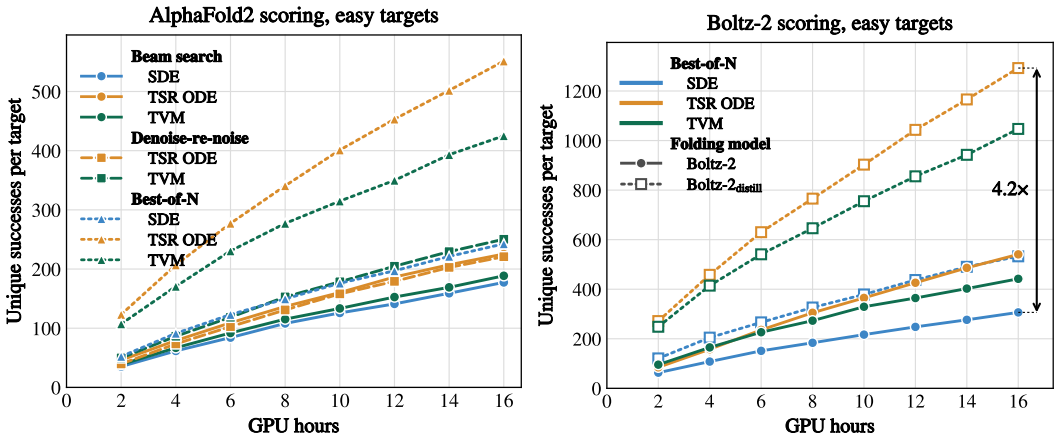

Figure 23: **Inference-time search on easy targets.** *(left)* AlphaFold2-Multimer scoring with different search algorithms and samplers. *(right)* Best-of-N with different samplers, scored with Boltz-2 and Boltz-2_{distill}. Budgets include generation and evaluation.

generation with Boltz-2_{distill} scoring yields $4.4\times$ as many unique successes per GPU-hour as SDE generation scored with regular Boltz-2 (Fig. 23, right).

Per-target results. The per-target curves show that the aggregate ordering of Fig. 8 does not hold uniformly. Accelerated generation improves over the SDE sampler on every target and at every budget, but the best search strategy varies. On targets with moderate success rates, such as SpCas9 and VEGFA, best-of-N with accelerated samplers is at or near the top. On the hardest targets, where successful designs are rare, guided search appears to be more effective: on HER2, beam search and denoise-re-noise with accelerated samplers produce more unique successes than best-of-N at the same budget.

