ATOMIC STRUCTURE MODELING WITH SPARSE VOXEL REPRESENTATIONS IN CRYO-EM

Anonymous authors
Paper under double-blind review

ABSTRACT

High-resolution cryo-electron microscopy maps provide rich stereochemical cues for proteins, nucleic acids, and ligands, but preserving this information throughout atomic modeling remains challenging. Existing learned builders rely on local map crops, patch tokens, or density samples at predicted atoms, without retaining voxel-wise semantics through coordinate generation. We present CryoASTRA, which computes on the 1 Å grid only within the macromolecular region and retains a semantic voxel representation through atomic modeling. It introduces (i) a *two-scale representation*: *EM points* aligned with the input sequences, and a semantic voxel representation from the voxel encoder for atomic modeling; (ii) *Heterogeneous BallTree Attention*, a sparse cross-attention where atoms query nearby voxels with linear attention cost for a fixed leaf size; and (iii) a GPU sparse pipeline for *whole-map inference* without sliding windows. On 103 sequence-novel maps from 2026 (1.5–3.9 Å), our model places on average 92.5% of protein atoms within 3 Å of deposited positions, has the highest TM-score among the evaluated builders on proteins and nucleic acids, and models the input ligands.

1 INTRODUCTION

Cryo-electron microscopy (cryo-EM) enables near-atomic determination of macromolecular structures in their native states (Nogales & Scheres, 2015). Automated model building converts an experimental 3D map, often together with the sequences of its components, into all-atom coordinates in the *map frame*, without a starting model of the target. High-resolution maps (better than 4 Å) grew from 16% of new EMDB releases in 2015 to 76% in 2025 (Appendix A), and at this resolution side chains, nucleotide bases, and bound ligands often become discernible in the map. Yet most learned builders model proteins only (Pfab et al., 2021; Giri & Cheng, 2024; Li et al., 2026b; Wang et al., 2025; Li et al., 2026a), a few also build nucleic acids (Jamali et al., 2024; Su et al., 2026b; Zhang et al., 2026), and dedicated builders model nucleic acids alone (Wang et al., 2023; Li et al., 2024); ligands are placed afterward by separate fitting tools (Muenks et al., 2023; Haloi et al., 2025).

Preserving such atomic features throughout atomic modeling calls for fine map sampling: each atom's density envelope is about as wide as a covalent bond (Appendix B), so the density features that separate neighboring atoms span only one or two voxels even at 1 Å. High-resolution cryo-EM maps typically contain between 256^3 and 512^3 grid points, and dense 3D networks such as 3D U-Nets (Çiçek et al., 2016; Falk et al., 2019) process every one of them. Yet in the median of 13,030 deposited maps at 4 Å or better, only 0.54% of the voxels lie at or above the author-recommended contour level (Appendix C). Existing learned builders reduce the map before atomic modeling, to *residue anchors* (Pfab et al., 2021; Giri & Cheng, 2024; Jamali et al., 2024; Su et al., 2026a;b; Li et al., 2026b; Zhang et al., 2026), *patch tokens* (Wang et al., 2025; Li et al., 2026a), or *density samples at atoms* (Li et al., 2026a). Each reduction limits what atomic modeling sees: anchors are defined per polymer residue, so a ligand has none, and side-chain detail survives at most in local crops of raw density; a token summarizes a whole patch in one vector; and a density sample exists only where an atom was already placed. Although several builders segment the map into atom and residue classes, *voxel-wise semantics are not retained through atomic modeling*. We ask whether a builder can retain a semantic voxel representation on a 1 Å grid throughout atomic modeling while confining computation to the macromolecular region.

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.

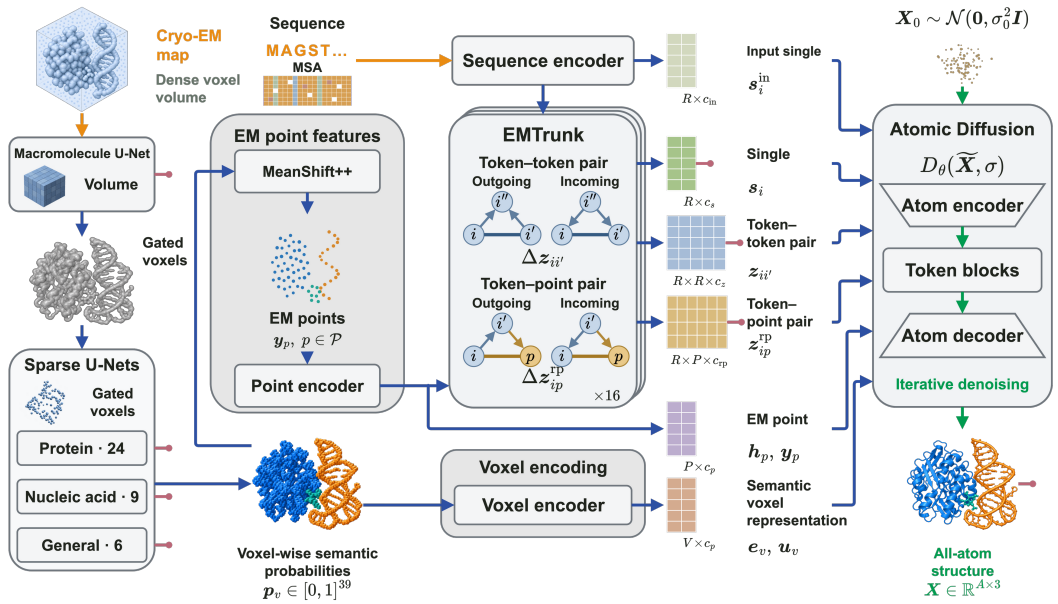

Figure 1: **CryoASTRA overview.** Sparse U-Nets predict voxel-wise semantic probabilities $\boldsymbol{p}_v$; the union across thresholded channels selects active voxels. MeanShift++ extracts EM points $\boldsymbol{y}_p$, aligned with sequence tokens by EMTrunk. The voxel encoder maps density and $\boldsymbol{p}_v$ to the semantic voxel representation $\boldsymbol{e}_v$, which conditions the atomic diffusion module alongside EMTrunk outputs to model all-atom coordinates $\boldsymbol{X}$. Coral indicates supervision.

We introduce **CryoASTRA** (**Cryo**-EM **A**tomic **STR**ucture modeling on 1 **Å** grids), which models proteins, nucleic acids, and input ligands in the map frame (Figure 1). Submanifold sparse U-Nets (Graham et al., 2018) predict voxel-wise probabilities for structural components and residue identities within the macromolecular region at 1 Å spacing; thresholding these probabilities selects a smaller set of *active voxels* for subsequent modeling. The model then builds a *two-scale representation*: MeanShift++ (Jang & Jiang, 2021) extracts *EM points*, targeting one per polymer residue or ligand atom, while the voxel encoder maps density and probabilities on active voxels to a *semantic voxel representation*. EMTrunk aligns EM points with the input sequences using whole-assembly context, starting from pretrained structure-prediction weights. The *atomic diffusion module* reads EMTrunk outputs and queries the semantic voxel representation through *Heterogeneous BallTree Attention*, in which atoms query nearby voxels within joint atom–voxel *Overlapping BallTree* leaves. With solvent excluded, the sparse U-Nets process up to 1.2×10^8 macromolecule-gated voxels in a single pass on one 80 GB GPU (Appendix D), enabling *whole-map inference* without sliding windows. Assembly size remains limited by EMTrunk's AlphaFold-style token-pair operators, whose cost grows quadratically to cubically with the number of tokens.

Our contributions are: **(i)** a *two-scale representation* that retains voxel-wise semantics through atomic modeling; **(ii)** *Heterogeneous BallTree Attention*, a sparse cross-attention between two point sets of different types, here atoms querying voxels, whose within-leaf attention cost is linear in both set sizes for fixed leaf size; and **(iii)** a GPU sparse pipeline (sparse convolution, Overlapping BallTree construction, and MeanShift++) for whole-map encoding without sliding windows. On a prospective evaluation set of 103 sequence-novel maps released in 2026, our model covers every residue and places on average 92.5% of protein atoms and 47.4% of nucleic-acid atoms within 3 Å of their deposited positions in the map frame, with TM-scores of 0.972 and 0.561 (Table 1). It also models the input ligands. An atom error head built from the same attention operators ranks atoms by their risk of map-frame error.

2 Related Work

Learning-based cryo-EM model building. Classical builders trace macromolecular chains through the map via heuristic or likelihood-based searches (Terwilliger et al., 2018; Cowtan, 2006); we group learned builders by what their atomic modeling stage receives from the map. *Residue anchors:* Most builders detect one C_α, phosphorus, or $\mathrm{C1}'$ position per residue and build from these anchors with traveling-salesman tracing, hidden Markov models, or attention over local crops (Pfab et al., 2021; Giri & Cheng, 2024; Jamali et al., 2024; Su et al., 2026a;b; Li et al., 2026b; Zhang et al., 2026); dedicated nucleic-acid builders trace sugar and phosphate backbones (Wang et al., 2023; Li et al., 2024); and the released E3-CryoFold conditions backbone generation on detected C_α positions (Wang et al., 2025). *Patch tokens:* CryoACE and the published formulation of E3-CryoFold encode the map as one token per cubic patch (Wang et al., 2025; Li et al., 2026a). *Density samples at atoms:* CryoACE also samples raw density at atomic positions predicted in previous self-refinement rounds (Li et al., 2026a). Across these builders, *voxel-wise semantics are not retained through atomic modeling*: ModelAngelo and CryoAtom2 predict probabilities only for anchor atoms; E3-CryoFold and CryoACE predict no voxel-wise classes; and while DeepTracer, Cryo2Struct, EMProt, and CryoZeta segment maps into component or residue classes, their segmentations are discarded once anchors are extracted. Furthermore, most map networks operate on sliding windows of 32^3–64^3 voxels (Pfab et al., 2021; Giri & Cheng, 2024; Su et al., 2026a;b; Li et al., 2026b; 2024). Ligands fall outside all of these builders: they are placed afterward by separate fitting tools (Muenks et al., 2023; Haloi et al., 2025) or not demonstrated (Zhang et al., 2026). CryoASTRA retains a semantic voxel representation through atomic modeling, using sparse computation over active voxels across the whole map.

Structure-prediction priors. Structure-prediction priors enter learned builders as initial weights (Li et al., 2026a), training-free guidance (Raghu et al., 2025), or docked sequence-based predictions (Jumper et al., 2021; Abramson et al., 2024) as in DeepMainmast and EMProt (Terashi et al., 2024; Li et al., 2026b). CryoASTRA initializes EMTrunk from pretrained AlphaFold-style weights (section 3.5), while its atomic diffusion module is trained from scratch in the map frame (section 3.6).

Sparse 3D networks and local geometric attention. Volumetric 3D networks scale cubically with box edge length, forcing foundation models to downsample or patchify dense volumes (Zhou et al., 2025). Submanifold sparse convolution avoids this explosion by evaluating kernels strictly at active voxels without dilation (Graham & van der Maaten, 2017; Graham et al., 2018), supported by GPU sparse engines (Choy et al., 2019; Tang et al., 2022; 2023); CryoASTRA also replaces quadratic mean shift (Comaniciu & Meer, 2002) with grid-based GPU MeanShift++ (Jang & Jiang, 2021). For local attention over irregular 3D point sets, point transformers partition coordinates via Swin-style windows (Sun et al., 2022; Liu et al., 2023), space-filling curves or octrees (Hilbert, 1891; Morton, 1966; Chen et al., 2022; Wang, 2023; Wu et al., 2024), or hashing (Miao et al., 2024), which cut Euclidean neighborhoods at partition boundaries. Tree transformers preserve metric neighborhoods (Omohundro, 1989); Erwin attends within equal-size BallTree leaves and alternates rotated partitions, but relies on multi-threaded CPU construction and virtual-node padding to balance trees (Zhdanov et al., 2025). CryoASTRA constructs an Overlapping BallTree entirely on the GPU without virtual nodes (section 3.3), powering BallTree attention for coordinates and Heterogeneous BallTree Attention for voxel-to-atom cross-attention with linear $\mathcal{O}((A+V)\ell)$ complexity.

3 Methods

CryoASTRA takes a cryo-EM map and the sequences of its polymer chains and ligand components, and models all-atom coordinates in the map frame (Figure 1).

Notation and tokenization. The map is resampled to a 1 Å grid, and the macromolecule network removes solvent before the sparse U-Nets run (section 3.1). Each output channel of the sparse U-Nets is binarized by a per-map Li threshold (Li & Lee, 1993; Li & Tam, 1998), and the *active voxels* $\mathcal{V}$ are the union of the voxels kept by any of these channels. Active voxels are indexed by v, with position $\boldsymbol{u}_v \in \mathbb{R}^3$ in the map frame and density ρ_v. The *voxel-wise semantic probabilities* $\boldsymbol{p}_v \in [0,1]^C$ have $C = 39$ channels indexed by c. *EM points* $\mathcal{P}$ are indexed by p with positions $\boldsymbol{y}_p$. Following AlphaFold 3 (Abramson et al., 2024), each standard polymer residue is one *token*

and each heavy atom of a ligand or modified residue is one token. Tokens $i = 1, \ldots, R$ carry single representations $\boldsymbol{s}_i \in \mathbb{R}^{c_s}$, token–token pair representations $\boldsymbol{z}_{ii'} \in \mathbb{R}^{c_z}$, and token–point pair representations $\boldsymbol{z}_{ip}^{\mathrm{rp}} \in \mathbb{R}^{c_{\mathrm{rp}}}$; atoms $a = 1, \ldots, A$ have coordinates $\boldsymbol{X} \in \mathbb{R}^{A \times 3}$.

3.1 VOXEL CLASSIFICATION AND SELECTION

At near-atomic resolution, neighboring atomic density envelopes overlap, so density maxima can merge or shift from atomic centers (Afonine et al., 2018), and a density value read at one atom position carries no local shape or gradient (Appendix B). CryoASTRA therefore predicts voxel-wise probabilities of structural components and residue identities within the macromolecular region. These probabilities select active voxels and, together with density, are encoded for atomic modeling (section 3.4).

Macromolecule network. The macromolecule network (Macromolecule U-Net in Fig. 1) is a 3D residual U-Net (Çiçek et al., 2016; Falk et al., 2019) that predicts a binary macromolecule-versus-solvent mask on a 128^3 resampling of the map; its input is cropped at the author-recommended contour level and has two channels, the normalized density and the same density with values below that contour set to zero. The mask is mapped back to the 1 Å grid, smoothed with a 3 Å Gaussian, and thresholded by Li's minimum cross-entropy criterion (Li & Lee, 1993; Li & Tam, 1998); voxels inside it are the *macromolecule-gated voxels* on which the sparse U-Nets run, and everything else is discarded before any later stage (Table 4).