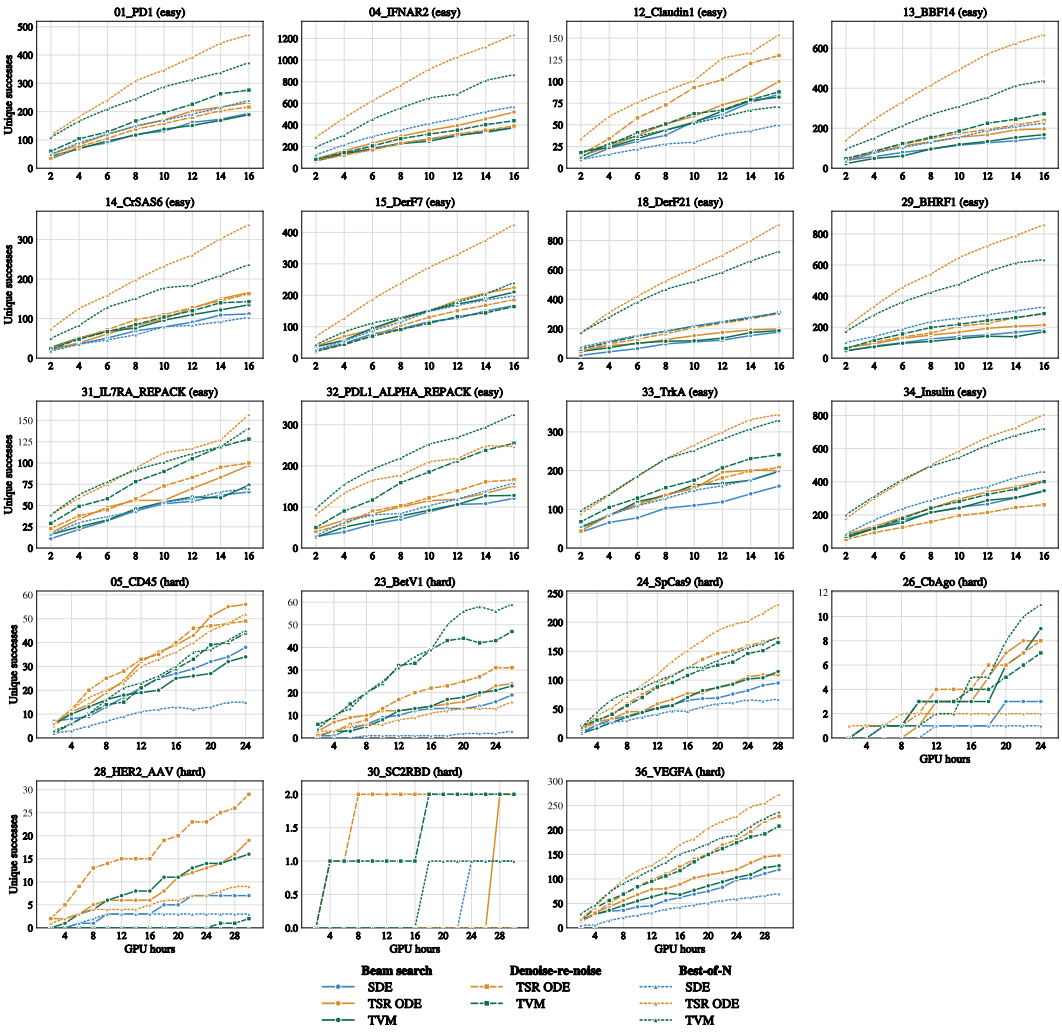

Figure 24: **Per-target inference-time search with AlphaFold2-Multimer scoring.** Unique successful designs against GPU-hours for each target, comparing beam search, denoise-re-noise and best-of-N with the SDE sampler, *Proteina-Pronto*$_{TSR}$ and *Proteina-Pronto*$_{TVM}$. Easy and hard targets are marked in the panel titles.

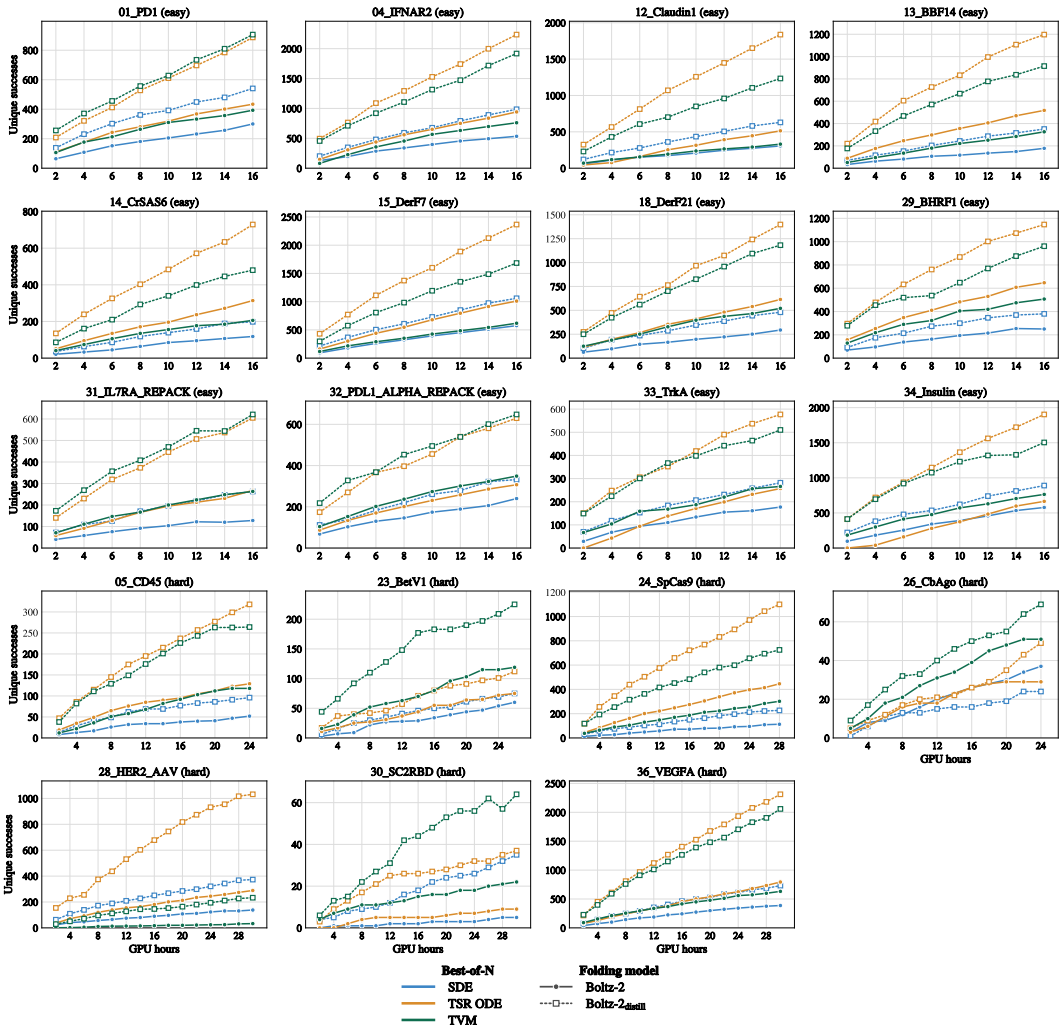

Figure 25: **Per-target best-of-N with Boltz-2 and Boltz-2$_{\text{distill}}$ scoring.** Unique successful designs against GPU-hours for each target, for the SDE sampler, *Proteina-Pronto*$_{TSR}$ and *Proteina-Pronto*$_{TVM}$, scored with the 200-step Boltz-2 and our 4-step Boltz-2$_{\text{distill}}$. Easy and hard targets are marked in the panel titles,