Sparse U-Nets. Three submanifold sparse-convolutional residual U-Nets (Graham et al., 2018) operate independently on the positions and densities of the macromolecule-gated voxels and predict sigmoid probabilities for disjoint label groups: six structural components including ligand (*general*), 24 protein channels (residue center, disto atom, backbone, side chain, 20 amino-acid identities), and nine nucleic-acid channels (residue center, disto atom, sugar, phosphate, base, four nucleotide identities). The residue center is C_α for amino acids and $\mathrm{C1}'$ for nucleotides; the disto atom is the AlphaFold 3 distogram atom, C_β (C_α for glycine), C4 for purines, C2 for pyrimidines, and the ligand atom itself. The general network runs on every map and is the only source of the ligand channel; the protein and nucleic-acid networks run only when the input contains a chain of that type, and the channels of a network that is not run are set to zero. Concatenating the three outputs gives the 39-channel voxel-wise semantic probabilities $\boldsymbol{p}_v$. Because submanifold convolution never dilates the active set, cost is linear in the number of gated voxels rather than in the enclosing volume, and whole-map inference removes the window boundaries at which patch-based networks lose context. Each network minimizes a multi-label soft Dice loss $\mathcal{L}_{\mathrm{dice}}$ on labels painted at radius $\sqrt{2}$ Å, the face diagonal of the 1 Å grid, which keeps the labels of neighboring polymer disto atoms disjoint (Appendix B). The union of voxels retained by per-channel Li thresholds defines $\mathcal{V}$; thresholding reduces the voxel set but preserves continuous $\boldsymbol{p}_v$ for EM-point extraction and voxel encoding.

3.2 EM POINTS

EM points are extracted without learned parameters. For each of three source channels (protein disto atom, nucleic-acid disto atom, and ligand), a Li threshold selects voxels in $\mathcal{V}$, and MeanShift++ (Jang & Jiang, 2021) clusters their positions with a 1 Å bandwidth at a cost that grows with the number of selected voxels rather than with the map volume (Appendix E.1); for the ligand channel, 6-connected components smaller than eight voxels are removed first. Each cluster $\mathcal{M}_p$ of selected voxels yields one EM point at its mode $\boldsymbol{y}_p$, a point kind $\kappa_p \in \{\mathrm{prot}, \mathrm{NA}, \mathrm{lig}\}$, and the mean $\bar{\boldsymbol{p}}_p$ and elementwise maximum $\hat{\boldsymbol{p}}_p$ of $\boldsymbol{p}_v$ over its members; $\{(\boldsymbol{y}_p, \kappa_p, \bar{\boldsymbol{p}}_p, \hat{\boldsymbol{p}}_p)\}_{p \in \mathcal{P}}$ is the *EM point representation*. The target is one EM point per polymer residue or ligand token, so $P \approx R$, and the $R \times P$ token–point pair representation of EMTrunk does not grow with the map volume.

3.3 OVERLAPPING BALLTREE

Every local attention in CryoASTRA runs within the leaves of a spatial partition: *BallTree attention* is self-attention within leaves of EM points or atoms, and Heterogeneous BallTree Attention lets atoms query voxels within mixed atom–voxel leaves. As in Erwin (Zhdanov et al., 2025), each partition is a BallTree with leaves of equal size, alternated with a second partition $\mathcal{T}_{\mathrm{B}}$ built after a

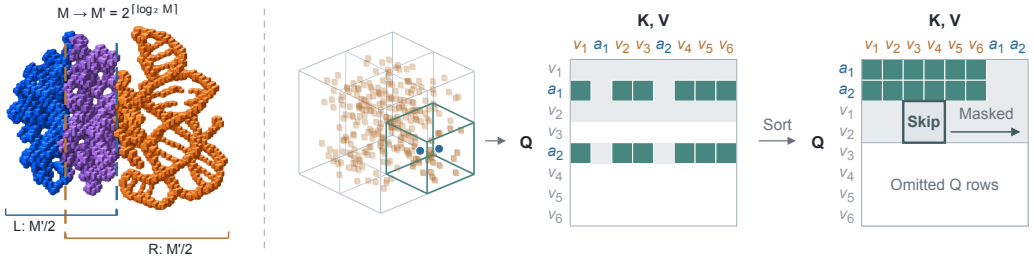

Figure 2: **Overlapping BallTree and Heterogeneous BallTree Attention. Left:** Balanced GPU partitioning where children share median rows (purple) to ensure power-of-two leaves without virtual padding. **Middle:** Joint atom–voxel partitioning into mixed leaves. **Right:** Sorted block mask where only the atom-query, voxel-key quadrant (teal) is computed, skipping fully masked tiles.

fixed 45° rotation of the primary partition $\mathcal{T}_{\mathrm{A}}$, so that information crosses leaf boundaries. Because atom positions change with each noise sample and sampler step, the trees are rebuilt at every evaluation of the atomic diffusion module, so we build an *Overlapping BallTree* on the GPU by batched argsort. Instead of adding Erwin's $M' - M$ virtual nodes, with $M' = 2^{\lceil \log_2 M \rceil}$, the root sorts the M rows along their most compact axis and takes the first and the last $M'/2$ rows as its children, which share the $M' - M$ median rows (Figure 2, left); lower levels bisect along the widest axis, duplicates are removed, and only the last leaf is padded, so attention runs on M rather than M' rows (Appendix E.2). Spill trees also let sibling nodes share points, but keep them for approximate nearest-neighbor search (Liu et al., 2004); here the overlap only completes the row count.

3.4 Point and voxel encoders

Run once per map, the point and voxel encoders produce encoded point features $\boldsymbol{h}_p$ and the semantic voxel representation $\boldsymbol{e}_v$. EMTrunk reuses $\boldsymbol{h}_p$ across blocks; the atomic diffusion module reuses both across noise samples and sampler steps.

Point encoder. An MLP embeds each EM point from its position, centered on the EM-point centroid, Fourier features of the position, a kind embedding, and $\bar{\boldsymbol{p}}_p, \hat{\boldsymbol{p}}_p$ into $\boldsymbol{h}_p^{(0)} \in \mathbb{R}^{c_p}$, $c_p = 128$. Four BallTree attention blocks with gated transitions and no rotary embeddings then refine the encoded point features $\boldsymbol{h}_p$ within leaves of at most $\ell = 256$ EM points, alternating between $\mathcal{T}_{\mathrm{A}}$ and $\mathcal{T}_{\mathrm{B}}$ (section 3.3); both residual branches are zero-initialized (Table 5).

Voxel encoder. A four-block submanifold sparse-convolutional encoder without downsampling maps the *voxel encoder input* $\boldsymbol{d}_v = [\rho_v; \boldsymbol{p}_v] \in \mathbb{R}^{40}$ to the *semantic voxel representation* $\boldsymbol{e}_v \in \mathbb{R}^{c_p}$, $c_p = 128$, keeping one row per active voxel at 1 Å registration.

3.5 EMTrunk

EMTrunk is a 16-block AlphaFold-style Pairformer with $c_s = 384$ and $c_z = 128$ (Abramson et al., 2024), preceded by the AlphaFold-style input embedder and a one-block MSA module that reads protein multiple sequence alignments (the sequence encoder in Fig. 1), and extended by a token–point stream that carries $\boldsymbol{h}_p$ and $\boldsymbol{y}_p$. Only the modules it shares with the pretrained AlphaFold-style model (input embedder, MSA module, and Pairformer blocks) start from that model's released weights (section 4.1; Appendix G); every EM addition below is newly initialized.

Token–point stream. Alongside $\boldsymbol{s}_i$ and $\boldsymbol{z}_{ii'}$, EMTrunk maintains a token–point pair $\boldsymbol{z}_{ip}^{\mathrm{rp}}$, initialized as $\boldsymbol{W}_{\mathrm{r}}\boldsymbol{s}_i + \boldsymbol{W}_{\mathrm{p}}\boldsymbol{h}_p$. Each block first applies the AlphaFold-style pair updates to $\boldsymbol{z}_{ii'}$, then updates $\boldsymbol{z}^{\mathrm{rp}}$ with (i) outgoing and incoming *mixed triangle multiplication*, which contracts $\boldsymbol{z}_{ii'}$ against $\boldsymbol{z}_{i'p}^{\mathrm{rp}}$ so token–token geometry propagates to token–point pairs, (ii) token-axis attention biased by $\boldsymbol{z}_{ii'}$, (iii) point-axis attention with a shared EM-point distance bias and 3D rotary embeddings of $\boldsymbol{y}_p$, and (iv) a transition. An outer-product mean over EM points writes $\boldsymbol{z}^{\mathrm{rp}}$ back to $\boldsymbol{z}_{ii'}$, linking token pairs

that see the same EM points. After the AlphaFold-style pair-biased attention, the single stream adds a point-to-token attention in which $\boldsymbol{s}_i$ queries $\{\boldsymbol{h}_p\}$ with bias $\boldsymbol{z}_{ip}^{\mathrm{rp}}$ (Algorithm 8).

Heads and losses. Beyond the token–token distogram, zero-initialized heads predict a token–point distogram from $\boldsymbol{z}^{\mathrm{rp}}$, a disto-atom coordinate per token from $\boldsymbol{s}_i$, and a refined coordinate per EM point from $\boldsymbol{h}_p$; the per-token coordinate supervises sequence–EM-point alignment in the shared map frame. The EMTrunk objective $\mathcal{L}_{\mathrm{trunk}} = \mathcal{L}_{\mathrm{rr}} + \mathcal{L}_{\mathrm{rp}} + 0.1\,\mathcal{L}_{\mathrm{res}} + 0.1\,\mathcal{L}_{\mathrm{pt}}$ combines cross-entropy distogram losses $\mathcal{L}_{\mathrm{rr}}, \mathcal{L}_{\mathrm{rp}}$ and smooth-L_1 coordinate losses $\mathcal{L}_{\mathrm{res}}, \mathcal{L}_{\mathrm{pt}}$ in the map frame (Table 5).

3.6 Atomic diffusion module

The atomic diffusion module follows the EDM formulation of AlphaFold 3 (Karras et al., 2022; Abramson et al., 2024), with $\sigma_{\mathrm{data}} = 16$ and the preconditioned denoiser D_θ of Eq. 2 (Appendix E), and generates coordinates in the map frame. Its network F_θ is conditioned on the EMTrunk outputs, the encoded point features $\boldsymbol{h}_p$, and the semantic voxel representation $\boldsymbol{e}_v$.

Atom stream architecture. Atoms are embedded from AlphaFold-style reference features together with EDM-scaled noisy positions and Fourier features of the un-scaled coordinates in angstroms, so atoms are placed in the same absolute map frame as the voxels. The atom encoder stacks four Heterogeneous BallTree Attention blocks followed by four BallTree attention blocks, and the decoder mirrors this order. Where AlphaFold 3 restricts atom self-attention to windows along the sequence, BallTree attention partitions atoms in 3D into Overlapping BallTree leaves of $\ell_a = 128$ atoms; it uses AlphaFold-style atom-pair biases and 3D rotary position embeddings.

Heterogeneous BallTree Attention. Heterogeneous BallTree Attention routes the semantic voxel representation $\boldsymbol{e}_v$ to the atom states $\boldsymbol{a}_a$ (Figure 2). Atoms and active voxels are partitioned jointly into mixed Overlapping BallTree leaves of $\ell_{av} = 256$ rows, in which atoms query voxels: queries are projected only for atoms and keys and values only for voxels. Rows in each leaf are sorted so that atoms precede voxels, and FlexAttention (Dong et al., 2025) masks the $A \to A$, $V \to V$, and $V \to A$ blocks (written query→key) and skips fully masked tiles, so only the atom-query, voxel-key quadrant is computed up to partially masked boundary tiles; leaves without atoms are dropped. Logits are biased by Gaussian radial basis functions of $\|\boldsymbol{x}_a - \boldsymbol{u}_v\|$, and queries and keys carry 3D rotary embeddings.

Token stream. Atom features are pooled to R tokens ($c_t = 768$) and processed by eight blocks of $\boldsymbol{z}_{ii'}$-biased token self-attention, point-to-token attention as in EMTrunk, and a conditioned transition; tokens are broadcast back to atoms for the decoder, whose linear head produces F_θ.

Training objective and sampling. Noise levels follow $\log(\sigma/\sigma_{\mathrm{data}}) \sim \mathcal{N}(-1.2, 1.5^2)$ with $S = 16$ noise samples per structure per step. The loss is $\mathcal{L} = \mathcal{L}_{\mathrm{trunk}} + \frac{1}{3}\,\mathcal{L}_{\mathrm{mse}} + \mathcal{L}_{\mathrm{lddt}}$, where $\mathcal{L}_{\mathrm{mse}}$ is the EDM-weighted squared error of D_θ (Eq. 3) with per-atom weights $w_a = 1$, 6, and 11 for protein, nucleic-acid, and ligand atoms, and $\mathcal{L}_{\mathrm{lddt}}$ the smooth lDDT of AlphaFold 3 on local atom pairs. Targets are symmetry-permuted, and $\mathcal{L}_{\mathrm{mse}}$ is computed in the map frame without aligning the prediction to the reference, so the model must learn absolute placement from the map. Sampling follows a deterministic Euler solver with a Karras noise schedule, generating five samples per map; the complete inference settings are given in Appendix E.7.

3.7 Atom error head

AlphaFold-style confidence heads estimate superposition-free or locally aligned errors (pLDDT, PAE, PDE) and so cannot measure map-frame error (Abramson et al., 2024). CryoASTRA adds an atom error head with its own weights: eight blocks of BallTree attention over the generated atoms, each followed by Heterogeneous BallTree Attention to the frozen $\boldsymbol{e}_v$, classify each atom's map-frame error e_a into $[0,1), [1,3), [3,5), [5,\infty)$ Å, giving the risks $r_{a,t}$ that $e_a \geq t$ Å, $t \in \{1,3,5\}$. It is trained last on samples of the frozen model (Appendix E.8).

Training procedure. Stages are trained in order on the *inference* outputs of the previous ones (macromolecule network and sparse U-Nets; EMTrunk with the point encoder; the atomic diffusion

module and the voxel encoder, jointly with EMTrunk; the atom error head on the frozen model), on crops near the deposited atoms, and run on complete assemblies. Training takes two weeks on eight NVIDIA H100 GPUs. Appendix G gives the stage-specific settings.

4 EXPERIMENTS

4.1 SETUP

Data. The training set contains cryo-EM entries released through 2025 at 4 Å or better (Appendix F). We separately prepare an evaluation set of single-particle entries first released between May and September 2026, after the training data of every CryoASTRA component, including the MSA version of Protenix-Mini (Gong et al., 2025) from which EMTrunk starts (Appendix G). Candidates are kept only if their MMseqs2 cluster (40% protein, 80% nucleic-acid identity; Steinegger & Söding, 2017) contains no entry of our training data or of Cryo2StructData (Giri et al., 2024), one entry is drawn per novel cluster, and protein MSAs use sequence databases released before the evaluation window. The evaluation set contains 103 targets that span 1.5–3.9 Å, reach 6,110 residues, and comprise 101 protein-, 14 nucleic-acid-, and 66 ligand-bearing targets, each evaluated as a complete assembly (Appendix I.1).

Baselines. We compare with ModelAngelo (Jamali et al., 2024), CryoAtom2 (Su et al., 2026a;b), the released version of E3-CryoFold (Wang et al., 2025), which generates protein backbones conditioned on C_α positions detected in the map, EMProt (Li et al., 2026b), and CryoACE (Li et al., 2026a), run with their official implementations on the complete deposited map and the input sequences (inputs and per-target coverage in Appendix I.1). CryoACE completed 50 targets, none above 2,050 residues, and Appendix I.2 compares all methods on them. On the 14 nucleic-acid targets we also run the dedicated nucleic-acid builders CryoREAD (Wang et al., 2023) and EM2NA (Li et al., 2024).

Evaluation protocol. All metrics except the TM-score are computed in the map frame without rigid-body alignment, so errors include global placement. CryoASTRA metrics are averaged over its five samples, without reference-based selection; CryoACE's available samples are averaged in the same way. Accuracy@3 Å counts unmodeled reference atoms as misses; RMSD is computed over identity-aware matched atoms (Kufareva & Abagyan, 2012) and is read with coverage, the fraction of reference atoms matched, and completeness, the fraction of residues or ligand instances modeled, which are 1 and 100% for CryoASTRA's proteins and nucleic acids by construction. The TM-score (Zhang & Skolnick, 2004) uses US-align (Zhang et al., 2022) in sequence-dependent mode (`-TMscore 7`). Metrics are macro-averaged over the targets on which they are defined (Appendix I.1).

4.2 STRUCTURE MODELING

Table 1 reports all-atom modeling; among the evaluated builders, only CryoASTRA builds proteins, nucleic acids, and ligands.

Proteins. On the 101 protein targets, CryoASTRA places 92.5% of atoms within 3 Å of their deposited positions (CryoAtom2: 90.3%; the other builders: 67.1–85.4%), reaches a TM-score of 0.972 (others: 0.830–0.891), and achieves the lowest matched RMSD at 3.47 Å (others: 5.24–9.96 Å, with CryoAtom2 at 6.64 Å). The baselines model 83.7–92.5% of the residues and CryoASTRA all of them, so part of the Accuracy@3 Å lead reflects completeness; against CryoAtom2, the paired TM-score gain is significant ($+0.117$, 95% bootstrap interval $[0.083, 0.152]$, Wilcoxon $p = 7 \times 10^{-11}$) and the Accuracy@3 Å gain is not ($+2.2$ points, $p = 0.43$). On the 50 targets that CryoACE completed, CryoASTRA has the highest TM-score and Accuracy@3 Å, while CryoAtom2 and EMProt have lower matched RMSDs over the residues they model (Appendix I.2).

Nucleic acids. On the 14 nucleic-acid targets, CryoASTRA reaches 47.4% Accuracy@3 Å and a joint TM-score of 0.561, against 40.6% and 0.424 for CryoAtom2; with 14 targets these differences are indicative, and a matched RMSD of 18.0 Å shows that many nucleotides are still misplaced. We

Table 1: **All-atom modeling on the evaluation set** (103 maps at 1.5–3.9 Å, map frame). CryoASTRA and CryoACE are averaged over samples (±: sample std. dev.). Bold: best; '—': unsupported or undefined. ‡: 50 completed targets (Appendix I.2). Metrics and variability in Appendices I.1–I.4.

Method	TM-score ↑	Accuracy@3 Å (%) ↑	RMSD (Å) ↓	Coverage ↑	Completeness (%) ↑
Protein *(101 targets)*					
CryoASTRA	**0.972** ±0.002	**92.49** ±0.36	**3.47** ±0.18	**1.000**	**100.00**
CryoACE‡	0.937 ±0.038	85.52 ±4.97	4.72 ±1.70	**1.000**	**100.00**
CryoAtom2	0.855	90.34	6.64	0.983	92.48
E3-CryoFold	0.891	67.05	9.96	0.958	90.61
EMProt	0.880	82.44	6.27	0.906	83.65
ModelAngelo	0.830	85.36	5.24	0.914	84.66
Nucleic acid (NA) *(14 targets)*					
CryoASTRA	**0.561** ±0.013	**47.39** ±1.34	**18.02** ±1.05	**1.000**	**100.00**
CryoAtom2	0.424	40.60	20.69	0.763	66.62
ModelAngelo	0.274	25.54	29.76	0.569	50.49
EM2NA	0.133	8.34	30.96	0.413	35.52
CryoREAD	0.143	6.85	46.44	0.800	71.88
Ligand *(66 targets)*					
CryoASTRA	—	47.35 ±4.65	15.08 ±1.44	**0.964**	**97.60**

attribute much of the gap to proteins to the maps themselves: nucleic-acid regions are often flexible, so their local resolution falls well below the reported global resolution and many nucleotides lack interpretable density.

Ligands. On the 66 ligand-bearing targets, with ligand components given as input, CryoASTRA models 97.6% of ligand instances and places 47.4% of ligand atoms within 3 Å, so more than half of the ligand atoms still lie away from their deposited sites. Two factors limit ligand modeling. Ligands often bind flexibly and are resolved at lower local resolution than the surrounding protein (Haloi et al., 2025), so their density frequently lacks atomic detail. In addition, the voxel-wise semantic probabilities contain a single ligand channel, with no ligand atom-type labels; ligand identities and chemical structures are supplied as inputs. The unmodeled instances are crystallization aids (Tris and phosphate), which are excluded from the input ligand list following the crystallization-aid list of AlphaFold 3 (Abramson et al., 2024); the only ligand of one target is such an aid, so ligand RMSD covers 65 targets. Sub-component breakdowns are in Appendix I.3, and Figure 3 shows examples.

4.3 EM-POINT LOCALIZATION

This probe evaluates voxel classification and EM-point extraction before voxel encoding and structure generation. EM points extracted from the disto-atom channels are matched one-to-one within 3 Å, in the map frame, to deposited disto atoms or residue centers (C_α, $C1'$), and ligand EM points to ligand-instance centroids (Appendix I.5). The evaluated builders detect at most one C_α or phosphorus position per residue, which ModelAngelo and CryoAtom2 delete by default, so the probe reports CryoASTRA alone.

Table 2: **EM-point localization** (map frame, 3 Å matching). Target counts in parentheses; protocol in Appendix I.5.

	Prec. (%)	Rec. (%)	Error (Å)	
			RMSD	Med.
Protein center (101)	98.36	94.57	1.52	1.52
Protein disto (101)	98.40	94.62	0.61	0.41
NA center (14)	87.34	84.38	2.47	2.49
NA disto (14)	94.26	90.78	0.64	0.35
Ligand instance (67)	34.41	65.36	1.36	1.28

Disto-atom matches reach 0.61 Å RMSD at 94.6% recall for proteins and 0.64 Å at 90.8% for nucleic acids (Table 2); the larger offsets to C_α (1.52 Å) and $C1'$ (2.47 Å) reflect residue geometry rather than localization error. Ligand precision is low by construction (one centroid, several EM points per instance); 65.4% of instances are recovered.

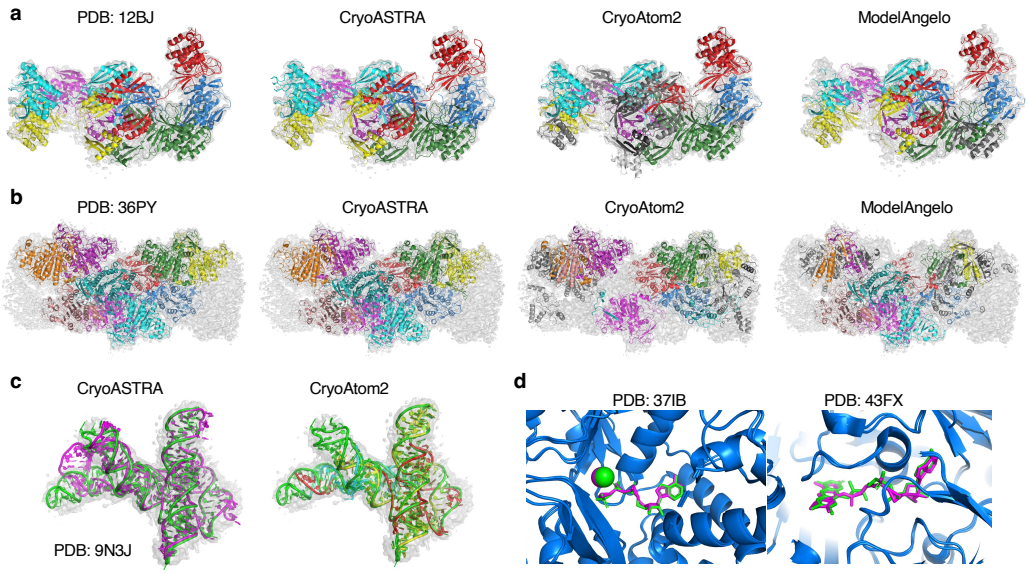

Figure 3: **Atomic models built by CryoASTRA on evaluation-set maps** in the map frame without rigid alignment. **(a, b)** Proteins PDB 12BJ (3.85 Å) and 36PY (3.70 Å) compared with CryoAtom2 and ModelAngelo (chains individually colored; CryoACE had no output due to out-of-memory errors). CryoASTRA builds complete models (TM-scores 0.994, 0.996) where baselines remain fragmented (0.577–0.831). **(c)** Nucleic acid target PDB 9N3J (3.27 Å; reference green, predictions colored). **(d)** Ligands: ATP/Mg^{2+} in PDB 37IB (3.20 Å) and FAD in PDB 43FX (2.69 Å; reference green, CryoASTRA magenta).

4.4 Operator efficiency

GPU Overlapping BallTree and MeanShift++ achieve up to 28.4× and 638× speedups over their CPU counterparts, respectively; sparse U-Nets use 4.2–4.4% less peak memory than TorchSparse++ (workloads, timings, and implementations in Appendix J).

4.5 Atom error estimation

Mean AUROCs are 0.912 [0.904, 0.920], 0.937 [0.922, 0.950], and 0.937 [0.920, 0.953] at $t = 1, 3, 5$ across 98 entries (five out-of-memory failures; 95% bootstrap CIs over entries; Appendix K). Pooled over atoms, $r_{a,3} \geq 0.8$ detects $e_a \geq 3$ Å with 91.6% [87.9, 93.9] precision and 16.2% [12.5, 20.6] recall, prioritizing inspection without certifying low-risk atoms.

5 Conclusion

CryoASTRA combines a two-scale representation with GPU sparse operators for whole-map modeling of proteins, nucleic acids, and input ligands in the map frame, retaining the semantic voxel representation at 1 Å throughout atomic modeling. On the evaluation set, it achieves the highest protein and nucleic-acid TM-scores among evaluated builders, with a protein matched RMSD of 3.47 Å. Overlapping BallTree and Heterogeneous BallTree Attention provide reusable GPU primitives for local geometric attention.

Limitations. Ablations are needed to isolate the contribution of the semantic voxel representation. Ligand identities are inputs, yet fewer than half of ligand atoms lie within 3 Å of deposited positions, requiring refinement. Local disorder and resolution may contribute to nucleic-acid errors, but were not measured. EMTrunk's token-pair operators remain a memory bottleneck, with quadratic-to-cubic computational cost.

AI USE STATEMENT

Generative AI tools were used to assist in code implementation and in drafting and polishing manuscript text. All AI-assisted code, algorithmic implementations, and text were thoroughly verified and validated by the authors. The authors take full responsibility for the entirety of this work.

ETHICS STATEMENT

This work uses only publicly released cryo-EM maps and atomic models from the EMDB and the PDB, together with their deposited sequences; it involves no human subjects and no personal or private data. CryoASTRA builds atomic models from experimentally determined maps and does not design new molecules, so it adds little capability for misuse beyond that of existing model-building software. Automatically built models can still contain errors, and nucleic-acid and ligand placements in particular remain unreliable in part (Section 4.2); they should be validated against the map and refined before deposition or downstream use. We follow the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, the sparse convolution engine optimizations and custom CUDA kernels are detailed in Appendix D. Explicit algorithmic pseudocode for all bespoke components—including GPU MeanShift++, Overlapping BallTree partitioning, BallTree attention, EMTrunk, Heterogeneous BallTree Attention, the atomic diffusion module, and the atom error head—is provided in Appendix E (Algorithms 1–13). Architectural specifications, loss functions, and optimization settings are documented in Section 3 and Appendix G (Tables 4–7). The training set and input preparation are described in Appendix F; the separately prepared evaluation set, its sequence-similarity exclusion against the training set and Cryo2StructData, and the evaluation protocol are detailed in Section 4.1 and Appendix I.1. Due to the complexity and specialized multi-GPU compilation environment required by the full end-to-end training and inference infrastructure, code is not bundled in an anonymous submission archive; upon publication, we will release the standalone inference pipeline, trained model checkpoints, and a public web server to make CryoASTRA directly accessible to the structural biology community.

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 Žemgulytė, 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 Žídek, Victor Bapst, Pushmeet Kohli, Max Jaderberg, Demis Hassabis, and John M. Jumper. Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature*, 630(8016):493–500, 2024. doi: 10.1038/s41586-024-07487-w.

Pavel V. Afonine, Billy K. Poon, Randy J. Read, Oleg V. Sobolev, Thomas C. Terwilliger, Alexandre Urzhumtsev, and Paul D. Adams. Real-space refinement in PHENIX for cryo-EM and crystallography. *Acta Crystallographica Section D: Structural Biology*, 74(6):531–544, 2018. doi: 10.1107/S2059798318006551.

ByteDance AML AI4Science Team, Xinshi Chen, Yuxuan Zhang, Chan Lu, Wenzhi Ma, Jiaqi Guan, Chengyue Gong, Jincai Yang, Hanyu Zhang, Ke Zhang, Shenghao Wu, Kuangqi Zhou, Yanping Yang, Zhenyu Liu, Lan Wang, Bo Shi, Shaochen Shi, and Wenzhi Xiao. Protenix – advancing structure prediction through a comprehensive AlphaFold3 reproduction. *bioRxiv*, 2025. doi: 10.1101/2025.01.08.631967.

Wanli Chen, Xinge Zhu, Guojin Chen, and Bei Yu. Efficient point cloud analysis using Hilbert curve. In *Computer Vision – ECCV 2022*, volume 13662 of *Lecture Notes in Computer Science*, pp. 730–747, 2022. doi: 10.1007/978-3-031-20086-1_42.

Christopher B. Choy, JunYoung Gwak, and Silvio Savarese. 4D spatio-temporal ConvNets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3075–3084, 2019. doi: 10.1109/CVPR.2019.00319.

Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. 3D U-Net: Learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, volume 9901 of *Lecture Notes in Computer Science*, pp. 424–432, 2016. doi: 10.1007/978-3-319-46723-8_49.

Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002. doi: 10.1109/34.1000236.

Kevin Cowtan. The Buccaneer software for automated model building. 1. tracing protein chains. *Acta Crystallographica Section D*, 62(9):1002–1011, 2006. doi: 10.1107/S0907444906022116.

Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. FlexAttention: A programming model for generating fused attention variants. In *Proceedings of Machine Learning and Systems*, 2025. URL https://proceedings.mlsys.org/paper_files/paper/2025/hash/61a9278dfef5f871b5e472389f8d6fa1-Abstract-Conference.html. arXiv:2412.05496.

Thorsten Falk, Dominic Mai, Robert Bensch, Özgün Çiçek, Ahmed Abdulkadir, Yassine Marrakchi, Anton Böhm, Jan Deubner, Zoe Jäckel, Katharina Seiwald, Alexander Dovzhenko, Olaf Tietz, Cristina Dal Bosco, Sean Walsh, Deniz Saltukoglu, Tuan Leng Tay, Marco Prinz, Klaus Palme, Matias Simons, Ilka Diester, Thomas Brox, and Olaf Ronneberger. U-Net: deep learning for cell counting, detection, and morphometry. *Nature Methods*, 16(1):67–70, 2019. doi: 10.1038/s41592-018-0261-2.

Nabin Giri and Jianlin Cheng. De novo atomic protein structure modeling for cryoEM density maps using 3D transformer and HMM. *Nature Communications*, 15(1):5511, 2024. doi: 10.1038/s41467-024-49647-6.

Nabin Giri, Liguo Wang, and Jianlin Cheng. Cryo2StructData: A large labeled cryo-EM density map dataset for AI-based modeling of protein structures. *Scientific Data*, 11:458, 2024. doi: 10.1038/s41597-024-03299-9.

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.

Benjamin Graham and Laurens van der Maaten. Submanifold sparse convolutional networks. *CoRR*, abs/1706.01307, 2017. URL https://arxiv.org/abs/1706.01307.

Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3D semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9224–9232, 2018. doi: 10.1109/CVPR.2018.00961.

Nandan Haloi, Rebecca J. Howard, and Erik Lindahl. Cryo-EM ligand building using AlphaFold3-like model and molecular dynamics. *PLOS Computational Biology*, 21(8):e1013367, 2025. doi: 10.1371/journal.pcbi.1013367.

David Hilbert. Ueber die stetige Abbildung einer Linie auf ein Flächenstück. *Mathematische Annalen*, 38:459–460, 1891. doi: 10.1007/BF01199431.

Kiarash Jamali, Lukas Käll, Rui Zhang, Alan Brown, Dari Kimanius, and Sjors H. W. Scheres. Automated model building and protein identification in cryo-EM maps. *Nature*, 628(8007):450–457, 2024. doi: 10.1038/s41586-024-07215-4.

Jennifer Jang and Heinrich Jiang. MeanShift++: Extremely fast mode-seeking with applications to segmentation and object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4102–4113, 2021. doi: 10.1109/CVPR46437.2021.00409.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, 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(7873):583–589, 2021. doi: 10.1038/s41586-021-03819-2.

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*, volume 35, pp. 26565–26577, 2022. doi: 10.52202/068431-1926.

Irina Kufareva and Ruben Abagyan. Methods of protein structure comparison. In Andrew J. W. Orry and Ruben Abagyan (eds.), *Homology Modeling: Methods and Protocols*, volume 857 of *Methods in Molecular Biology*, pp. 231–257. Humana Press, Totowa, NJ, 2012. doi: 10.1007/978-1-61779-588-6_10.

C. H. Li and C. K. Lee. Minimum cross entropy thresholding. *Pattern Recognition*, 26(4):617–625, 1993. doi: 10.1016/0031-3203(93)90115-D.

C. H. Li and P. K. S. Tam. An iterative algorithm for minimum cross entropy thresholding. *Pattern Recognition Letters*, 19(8):771–776, 1998. doi: 10.1016/S0167-8655(98)00057-9.

Minzhang Li, Mingrui Li, Weichen Qin, Qihe Chen, Sixian Shen, Yuan Pei, Jiakai Zhang, and Jingyi Yu. CryoACE: An atom-centric framework for accurate and automated model building in cryo-EM. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026a. URL https://arxiv.org/abs/2606.31332.

Tao Li, Hong Cao, Jiahua He, and Sheng-You Huang. Automated detection and de novo structure modeling of nucleic acids from cryo-EM maps. *Nature Communications*, 15:9367, 2024. doi: 10.1038/s41467-024-53721-4.

Tao Li, Ji Chen, Hao Li, Hong Cao, and Sheng-You Huang. EMProt improves structure determination from cryo-EM maps. *Nature Structural & Molecular Biology*, 33(2):341–350, 2026b. doi: 10.1038/s41594-025-01723-1.

Ting Liu, Andrew Moore, Alexander Gray, and Ke Yang. An investigation of practical approximate nearest neighbor algorithms. In L. Saul, Y. Weiss, and L. Bottou (eds.), *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2004. URL https://proceedings.neurips.cc/paper_files/paper/2004/file/1102a326d5f7c9e04fc3c89d0ede88c9-Paper.pdf.

Zhijian Liu, Xinyu Yang, Haotian Tang, Shang Yang, and Song Han. FlatFormer: Flattened window attention for efficient point cloud transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1200–1211, 2023. doi: 10.1109/CVPR52729.2023.00122.

Saori Maki-Yonekura, Keisuke Kawakami, Kiyofumi Takaba, Tasuku Hamaguchi, and Koji Yonekura. Measurement of charges and chemical bonding in a cryo-EM structure. *Communications Chemistry*, 6(1):98, 2023. doi: 10.1038/s42004-023-00900-x.

Siqi Miao, Zhiyuan Lu, Mia Liu, Javier Duarte, and Pan Li. Locality-sensitive hashing-based efficient point transformer with applications in high-energy physics. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 35546–35569. PMLR, 2024.

G. M. Morton. A computer oriented geodetic data base and a new technique in file sequencing. Technical report, IBM Ltd., Ottawa, Canada, 1966.

Andrew Muenks, Samantha Zepeda, Guangfeng Zhou, David Veesler, and Frank DiMaio. Automatic and accurate ligand structure determination guided by cryo-electron microscopy maps. *Nature Communications*, 14:1164, 2023. doi: 10.1038/s41467-023-36732-5.

Eva Nogales and Sjors H. W. Scheres. Cryo-EM: A unique tool for the visualization of macromolecular complexity. *Molecular Cell*, 58(4):677–689, 2015. doi: 10.1016/j.molcel.2015.02.019.

Stephen M. Omohundro. Five Balltree construction algorithms. Technical Report TR-89-063, International Computer Science Institute, Berkeley, CA, 1989. URL https://steveomohundro.com/wp-content/uploads/2009/03/omohundro89_five_balltree_construction_algorithms.pdf.

Jonas Pfab, Nhut Minh Phan, and Dong Si. DeepTracer for fast de novo cryo-EM protein structure modeling and special studies on CoV-related complexes. *Proceedings of the National Academy of Sciences*, 118(2):e2017525118, 2021. doi: 10.1073/pnas.2017525118.

Rishwanth Raghu, Axel Levy, Gordon Wetzstein, and Ellen D. Zhong. Multiscale guidance of protein structure prediction with heterogeneous cryo-EM data. In *Advances in Neural Information Processing Systems*, volume 38, pp. 11329–11357, 2025. doi: 10.52202/085713-0344.

Martin Steinegger and Johannes Söding. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature Biotechnology*, 35(11):1026–1028, 2017. doi: 10.1038/nbt.3988.

Baoquan Su, Kun Huang, Zhenling Peng, Alexey Amunts, and Jianyi Yang. CryoAtom improves model building for cryo-EM. *Nature Structural & Molecular Biology*, 33(2):351–361, 2026a. doi: 10.1038/s41594-025-01713-3.

Baoquan Su, Lingfan Zhu, Wenfei Li, Wenkai Wang, Jiahui Liang, Kailong Zhao, Zhenling Peng, and Jianyi Yang. CryoAtom2: inverse folding-inspired deep learning enables accurate model building of protein–nucleic acid complexes in cryo-EM. LangTaoSha Preprint Server, 2026b. URL https://langtaosha.org.cn/lts/en/preprint/view/185.

Pei Sun, Mingxing Tan, Weiyue Wang, Chenxi Liu, Fei Xia, Zhaoqi Leng, and Dragomir Anguelov. SWFormer: Sparse window transformer for 3D object detection in point clouds. In *Computer Vision – ECCV 2022*, volume 13670 of *Lecture Notes in Computer Science*, pp. 426–442, 2022. doi: 10.1007/978-3-031-20080-9_25.

Haotian Tang, Zhijian Liu, Xiuyu Li, Yujun Lin, and Song Han. TorchSparse: Efficient point cloud inference engine. In *Proceedings of Machine Learning and Systems*, volume 4, pp. 302–315, 2022.

Haotian Tang, Shang Yang, Zhijian Liu, Ke Hong, Zhongming Yu, Xiuyu Li, Guohao Dai, Yu Wang, and Song Han. TorchSparse++: Efficient training and inference framework for sparse convolution on GPUs. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 225–239, 2023. doi: 10.1145/3613424.3614303.

Genki Terashi, Xiao Wang, Devashish Prasad, Tsukasa Nakamura, and Daisuke Kihara. DeepMainmast: integrated protocol of protein structure modeling for cryo-EM with deep learning and structure prediction. *Nature Methods*, 21(1):122–131, 2024. doi: 10.1038/s41592-023-02099-0.

Thomas C. Terwilliger, Paul D. Adams, Pavel V. Afonine, and Oleg V. Sobolev. A fully automatic method yielding initial models from high-resolution cryo-electron microscopy maps. *Nature Methods*, 15(11):905–908, 2018. doi: 10.1038/s41592-018-0173-1.

Jue Wang, Cheng Tan, Zhangyang Gao, Guijun Zhang, Yang Zhang, and Stan Z. Li. End-to-end cryo-EM complex structure determination with high accuracy and ultra-fast speed. *Nature Machine Intelligence*, 7(7):1091–1103, 2025. doi: 10.1038/s42256-025-01056-0.

Peng-Shuai Wang. OctFormer: Octree-based transformers for 3D point clouds. *ACM Transactions on Graphics*, 42(4):155:1–155:11, 2023. doi: 10.1145/3592131. SIGGRAPH 2023.

Xiao Wang, Genki Terashi, and Daisuke Kihara. CryoREAD: de novo structure modeling for nucleic acids in cryo-EM maps using deep learning. *Nature Methods*, 20:1739–1747, 2023. doi: 10.1038/s41592-023-02032-5.

Francis Williams, Jiahui Huang, Jonathan Swartz, Gergely Klar, Vijay Thakkar, Matthew Cong, Xuanchi Ren, Ruilong Li, Clement Fuji-Tsang, Sanja Fidler, Eftychios Sifakis, and Ken Museth. fVDB: A deep-learning framework for sparse, large scale, and high performance spatial intelligence. *ACM Transactions on Graphics*, 43(4):133:1–133:15, 2024. doi: 10.1145/3658226.

Jeremy Wohlwend, Gabriele Corso, Saro Passaro, Noah Getz, Mateo Reveiz, Ken Leidal, Wojtek Swiderski, Liam Atkinson, Tally Portnoi, Itamar Chinn, Jacob Silterra, Tommi Jaakkola, and Regina Barzilay. Boltz-1: Democratizing biomolecular interaction modeling. *bioRxiv*, 2024. doi: 10.1101/2024.11.19.624167.

Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer V3: Simpler, faster, stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4840–4851, 2024. doi: 10.1109/CVPR52733.2024.00463.

wwPDB Consortium. EMDB—the electron microscopy data bank. *Nucleic Acids Research*, 52 (D1):D456–D465, 2024. doi: 10.1093/nar/gkad1019.

Chengxin Zhang, Morgan Shine, Anna Marie Pyle, and Yang Zhang. US-align: universal structure alignments of proteins, nucleic acids, and macromolecular complexes. *Nature Methods*, 19(9): 1109–1115, 2022. doi: 10.1038/s41592-022-01585-1.

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. doi: 10.1002/prot.20264.

Zicong Zhang, Shu Li, Farhanaz Farheen, Yuki Kagaya, Boyuan Liu, Nabil Ibtehaz, Genki Terashi, Tsukasa Nakamura, Han Zhu, Kafi Khan, Yuanyuan Zhang, and Daisuke Kihara. Accurate macromolecular complex modeling for cryo-EM with CryoZeta. *bioRxiv*, 2026. doi: 10.64898/2026.02.13.705846.

Maksim Zhdanov, Max Welling, and Jan-Willem van de Meent. Erwin: A tree-based hierarchical transformer for large-scale physical systems. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 78023–78040, 2025. URL https://proceedings.mlr.press/v267/zhdanov25a.html.

Yi Zhou, Yilai Li, Jing Yuan, and Quanquan Gu. CryoFM: A flow-based foundation model for cryo-EM densities. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=T4sMzjy7fO.

A EMDB RESOLUTION TREND

Fig. 4 shows yearly counts across the full catalog history (2002–2026), colored by reported reconstruction resolution, from an analysis of EMDB release metadata (catalog snapshot through September 9, 2026; the archive is described by (wwPDB Consortium, 2024)). Maps better than 4 Å grew from 16% of EMDB releases in 2015 to 76% in 2025 and made up more than 60% of entries released in 2022. The main text treats this trend as background rather than as the computational claim.

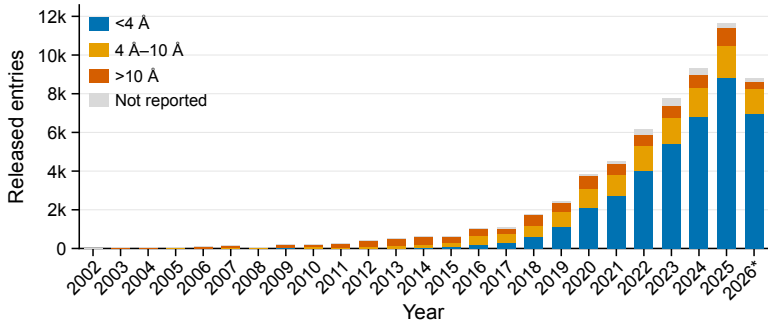

Figure 4: Released EMDB maps per year across the full catalog history (2002–2026), colored by reported reconstruction resolution: blue, below 4 Å; orange, 4–10 Å inclusive; vermilion, above 10 Å; gray, not reported. The catalog snapshot includes releases through September 9, 2026; 2026 is a partial year.

B MAP FORMATION AND DESIGN RATIONALE

This section expands the map-physics motivation and the choice of label radius in section 3.1. A cryo-EM map, which CryoASTRA resamples to a 1 Å grid, samples the electrostatic potential Φ of an assembly with atomic coordinates $\boldsymbol{X} = \{\boldsymbol{x}_a\}_{a=1}^{A}$, blurred by the microscope point spread function (PSF) and corrupted by noise η:

$$\rho(\boldsymbol{r}) = (\Phi * \mathrm{PSF})(\boldsymbol{r}) + \eta(\boldsymbol{r}) \approx \sum_{a=1}^{A} g_a(\boldsymbol{r} - \boldsymbol{x}_a) + \eta(\boldsymbol{r}), \tag{1}$$

where g_a is an isotropic Gaussian envelope whose width σ_a grows as the resolution worsens and with the B-factor.

Envelope width. Each atom contributes a Gaussian envelope $g_a(\boldsymbol{r}) = \frac{Z_a}{(2\pi\sigma_a^2)^{3/2}} \exp\big(-\frac{\|\boldsymbol{r}\|^2}{2\sigma_a^2}\big)$ with atomic number Z_a and width $\sigma_a \approx \sqrt{\sigma_{\mathrm{res}}^2 + B_a/(8\pi^2)}$, where σ_{res} reflects the map resolution and B_a is the atomic B-factor. At reported resolutions between 2.5 and 4.0 Å, $\sigma_a \approx 0.8$–1.2 Å, comparable to covalent bond lengths (1.2–1.5 Å) and not far below non-bonded packing distances (2.8–3.5 Å).

Peak displacement. Two equal isotropic envelopes a distance d apart sum to a single maximum when $d \le 2\sigma_a$ and to two maxima displaced toward each other when $d > 2\sigma_a$, so density maxima need not coincide with atomic centers. Refining a correct model toward higher sampled map values can accordingly move atoms away from the positions that generated the map, although these shifts are small relative to the resolved features (Afonine et al., 2018); density maxima alone therefore do not fix atomic coordinates.

Label radius. The sparse U-Nets are trained on labels painted as spheres of radius r around deposited atoms. If the spheres of two neighboring disto atoms touch, the disto-atom channel forms a continuous tube along the chain, and MeanShift++ merges or displaces the corresponding EM points. The closest neighboring polymer disto atoms are consecutive glycines (C_α–$C_\alpha \approx 3.8$ Å) and stacked bases (≈ 3.4 Å); consecutive C_β atoms are farther apart. A 2 Å radius ($2r = 4$ Å)

would fuse these neighbors. CryoASTRA uses $r = \sqrt{2}\,\text{Å} \approx 1.414\,\text{Å}$, the face diagonal of the 1 Å grid, so $2r \approx 2.83\,\text{Å}$ keeps the labels of neighboring polymer disto atoms disjoint and MeanShift++ can resolve one mode per residue. Ligand tokens are single atoms whose covalent neighbors lie 1.2–1.5 Å apart, so their labels overlap at this radius.

Scalar density sampling. Reading the raw density $\rho(\boldsymbol{x}_a) \in \mathbb{R}$ at an atom gives one scalar without directional context: two positions on opposite sides of a symmetric peak return the same value although they require opposite corrections, and a network applied to that scalar cannot recover the local gradient $\nabla\rho$, curvature, or anisotropic shape. Because cryo-EM measures the Coulomb potential rather than electron density, negatively charged groups such as aspartate and glutamate carboxylates can appear weak or vanish at the usual contour, depending on the retained spatial frequencies (Maki-Yonekura et al., 2023).

C ORIGINAL-MAP CONTOUR OCCUPANCY

To quantify the sparsity cited in the introduction, we measured each complete deposited MRC density grid for the 13,030 unique EMDB maps in our curated collection of EMDB maps at 4 Å or better. For each map, occupancy is the number of voxels at or above its author-recommended contour divided by the total number of voxels in the original grid, including zero-valued voxels. Shared maps are counted once, even if they have multiple associated structures; no crop, resampling, or model-derived mask enters this calculation. All 13,030 maps were measurable. Median occupancy is 0.538% (interquartile range 0.287%–1.003%), and 13,000/13,030 maps (99.77%) fall below 10% (Fig. 5).

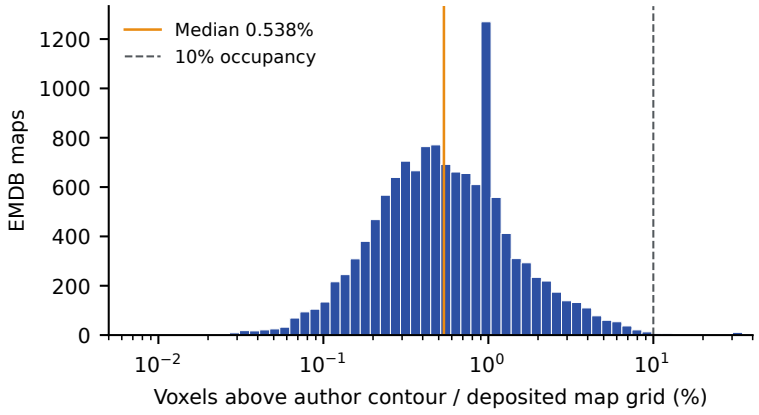

Figure 5: Original-map occupancy at the author-recommended contour for 13,030 deduplicated EMDB maps in our curated collection of EMDB maps at 4 Å or better. Blue bars show map counts on a logarithmic occupancy axis; orange marks the median and the dashed gray line marks 10% occupancy. No map had zero occupancy. These values do not measure the macromolecule network's active-voxel set.

D SPARSE CONVOLUTION ENGINE

The sparse U-Nets and the voxel encoder run on a customized CUDA sparse-convolution engine, updated from TorchSparse++ (Tang et al., 2023) (MIT license) for the operators these stages need and for cryo-EM maps: active sets of up to 1.2×10^8 voxels in a single inference pass, BF16 training with activation checkpointing, LayerNorm rather than BatchNorm, and `torch.compile` compatibility. Table 3 summarizes the differences.

Table 3: CryoASTRA's sparse-convolution engine versus upstream TorchSparse++ v2.1.

	TorchSparse++ v2.1	CryoASTRA (ours)
Dataflows	implicit GEMM, gather–GEMM–scatter, fetch-on-demand; autotuner selects	implicit GEMM only; fixed tile heuristic (Settings 1–3); forward split-K for under-filled grids
Kernel map	hashmap-on-the-fly (Murmur3 open addressing), sorted / split-mask variants	same hash map with allocator-backed tables, insert-failure counting, and resizing; new two-level *blockgrid* builder for strided downsampling
Precision	FP16 / TF32 / FP32	FP16 / BF16 / FP32 (TF32 MMA); BF16 unified with FP16 templates; SM80+
Convolutions	submanifold, strided, transposed, generative	submanifold, strided, transposed (map reuse required); generative removed
Fused epilogue	none	optional residual+bias epilogue on Setting-3 GEMM (FP16/BF16), with registered residual, bias, input, and weight gradients
Normalization	BatchNorm / InstanceNorm / GroupNorm wrappers	Triton mixed-dtype LayerNorm and fused LayerNorm+SiLU (BF16 activations, FP32 statistics and affine gradients)
Tensor type	mutable `SparseTensor` with per-tensor or global kernel-map cache	immutable pytree `SparseTensor`; kernel maps in a static `IndiceStore` keyed by `indice_key`; no process-global cache
Checkpointing	not a contract; maps live in mutable caches	`SparsePack` boundary with exactly two tensor children; non-reentrant `torch.utils.checkpoint` recomputes convolutions without rebuilding maps; backward-map materialization is idempotent
Compile / export	pybind extension	`TORCH_LIBRARY` ops with fake kernels, autocast and autograd registration; dynamic point dimension; compile and export supported
Extra ops	voxelize / devoxelize, `to_dense`, BEV	`SparseGridTable`: collision-free hash of the active grid with 8-corner trilinear query
Removed	—	CPU backend, Cython, autotuner, gather–scatter and fetch-on-demand kernels, plugin adapters

Kernel maps and implicit GEMM. A submanifold convolution with kernel offsets Δ computes $\boldsymbol{f}'_v = \sum_{\boldsymbol{\delta}\in\Delta} \boldsymbol{W}_{\boldsymbol{\delta}}\, \boldsymbol{f}_{n(v,\boldsymbol{\delta})}$ from input features $\boldsymbol{f}$ only for stored voxels v of the current sparse tensor, where $n(v,\boldsymbol{\delta})$ is the stored neighbor at offset $\boldsymbol{\delta}$ or empty. The neighbor table (kernel map) is built once per `indice_key` by inserting that tensor's integer coordinates into a GPU open-addressing hash table and probing each (voxel, offset) pair; all convolutions that share the same active set reuse the map, so a U-Net level pays for one map build regardless of depth. Convolution itself is the TorchSparse++ implicit GEMM: the sparse contraction is treated as a dense matrix product with $K = |\Delta|\, c_{\text{in}}$ whose rows are gathered inside the tensor-core tiles from the kernel map, with no materialized gather buffer. We keep only this dataflow: for the widths and active-set sizes of our U-Nets the autotuner in upstream almost always selected implicit GEMM, and removing the alternatives simplified the autograd, autocast, and fake-tensor registrations. Input gradients reuse the forward kernel on the transposed map; for odd-sized submanifold kernels the backward map aliases the forward map with flipped offset columns, so no second map is built. For strided downsampling

we added a two-level *blockgrid* builder that marks occupied coarse pages, counts neighbor pairs per page with a small scan, and materializes compact pair lists; it replaces the hash probe when the coarse grid is dense enough to make hashing the bottleneck, and submanifold maps keep the hash path where the blockgrid measured slower.

Checkpointing without map rebuilds. Training the sparse U-Nets on 1.6×10^7 voxels per rank is memory bound, so encoder and bottleneck stages run under `torch.utils.checkpoint`. Upstream keeps kernel maps in mutable caches attached to the tensor, which recomputation either loses or rebuilds. The customized engine makes the `SparseTensor` an immutable pytree whose only tensor children are features and coordinates; kernel maps live in a static `IndiceStore` that is excluded from the autograd graph and from Dynamo inputs. A `SparsePack` wraps the pair at the checkpoint boundary, so the recomputed forward finds the same maps and materializes backward maps idempotently. Decoder stages are never checkpointed because their skip connections are retained anyway. The same immutability lets the point dimension be marked dynamic for `torch.compile`, and every operator is registered through `TORCH_LIBRARY` with a fake kernel, so the voxel encoder compiles together with the atomic diffusion module that reads its output. Maps are rebuilt for each new active set, and checkpoint recomputation reuses them.

Fused mixed-dtype LayerNorm. Our U-Nets use LayerNorm rather than BatchNorm because active-voxel counts vary by two orders of magnitude between maps. In BF16 training the native implementation casts activations to FP32, normalizes, and casts back. The updated engine adds a Triton LayerNorm (optionally fused with SiLU) that reads BF16 rows, accumulates mean and variance in FP32, writes BF16 outputs, and keeps FP32 gradients for the affine parameters. The module remains an `nn.LayerNorm` subclass so checkpoint keys are unchanged.

Recorded comparisons against native LayerNorm used the same model and input on an NVIDIA RTX 5090 with PyTorch 2.13, CUDA 12.9, Triton 3.7.1, and seed 101. Timings are warm CUDA-event medians; training uses BF16 mixed precision and activation checkpointing. On a synthetic 10^6-voxel input, inference took 20.114 versus 33.520 ms ($1.67\times$ faster), and a training step took 72.791 versus 103.985 ms ($1.43\times$ faster). Peak allocated memory fell from 1.083 to 0.662 GiB for inference (39%) and from 5.248 to 3.017 GiB for training (43%). On a 152,190-voxel real map, inference took 5.444 versus 5.111 ms (7% longer), and training took 34.994 versus 23.770 ms (47% longer), while peak allocation fell from 0.219 to 0.167 GiB (24%) and from 0.887 to 0.545 GiB (39%), respectively. These are benchmark-wide peak allocations, not isolated LayerNorm workspace sizes. We retain the mixed-dtype path at small input sizes to trade execution time for lower peak memory; the production voxel budgets are established separately by the capacity sweep below.

Capacity. We measured capacity with synthetic sparse inputs and the production U-Net of Table 4, increasing the input-voxel count until allocation failed. A training trial comprised a complete BF16 forward, backward, and AdamW step, rather than the inference-only forward measured in Figure 9; runtime counters confirmed that activation checkpointing and fused mixed-dtype LayerNorm were both exercised. On one 80 GB A100, training passed at 1.9×10^7 input voxels (75.322 GiB peak allocated memory) and OOMed at 2.0×10^7. BF16 inference passed at 1.36×10^8 input voxels (77.129 GiB) and OOMed at 1.40×10^8. Production therefore keeps margin with 1.6×10^7 input voxels per rank for training and a whole-map inference budget of 1.2×10^8 input voxels in a single pass on one GPU.

E Algorithms

Pseudocode uses the notation of section 3 and Appendix G. LN is LayerNorm, $\mathrm{Attn}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}; \boldsymbol{b})$ is multi-head softmax attention with additive bias $\boldsymbol{b}$, Trans is a SwiGLU transition, and $\oplus$ denotes a residual addition. Dropout, masking of padded rows, and mixed-precision casts are omitted.

E.1 GPU MeanShift++

MeanShift (Comaniciu & Meer, 2002) finds the local maxima of an empirical density by repeatedly moving every point to a kernel-weighted mean of its neighbors. With a Gaussian or Epanechnikov

kernel, each iteration evaluates all pairs of the K selected voxels ($\mathcal{O}(K^2)$) or queries a spatial tree, which parallelizes poorly on the GPU.

Algorithmic formulation of MeanShift++. MeanShift++ (Jang & Jiang, 2021) quantizes space into cubic cells of width h and moves every voxel to the unweighted mean of the voxels in its cell and the 26 adjacent cells, so voxels of one cell move together. Its cost per iteration grows with the number of voxels rather than with the number of cells in the map volume.

GPU implementation and acceleration design. On high-resolution cryo-EM maps, a direct GPU port faces two key challenges: (i) *Volume sparsity*: a dense grid over the bounding box of a map (e.g., $400^3 \approx 6.4 \times 10^7$ cells) spends memory and bandwidth on cells that contain no selected voxel, although the selected voxels occupy only a small fraction of the box. (ii) *Memory bandwidth*: repeatedly reading, updating, and writing K coordinate vectors across iterations is bandwidth bound.

CryoASTRA accelerates MeanShift++ on the GPU with four optimizations (Algorithms 1 and 2):

1. **Sparse coordinate quantization and 1D key packing:** Instead of allocating a dense 3D volume, the coordinates of the selected voxels $\boldsymbol{Y} \in \mathbb{R}^{K\times 3}$ are mapped to integer bin coordinates $\boldsymbol{B} = \lfloor \boldsymbol{Y}/h \rfloor \in \mathbb{Z}^{K\times 3}$. Given the 3D bounding box $[\mathrm{mins}, \mathrm{maxs}]$ and strides $\mathbf{s}$, each 3D bin is packed into a single 64-bit integer key $k = \sum_{d=0}^{2}(B_d - \mathrm{mins}_d)s_d$. One GPU sort and deduplication of the K keys (`torch.unique`) yields the $M \leq K$ occupied cells and their sorted keys, so no memory is allocated for unoccupied cells.

2. **Fused 27-neighbor lookup CUDA kernel (CELLMEANS):** Weighted voxel coordinates and weights are accumulated into per-cell sums $\boldsymbol{S}$ and counts n with `scatter_add`. We then launch a dedicated CUDA kernel with one thread per occupied cell. For each cell (b_x, b_y, b_z), the thread iterates over the 27 spatial neighbor offsets $\boldsymbol{\delta} \in \{-1, 0, 1\}^3$, packs each neighbor's 1D key, and executes an in-register binary search over the sorted array of occupied keys. If a neighbor cell exists, its coordinate sum and point count are accumulated directly into thread registers. The thread then writes out the normalized 27-neighbor mean $\boldsymbol{Y}^{\mathrm{cell}} = \sum \boldsymbol{S} / \max(\sum n, 1)$. This fuses neighbor identification and coordinate reduction into a single kernel launch without materializing a neighbor list.

3. **Dynamic per-cell mode compression:** In the first iteration ($t = 1$), all K selected voxels move to the neighbor-cell means of their cells. From the second iteration on, the implementation replaces the voxels by the M resulting modes, each weighted by its cell count ($w = n$), and iterates only on these weighted modes while recording the mode-to-mode mapping inv_t. Because the voxels of one cell share their position after the first step, this compression leaves the result unchanged.

4. **Exact label recovery without spatial search:** When the maximum mode shift falls below the convergence threshold $\tau = 10^{-4}$ Å, coincident modes are merged via `torch.unique` to define the final EM points $\{\boldsymbol{y}_p\}$. The full voxel-to-cluster assignment $\mathcal{M}_p$ is reconstructed by chaining the recorded inverse mappings $\mathrm{mode}[\mathrm{label}[\mathrm{inv}_0[v]]]$, without a nearest-neighbor search.

Algorithm 1 Fused neighbor-cell means (CELLMEANS)

Require: positions $\boldsymbol{Y} \in \mathbb{R}^{K\times 3}$, weights $w \in \mathbb{R}^K$, bandwidth h

function CELLMEANS($\boldsymbol{Y}$, w)
  $\boldsymbol{B} \leftarrow \lfloor \boldsymbol{Y}/h \rfloor$; $k \leftarrow$ pack rows of $\boldsymbol{B}$ into lexicographic integer keys
  (keys, inv) $\leftarrow \mathrm{unique}(k)$ ▷ sorted occupied cells; inv maps rows to cells
  $\boldsymbol{S} \leftarrow \mathrm{scatter_add}(\text{inv}, w \odot \boldsymbol{Y})$; $n \leftarrow \mathrm{scatter_add}(\text{inv}, w)$
  CUDA kernel, one thread per occupied cell b
  $\boldsymbol{r} \leftarrow \mathbf{0}$; $m \leftarrow 0$
  for $\boldsymbol{\delta} \in \{-1, 0, 1\}^3$ **do**
    $k' \leftarrow \mathrm{pack}(\boldsymbol{B}_b + \boldsymbol{\delta})$; $b' \leftarrow \mathrm{bsearch}(\text{keys}, k')$
    if found **then**
      $\boldsymbol{r} \mathrel{+}= \boldsymbol{S}_{b'}$; $m \mathrel{+}= n_{b'}$
    end if
  end for
  $\boldsymbol{Y}^{\text{cell}} \leftarrow \boldsymbol{r}/\max(m, 1)$
  return $\boldsymbol{Y}^{\text{cell}}$, n, inv
end function

Algorithm 2 GPU MeanShift++ with per-cell mode compression

Require: positions $\boldsymbol{U} = \{\boldsymbol{u}_v\}_{v=1}^K$ of the voxels selected by Li thresholding (Li & Lee, 1993; Li & Tam, 1998), bandwidth h, tolerance τ, max iterations T

$\boldsymbol{Y}$, w, $\text{inv}_0 \leftarrow$ CELLMEANS($\boldsymbol{U}, \mathbf{1}$) ▷ first iteration: one mode per occupied cell
label $\leftarrow$ identity on cells
for $t = 2, \ldots, T$ **do**
  $\boldsymbol{Y}'$, w', inv $\leftarrow$ CELLMEANS($\boldsymbol{Y}, w$) ▷ weighted modes move together
  label $\leftarrow$ inv[label]; $\delta \leftarrow \max_b \|\boldsymbol{Y}'[\text{inv}]_b - \boldsymbol{Y}_b\|$
  $\boldsymbol{Y}, w \leftarrow \boldsymbol{Y}', w'$
  if $\delta \le \tau$ **then**
    break
  end if
end for
$(\{\boldsymbol{y}_p\}, \text{mode}) \leftarrow \mathrm{unique}(\boldsymbol{Y})$ ▷ coincident modes merge into one EM point
return EM points $\{\boldsymbol{y}_p\}$; member sets $\mathcal{M}_p = \{v : \text{mode}[\text{label}[\text{inv}_0[v]]] = p\}$

E.2 OVERLAPPING BALLTREE

Overlapping BallTrees are built on the GPU by batched argsort (Algorithm 3); Algorithm 4 derives the two partitions. The root sorts the M rows along their most compact axis and takes the first and the last $M'/2$ rows of that order as its two children, $M' = 2^{\lceil \log_2 M \rceil}$, so the children share the $M' - M$ median rows in a thin slab normal to that axis instead of being completed with the virtual nodes of Zhdanov et al. (2025). Every lower level bisects each node along its widest axis. The duplicated rows are then removed, keeping the copy in the first child, and leaves are consecutive blocks of ℓ rows in the resulting order; only the last leaf is padded, with fewer than ℓ repeated rows, so attention runs on M rather than M' rows. Following Zhdanov et al. (2025), a second partition $\mathcal{T}_{\mathrm{B}}$ is built after reordering the rows along the primary partition $\mathcal{T}_{\mathrm{A}}$ and applying a fixed 45° rotation; alternating blocks between $\mathcal{T}_{\mathrm{A}}$ and $\mathcal{T}_{\mathrm{B}}$ lets information cross leaf boundaries. Because atom positions change with every noise sample and sampler step, the atom partitions and the mixed atom–voxel partitions are rebuilt at every evaluation of the atomic diffusion module, and once per generated sample in the atom error head, which is why construction runs on the GPU.

Algorithm 3 Overlapping BallTree leaves (BUILDLEAVES)

Require: positions $\boldsymbol{U} \in \mathbb{R}^{M\times 3}$ of EM points, atoms, or atoms and voxels; leaf size ℓ

function BUILDLEAVES($\boldsymbol{U}$, ℓ)
  $M' \leftarrow 2^{\lceil \log_2 M \rceil}$
  $d \leftarrow \arg\min_d \mathrm{spread}_d(\boldsymbol{U})$ ▷ most compact axis
  $\pi \leftarrow \mathrm{argsort}(\boldsymbol{U}_{:,d})$
  $I \leftarrow [\pi_{1:M'/2},\ \pi_{M-M'/2+1:M}]$ ▷ two rows, the root children; they share the $M' - M$ median rows
  while $|I_{\text{row}}| > \ell$ **do** ▷ I is a [rows × cols] index matrix
    $d \leftarrow \arg\max_d \mathrm{spread}_d(\boldsymbol{U}[I])$ per row
    $\pi \leftarrow$ row-wise $\mathrm{argsort}(\boldsymbol{U}[I]_{:,d})$
    $I \leftarrow$ split each row of $I[\pi]$ into two halves ▷ one batched gather and reshape
  end while
  $f_m \leftarrow$ first position of m in $\mathrm{flatten}(I)$, $m = 1, \ldots, M$ ▷ keeps the copy in the first root child
  $o \leftarrow \mathrm{argsort}(f)$ ▷ duplicates removed, $|o| = M$
  pad o with $(\ell - |o| \bmod \ell) \bmod \ell$ repeated rows ▷ fewer than ℓ; only the last leaf is padded
  return $\mathcal{T} \leftarrow \mathrm{reshape}(o, [-1, \ell])$
end function

Algorithm 4 Two Overlapping BallTree partitions (BUILDPARTITIONS)

Require: positions $\boldsymbol{U} \in \mathbb{R}^{M\times 3}$, leaf size ℓ, rotation angle $\theta = 45°$

function BUILDPARTITIONS($\boldsymbol{U}$, ℓ, θ)
  $\mathcal{T}_{\mathrm{A}} \leftarrow$ BUILDLEAVES($\boldsymbol{U}, \ell$)
  $o_{\mathrm{A}} \leftarrow \mathrm{flatten}(\mathcal{T}_{\mathrm{A}})$
  $\boldsymbol{U}' \leftarrow \boldsymbol{U}[o_{\mathrm{A}}]\, \boldsymbol{R}_\theta$ ▷ fixed 3D rotation of the reordered cloud
  $\mathcal{T}' \leftarrow$ BUILDLEAVES($\boldsymbol{U}', \ell$)
  $\mathcal{T}_{\mathrm{B}} \leftarrow o_{\mathrm{A}}[\mathcal{T}']$ ▷ compose back to original rows
  return $\mathcal{T}_{\mathrm{A}}, \mathcal{T}_{\mathrm{B}}$
end function

E.3 BALLTREE ATTENTION

BallTree attention is self-attention inside the leaves of one set (Algorithm 5): EM points in the point encoder, and atoms in the atomic diffusion module and the atom error head. Both variants gather the rows of each Overlapping BallTree leaf, attend within the leaf, and scatter the updates back, averaging over repeated indices so that each row that pads the last leaf receives one update. The point encoder uses no rotary embeddings or pair bias: coordinates enter through its initial position/Fourier features and the tree partition. Atom self-attention rotates queries and keys with 3D rotary embeddings at the current atom coordinates and adds a block-specific projection of AlphaFold-style atom-pair features. In the atom error head, these coordinates are the generated sample. EMTrunk's separate point-axis attention uses 3D rotary embeddings (Algorithm 8).

Algorithm 5 BallTree attention (BALLTREEATTN)

Require: features $\boldsymbol{H} \in \mathbb{R}^{M\times c}$, leaves $\mathcal{T} \in \{1..M\}^{L\times \ell}$, per-leaf bias $\boldsymbol{b}$; atom positions $\boldsymbol{U}$, or $\emptyset$ for EM points

function BALLTREEATTN($\boldsymbol{H}, \mathcal{T}, \boldsymbol{b}, \boldsymbol{U}$)
  $\boldsymbol{G} \leftarrow \mathrm{LN}(\boldsymbol{H})[\mathcal{T}] \in \mathbb{R}^{L\times \ell\times c}$ ▷ gather one window per leaf
  $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \leftarrow \boldsymbol{W}_Q\boldsymbol{G}, \boldsymbol{W}_K\boldsymbol{G}, \boldsymbol{W}_V\boldsymbol{G}$
  if $\boldsymbol{U} \neq \emptyset$ **then** ▷ atoms only; EM points use $\boldsymbol{b} = 0$
    $\boldsymbol{Q} \leftarrow \mathrm{RoPE}_{\boldsymbol{U}[\mathcal{T}]}(\boldsymbol{Q})$, $\boldsymbol{K} \leftarrow \mathrm{RoPE}_{\boldsymbol{U}[\mathcal{T}]}(\boldsymbol{K})$
  end if
  $\boldsymbol{O} \leftarrow \mathrm{Attn}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}; \boldsymbol{b})$ ▷ full attention inside each leaf
  $\Delta \leftarrow \boldsymbol{0}_{M\times c}$; $n \leftarrow \boldsymbol{0}_M$
  $\Delta[\mathcal{T}] \mathrel{+}= \boldsymbol{W}_O\boldsymbol{O}$; $n[\mathcal{T}] \mathrel{+}= 1$ ▷ scatter-add; repeated indices average
  $\boldsymbol{H} \leftarrow \boldsymbol{H} + \Delta/\max(n, 1)$
  return $\boldsymbol{H} \oplus \mathrm{Trans}(\boldsymbol{H})$
end function

E.4 POINT ENCODER AND EMTRUNK

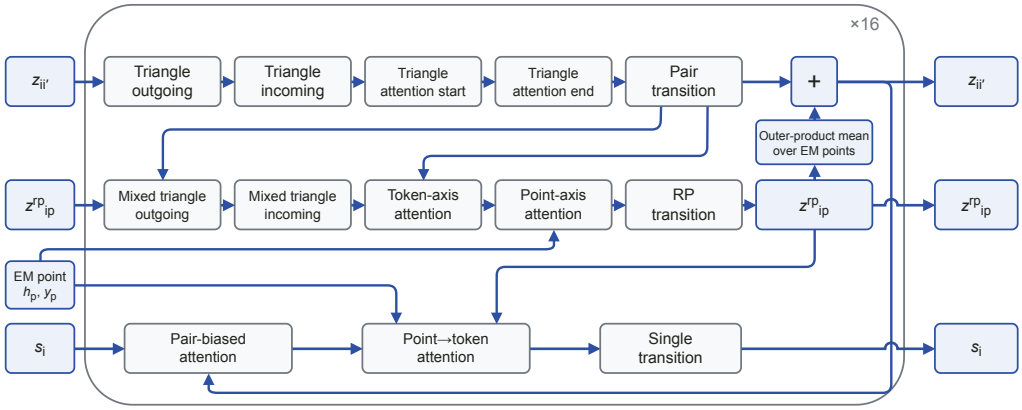

Figure 6: **EMTrunk block.** The enclosed block is repeated 16 times. The token–token pair stream applies outgoing and incoming triangle multiplication, triangle attention, and a transition. The token–point stream applies mixed triangle multiplication, token- and point-axis attention, and a transition. Its outer-product mean over EM points produces a pair update, added at + to the token–token pair. Encoded point features and EM-point positions feed point-axis attention and the point-to-token update of the single stream, where the token–point state supplies the attention bias. Cross-stream paths show where each representation is read; small arcs mark crossings without a merge. The three representations leave the block at right. Full operators are in Algorithm 8.

Algorithm 6 Point encoder

Require: EM points $\{(\boldsymbol{y}_p, \kappa_p, \bar{\boldsymbol{p}}_p, \hat{\boldsymbol{p}}_p)\}_{p=1}^{P}$
use $\boldsymbol{y}_p$ in the shared centered frame of Appendix F
$\boldsymbol{h}_p^{(0)} \leftarrow \boldsymbol{W}_2\,\mathrm{SiLU}\big(\boldsymbol{W}_1[\boldsymbol{y}_p;\ \mathrm{Fourier}(\boldsymbol{y}_p);\ \mathrm{Embed}(\kappa_p);\ \bar{\boldsymbol{p}}_p;\ \hat{\boldsymbol{p}}_p]\big)$ ▷ section 3.4
$\boldsymbol{H} \leftarrow [\boldsymbol{h}_1^{(0)}; \ldots; \boldsymbol{h}_P^{(0)}]$
$\mathcal{T}_\mathrm{A}, \mathcal{T}_\mathrm{B} \leftarrow$ BUILDPARTITIONS($\{\boldsymbol{y}_p\},\ \ell = 256,\ 45°$)
for $k = 1, \ldots, 4$ **do**
  $\boldsymbol{H} \leftarrow$ BALLTREEATTN($\boldsymbol{H},\ \mathcal{T}_\mathrm{A}$ if k odd else $\mathcal{T}_\mathrm{B},\ 0,\ \emptyset$) ▷ no RoPE
end for
return $\{\boldsymbol{h}_p\} \leftarrow$ rows of $\boldsymbol{H}$, $\{\boldsymbol{y}_p\}$

Algorithm 7 EMTrunk forward

Require: token inputs $\boldsymbol{s}_i^{\text{in}}$ (input embedder), MSA features msa, relative-position and bond features, encoded point features and positions $\{\boldsymbol{h}_p, \boldsymbol{y}_p\}$

$\boldsymbol{s}_i^{(0)} \leftarrow \boldsymbol{W}_s \boldsymbol{s}_i^{\text{in}}$; $\boldsymbol{z}_{ii'}^{(0)} \leftarrow \boldsymbol{W}_{z1} \boldsymbol{s}_i^{(0)} + \boldsymbol{W}_{z2} \boldsymbol{s}_{i'}^{(0)} + \mathrm{RelPos}(i, i') + \mathrm{Bond}(i, i')$ ▷ AlphaFold-style

$\boldsymbol{z}_{ip}^{\text{rp},(0)} \leftarrow \boldsymbol{W}_{\text{r}} \boldsymbol{s}_i^{(0)} + \boldsymbol{W}_{\text{p}} \boldsymbol{h}_p$ ▷ token–point initialization

$b_{pp'}^{\text{pp}} \leftarrow \boldsymbol{W}_{\text{rbf}} \mathrm{RBF}_{32}(\|\boldsymbol{y}_p - \boldsymbol{y}_{p'}\|) + g \odot \langle \boldsymbol{W}_q \boldsymbol{h}_p, \boldsymbol{W}_k \boldsymbol{h}_{p'} \rangle / \sqrt{8}$ ▷ per head; gate g zero-initialized

$\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{z}^{\text{rp}} \leftarrow \boldsymbol{0}$

for each recycling cycle (one in our runs) **do**

  $\boldsymbol{z} \leftarrow \boldsymbol{z}^{(0)} + \boldsymbol{W}\,\mathrm{LN}(\boldsymbol{z})$; $\boldsymbol{z} \leftarrow \mathrm{MSAModule}(\boldsymbol{z}, \mathrm{msa}, \boldsymbol{s}^{\text{in}})$ ▷ protein MSAs; pretrained

  $\boldsymbol{s} \leftarrow \boldsymbol{s}^{(0)} + \boldsymbol{W}\,\mathrm{LN}(\boldsymbol{s})$; $\boldsymbol{z}^{\text{rp}} \leftarrow \boldsymbol{z}^{\text{rp},(0)} + \boldsymbol{W}\,\mathrm{LN}(\boldsymbol{z}^{\text{rp}})$

  for $k = 1, \ldots, 16$ **do**

    $\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{z}^{\text{rp}} \leftarrow \text{EMTrunkBlock}(\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{z}^{\text{rp}}, \{\boldsymbol{h}_p, \boldsymbol{y}_p\}, b^{\text{pp}})$ ▷ Algorithm 8

  end for

end for

distograms $\boldsymbol{W}_{\text{rr}} \boldsymbol{z}_{ii'}$, $\boldsymbol{W}_{\text{rp}} \boldsymbol{z}_{ip}^{\text{rp}}$; coordinates $\boldsymbol{W}_{\text{res}} \boldsymbol{s}_i$, $\boldsymbol{W}_{\text{pt}} \boldsymbol{h}_p$ ▷ heads; section 3.5

return $\boldsymbol{s}^{\text{in}}, \boldsymbol{s}, \boldsymbol{z}, \{\boldsymbol{h}_p, \boldsymbol{y}_p\}, \boldsymbol{z}^{\text{rp}}$

Algorithm 8 One EMTrunk block (EMTrunkBlock); AlphaFold-style operators are marked †

Require: $\boldsymbol{s}_i$, $\boldsymbol{z}_{ii'}$, $\boldsymbol{z}_{ip}^{\text{rp}}$, encoded point features and positions $\{\boldsymbol{h}_p, \boldsymbol{y}_p\}$, point–point bias b^{pp}

$\boldsymbol{z} \leftarrow \boldsymbol{z} \oplus \mathrm{TriMulOut}^{\dagger}(\boldsymbol{z}) \oplus \mathrm{TriMulIn}^{\dagger}(\boldsymbol{z}) \oplus \mathrm{TriAttStart}^{\dagger}(\boldsymbol{z}) \oplus \mathrm{TriAttEnd}^{\dagger}(\boldsymbol{z}) \oplus \mathrm{Trans}^{\dagger}(\boldsymbol{z})$

$\boldsymbol{z}_{ip}^{\text{rp}} \leftarrow \boldsymbol{z}_{ip}^{\text{rp}} \oplus \sum_{i'} g(\boldsymbol{z}_{ii'}) \odot g'(\boldsymbol{z}_{i'p}^{\text{rp}})$ ▷ mixed triangle multiplication, outgoing

$\boldsymbol{z}_{ip}^{\text{rp}} \leftarrow \boldsymbol{z}_{ip}^{\text{rp}} \oplus \sum_{i'} g(\boldsymbol{z}_{i'i}) \odot g'(\boldsymbol{z}_{i'p}^{\text{rp}})$ ▷ incoming

$\boldsymbol{z}_{:p}^{\text{rp}} \leftarrow \boldsymbol{z}_{:p}^{\text{rp}} \oplus \mathrm{Attn}_i(\boldsymbol{z}_{:p}^{\text{rp}};\ b = \boldsymbol{W}_b \boldsymbol{z}_{ii'})$ ▷ token-axis attention, per point p

$\boldsymbol{z}_{i:}^{\text{rp}} \leftarrow \boldsymbol{z}_{i:}^{\text{rp}} \oplus \mathrm{Attn}_p(\mathrm{RoPE}_{\boldsymbol{y}}(\boldsymbol{z}_{i:}^{\text{rp}});\ b = b^{\text{pp}})$ ▷ point-axis attention, per token i, global over p

$\boldsymbol{z}^{\text{rp}} \leftarrow \boldsymbol{z}^{\text{rp}} \oplus \mathrm{Trans}(\boldsymbol{z}^{\text{rp}})$

$\boldsymbol{z}_{ii'} \leftarrow \boldsymbol{z}_{ii'} \oplus \mathrm{OPM}_p(\boldsymbol{z}_{ip}^{\text{rp}}, \boldsymbol{z}_{i'p}^{\text{rp}})$ ▷ outer-product mean over points: token–point writes back to token–token

$\boldsymbol{s}_i \leftarrow \boldsymbol{s}_i \oplus \mathrm{AttnPairBias}^{\dagger}(\boldsymbol{s};\ b = \boldsymbol{W}\boldsymbol{z}_{ii'})$

$\boldsymbol{s}_i \leftarrow \boldsymbol{s}_i \oplus \mathrm{Attn}_p\big(q = \boldsymbol{s}_i,\ k = \boldsymbol{h}_p,\ v = \boldsymbol{h}_p + \boldsymbol{W}_y \boldsymbol{y}_p;\ b = \boldsymbol{W}\boldsymbol{z}_{ip}^{\text{rp}}\big)$ ▷ point-to-token attention

$\boldsymbol{s} \leftarrow \boldsymbol{s} \oplus \mathrm{Trans}^{\dagger}(\boldsymbol{s})$

return $\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{z}^{\text{rp}}$

E.5 HETEROGENEOUS BALLTREE ATTENTION

Heterogeneous BallTree Attention is voxel-to-atom cross-attention on mixed Overlapping BallTree leaves of $\ell_{av} = 256$ rows that contain both atoms and active voxels (Algorithm 9, Figure 2): atoms query voxels. BallTree attention is never applied to the mixed partition. Active voxels far outnumber modeled atoms ($V \gg A$ on the 1 Å grid), so only the atom-query, voxel-key quadrant is computed. Updates are scattered back to the atoms as in BallTree attention. In the calls below, $\boldsymbol{U}$ collects the voxel positions $\boldsymbol{u}_v$; atom positions are the current noisy coordinates $\boldsymbol{X}'$ or the generated coordinates $\boldsymbol{X}^{\text{gen}}$.

Algorithm 9 Heterogeneous BallTree Attention (HETBALLTREE): atom queries, voxel keys/values

Require: atom states $\boldsymbol{a}_a$, positions $\boldsymbol{x}'_a$; semantic voxel representation $\boldsymbol{e}_v$, positions $\boldsymbol{u}_v$; mixed leaves $\mathcal{T}^{av}$

function HETBALLTREE($\boldsymbol{a}$, $\boldsymbol{e}$, $\mathcal{T}^{av}$, $\boldsymbol{X}'$, $\boldsymbol{U}$) ▷ $\boldsymbol{U}$ holds voxel positions $\boldsymbol{u}_v$
  $\boldsymbol{K}_v, \boldsymbol{V}_v \leftarrow \boldsymbol{W}_K \mathrm{LN}(\boldsymbol{e}_v),\ \boldsymbol{W}_V \mathrm{LN}(\boldsymbol{e}_v) + \boldsymbol{W}_u \boldsymbol{u}_v$ ▷ projected once per map, reused at every noise sample and sampler step
  $\mathcal{T}^{av} \leftarrow$ sort within each leaf to group atoms before voxels: $\mathcal{T} = [\mathcal{A}_\mathcal{T}, \mathcal{V}_\mathcal{T}]$
  for each mixed leaf $\mathcal{T}$ **do** ▷ atom rows $\mathcal{A}_\mathcal{T}$, voxel rows $\mathcal{V}_\mathcal{T}$; batched over leaves
    $\boldsymbol{Q}_a \leftarrow \mathrm{RoPE}_{\boldsymbol{x}'_a}(\boldsymbol{W}_Q \mathrm{LN}(\boldsymbol{a}_a)), a \in \mathcal{A}_\mathcal{T}$; $\boldsymbol{K}_v \leftarrow \mathrm{RoPE}_{\boldsymbol{u}_v}(\boldsymbol{K}_v), v \in \mathcal{V}_\mathcal{T}$
    $b_{av} \leftarrow \boldsymbol{W}_b \,\mathrm{RBF}_{32}(\|\boldsymbol{x}'_a - \boldsymbol{u}_v\|;\ 0\text{–}32\,\text{Å})$
    $\Delta_a \leftarrow \boldsymbol{W}_O \,\mathrm{Attn}(\boldsymbol{Q}_a, \boldsymbol{K}_{\mathcal{V}_\mathcal{T}}, \boldsymbol{V}_{\mathcal{V}_\mathcal{T}};\ b_{a\cdot})$ ▷ atoms query, voxels are keys/values only
  end for
  $\boldsymbol{a} \leftarrow \boldsymbol{a} + \mathrm{scatter_mean}(\Delta)$ ▷ scatter as in BallTree attention, Algorithm 5
  return $\boldsymbol{a} \oplus \mathrm{Trans}(\boldsymbol{a})$
end function

E.6 ATOMIC DIFFUSION MODULE

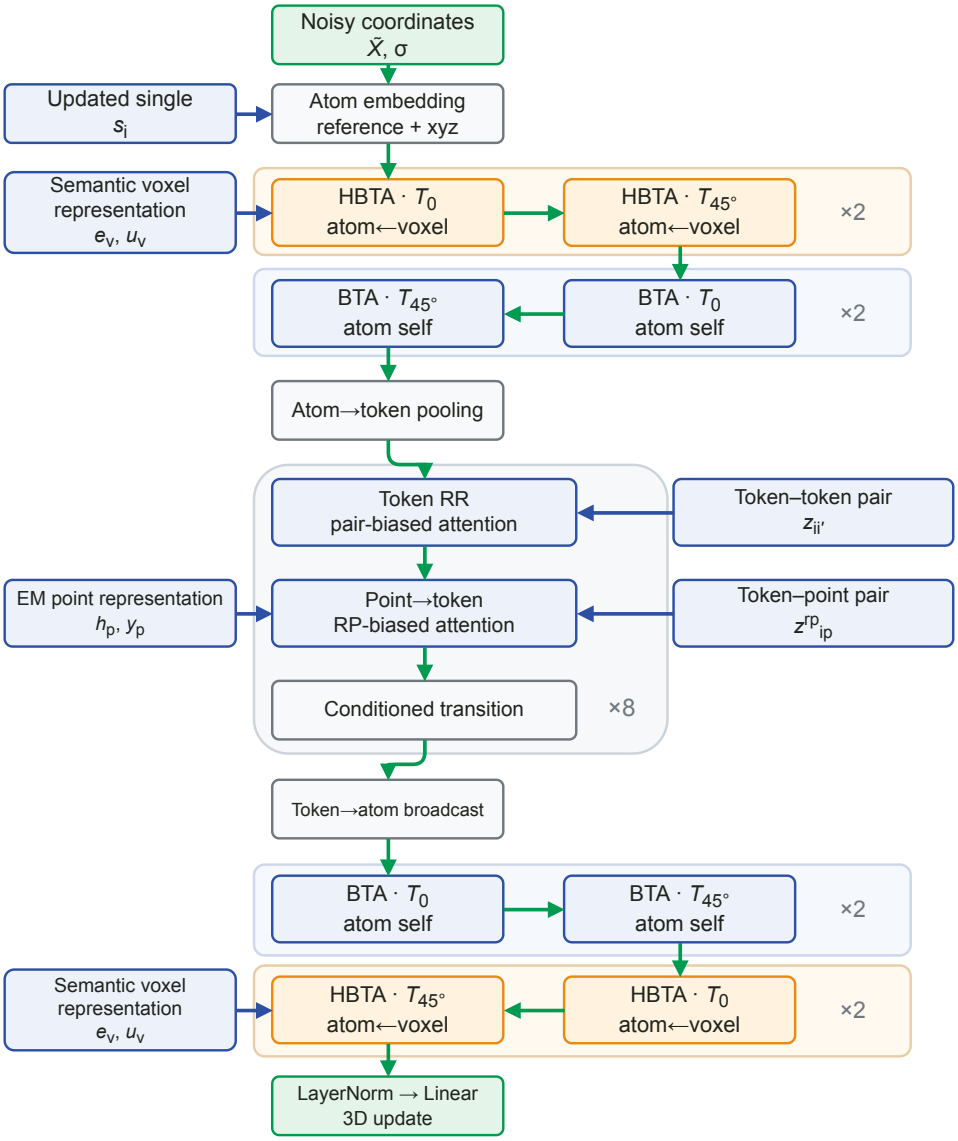

Figure 7: **Atomic diffusion module.** The connected path runs from noisy coordinates through the atom encoder and token stack to the atom decoder and coordinate update. Pale enclosures contain the operations repeated by each $\times 2$ or $\times 8$ label. **Top:** The encoder first reads the semantic voxel representation with Heterogeneous BallTree Attention (HBTA; atom queries, voxel keys and values), then applies atom self-attention with BallTree attention (BTA). The primary (T_0; $\mathcal{T}_A$ in Algorithm 10) and $45°$-rotated ($T_{45°}$; $\mathcal{T}_B$) Overlapping BallTree partitions alternate, and each pair repeats twice, yielding four blocks of each kind. **Middle:** Each of eight token blocks applies token–token pair-biased attention, point-to-token attention with token–point pair bias, and a conditioned transition. **Bottom:** The decoder applies four BTA blocks followed by four HBTA blocks, reusing the same semantic voxel representation. The 3D output head is LayerNorm followed by a bias-free linear projection. Blue marks representations and atom self-attention; orange marks voxel-to-atom attention; green marks the path through the atomic diffusion module.

Given noisy coordinates $\tilde{\boldsymbol{X}} = \boldsymbol{X} + \sigma\boldsymbol{\epsilon}$, the preconditioned denoiser (Karras et al., 2022) is

$$D_\theta(\tilde{\boldsymbol{X}}, \sigma) = \frac{\tilde{\boldsymbol{X}}}{1 + \sigma^2/\sigma_{\text{data}}^2} + \frac{\sigma}{\sqrt{1 + \sigma^2/\sigma_{\text{data}}^2}}\, F_\theta\Big(\frac{\tilde{\boldsymbol{X}}}{\sqrt{\sigma^2 + \sigma_{\text{data}}^2}}, \sigma\Big), \tag{2}$$

The training loss uses

$$\mathcal{L} = \mathcal{L}_{\text{trunk}} + \tfrac{1}{3}\,\mathcal{L}_{\text{mse}} + \mathcal{L}_{\text{lddt}}, \qquad \mathcal{L}_{\text{mse}} = \frac{\sigma^2 + \sigma_{\text{data}}^2}{\sigma^2\sigma_{\text{data}}^2}\;\frac{1}{A}\sum_a w_a\, \|D_\theta(\tilde{\boldsymbol{X}}, \sigma)_a - \boldsymbol{x}_a\|^2, \tag{3}$$

Algorithm 10 is the raw network F_θ of Eq. 2; Algorithm 9 is Heterogeneous BallTree Attention and Algorithm 11 the token block; Algorithm 12 is the sampler. The point encoder, the voxel encoder, and EMTrunk run once per map; F_θ, including the construction of its Overlapping BallTrees, runs once per noise sample and sampler step.

Algorithm 10 Raw network F_θ of the atomic diffusion module

Require: scaled noisy coordinates $\hat{\boldsymbol{X}} = \tilde{\boldsymbol{X}}/\sqrt{\sigma^2 + \sigma_{\text{data}}^2}$, σ, EMTrunk outputs $(\boldsymbol{s}^{\text{in}}, \boldsymbol{s}, \boldsymbol{z}, \{\boldsymbol{h}_p, \boldsymbol{y}_p\}, \boldsymbol{z}^{\text{rp}})$, semantic voxel representation $\{\boldsymbol{e}_v\}$ at positions $\{\boldsymbol{u}_v\}$

$\boldsymbol{X}' \leftarrow \hat{\boldsymbol{X}}\sqrt{\sigma^2 + \sigma_{\text{data}}^2}$ ▷ coordinates in angstroms for geometry

$\boldsymbol{s}_i^\sigma \leftarrow \text{MLP}([\boldsymbol{s}_i; \boldsymbol{s}_i^{\text{in}}]) + \text{Fourier}_{256}\big(\frac{1}{4}\log(\sigma/\sigma_{\text{data}})\big)$ ▷ noise-conditioned single

$\boldsymbol{z}^\sigma \leftarrow \text{Trans}(\text{Trans}(\boldsymbol{W}\,\text{LN}(\boldsymbol{z})))$ ▷ conditioned pair

$\mathcal{T}_{\text{A}}^a, \mathcal{T}_{\text{B}}^a \leftarrow$ BUILDPARTITIONS$(\{\boldsymbol{x}'_a\},\ \ell_a = 128,\ 45°)$ ▷ atom trees, BallTree attention

$\mathcal{T}_{\text{A}}^{av}, \mathcal{T}_{\text{B}}^{av} \leftarrow$ BUILDPARTITIONS$(\{\boldsymbol{x}'_a\} \cup \{\boldsymbol{u}_v\},\ \ell_{av} = 256,\ 45°)$; drop leaves without atoms ▷ mixed Overlapping BallTrees

$\boldsymbol{a}_a \leftarrow \text{RefEmbed}^\dagger(a) + \boldsymbol{W}\boldsymbol{s}^\sigma_{\text{tok}(a)} + \boldsymbol{W}_x\hat{\boldsymbol{x}}_a + \boldsymbol{W}_f[\hat{\boldsymbol{x}}_a; \text{Fourier}(\boldsymbol{x}'_a)]$ ▷ atom state, $c_a = 128$

$\boldsymbol{f}_j^{aa'} \leftarrow \text{MLP}(\text{ref. displacement},\ \text{identity},\ \boldsymbol{z}^\sigma_{\text{tok}(a)\text{tok}(a')})$ for leaves of $\mathcal{T}_j^a$, $j \in \{\text{A}, \text{B}\}$ ▷ shared pair features

$j_k \leftarrow$ A for odd k, B for even k ▷ alternate the two partitions

for $k = 1, \ldots, 4$ **do** ▷ encoder: Heterogeneous BallTree Attention first

  $\boldsymbol{a} \leftarrow$ HETBALLTREE$(\boldsymbol{a}, \boldsymbol{e}, \mathcal{T}_{j_k}^{av}, \boldsymbol{X}', \boldsymbol{U})$

end for

for $k = 1, \ldots, 4$ **do** ▷ then BallTree attention

  $b_k^{aa'} \leftarrow \boldsymbol{W}_{b,k}^{\text{enc}}\text{LN}_k^{\text{enc}}(\boldsymbol{f}_{j_k}^{aa'})$ ▷ per-block, per-head bias

  $\boldsymbol{a} \leftarrow$ BALLTREEATTN$(\boldsymbol{a}, \mathcal{T}_{j_k}^a,\ b_k^{aa'}, \boldsymbol{X}')$

end for

$\boldsymbol{t}_i \leftarrow \text{mean}_{a \in i}\text{ReLU}(\boldsymbol{W}\boldsymbol{a}_a) + \boldsymbol{W}\,\text{LN}(\boldsymbol{s}_i^\sigma)$; $\boldsymbol{x}_i^{\text{tok}} \leftarrow \boldsymbol{x}'_{\text{rep}(i)}$ ▷ tokens, $c_t = 768$; disto-atom positions

for $k = 1, \ldots, 8$ **do**

  $\boldsymbol{t} \leftarrow$ TOKENBLOCK$(\boldsymbol{t}, \boldsymbol{s}^\sigma, \boldsymbol{z}^\sigma, \{\boldsymbol{h}_p, \boldsymbol{y}_p\}, \boldsymbol{z}^{\text{rp}}, \boldsymbol{x}^{\text{tok}})$

end for

$\boldsymbol{a}_a \leftarrow \boldsymbol{a}_a + \boldsymbol{W}\,\text{LN}(\boldsymbol{t}_{\text{tok}(a)})$ ▷ broadcast tokens to atoms

for $k = 1, \ldots, 4$ **do** ▷ decoder: BallTree attention first

  $b_k^{aa'} \leftarrow \boldsymbol{W}_{b,k}^{\text{dec}}\text{LN}_k^{\text{dec}}(\boldsymbol{f}_{j_k}^{aa'})$

  $\boldsymbol{a} \leftarrow$ BALLTREEATTN$(\boldsymbol{a}, \mathcal{T}_{j_k}^a,\ b_k^{aa'}, \boldsymbol{X}')$

end for

for $k = 1, \ldots, 4$ **do** ▷ then Heterogeneous BallTree Attention

  $\boldsymbol{a} \leftarrow$ HETBALLTREE$(\boldsymbol{a}, \boldsymbol{e}, \mathcal{T}_{j_k}^{av}, \boldsymbol{X}', \boldsymbol{U})$

end for

return $F_\theta \leftarrow \boldsymbol{W}_{\text{out}}\,\text{LN}(\boldsymbol{a}) \in \mathbb{R}^{A\times 3}$ ▷ FP32

Algorithm 11 Token block (TOKENBLOCK)

Require: tokens $\boldsymbol{t}_i$, conditioned single $\boldsymbol{s}_i^\sigma$, conditioned pair $\boldsymbol{z}_{ii'}^\sigma$, encoded point features and positions $\{\boldsymbol{h}_p, \boldsymbol{y}_p\}$, $\boldsymbol{z}_{ip}^{\mathrm{rp}}$, token positions $\boldsymbol{x}_i^{\mathrm{tok}}$

1: $\boldsymbol{N} \leftarrow \mathrm{AdaLN}(\boldsymbol{t}; \boldsymbol{s}^\sigma)$; $\Delta \leftarrow \mathrm{Attn}(\boldsymbol{N}, \boldsymbol{N}, \boldsymbol{N};\ b = \boldsymbol{W}_b \boldsymbol{z}_{ii'}^\sigma)$ ▷ token–token attention, 16 heads

2: $\boldsymbol{t} \leftarrow \boldsymbol{t} + \mathrm{sigmoid}(\boldsymbol{W}_g \boldsymbol{s}^\sigma) \odot \Delta$

3: $\boldsymbol{t}_i \leftarrow \boldsymbol{t}_i \oplus \mathrm{Attn}_p\big(q = \mathrm{RoPE}_{\boldsymbol{x}_i^{\mathrm{tok}}}(\boldsymbol{W}_Q \boldsymbol{t}_i),\ k = \mathrm{RoPE}_{\boldsymbol{y}_p}(\boldsymbol{W}_K \boldsymbol{h}_p),\ v = \boldsymbol{W}_V \boldsymbol{h}_p;\ b = \boldsymbol{W} \boldsymbol{z}_{ip}^{\mathrm{rp}}\big)$ ▷ point-to-token attention

4: **return** $\boldsymbol{t} \oplus \mathrm{CondTrans}(\boldsymbol{t}; \boldsymbol{s}^\sigma)$

E.7 INFERENCE AND SAMPLING

Inference retains all input tokens, all predicted EM points, and all active voxels for each assembly. The input encoders and EMTrunk are evaluated once, and their outputs are reused across the five independently initialized coordinate samples. Each sample uses five Euler steps with the Karras schedule in Algorithm 12, no churn ($\gamma_0 = 0$), and step scale $\eta = 1$. The sampling random-number generator uses seed 101. Coordinates start from isotropic Gaussian noise in the shared EM-point-centered frame of Appendix F, with no starting structure. The generated coordinates are returned to the original map frame, and per-sample metrics are averaged without reference-based selection.

Algorithm 12 Sampling in the map frame

Require: trained D_θ (Eq. 2), EMTrunk outputs and encoded point and voxel features, steps $N = 5$, $\eta = 1$

1: $\sigma_n \leftarrow \sigma_{\mathrm{data}}\big(160^{1/7} + \frac{n}{N}\big((4 \times 10^{-4})^{1/7} - 160^{1/7}\big)\big)^7, n = 0, \ldots, N$; $\sigma_N \leftarrow 0$ ▷ Karras schedule

2: $\boldsymbol{X} \leftarrow \sigma_0\, \boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ ▷ centered at the EM-point centroid

3: **for** $n = 0, \ldots, N-1$ **do**

4: $\quad \bar{\boldsymbol{X}} \leftarrow D_\theta(\boldsymbol{X}, \sigma_n)$ ▷ no churn: $\gamma_0 = 0$

5: $\quad \boldsymbol{X} \leftarrow \boldsymbol{X} + \eta\,(\sigma_{n+1} - \sigma_n)\,(\boldsymbol{X} - \bar{\boldsymbol{X}})/\sigma_n$ ▷ Euler step, FP32

6: **end for**

7: **return** $\boldsymbol{X}+$ EM-point centroid ▷ map frame; no alignment

E.8 ATOM ERROR HEAD

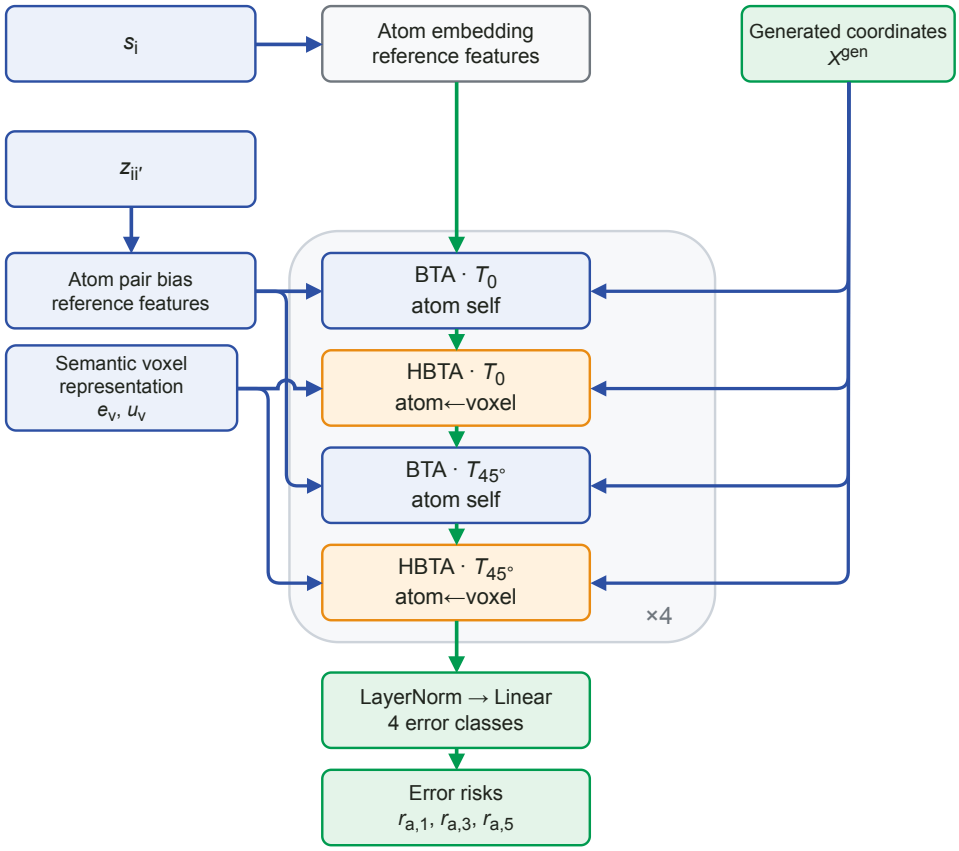

Figure 8: **Atom error head.** The head scores coordinates generated by the frozen atomic diffusion module. The single representation $\boldsymbol{s}_i$ and reference features feed the frozen atom embedding, followed by a trainable input projection. The pair representation $\boldsymbol{z}_{ii'}$ and reference features supply frozen atom-pair features; the atom-pair bias path includes an independent trainable projection in each BallTree attention (BTA) block. Generated coordinates supply the partitions, 3D rotary embeddings, and atom–voxel distances for BTA and Heterogeneous BallTree Attention (HBTA); T_0 and $T_{45°}$ indicate the alternating partitions $\mathcal{T}_\mathrm{A}$ and $\mathcal{T}_\mathrm{B}$ of Algorithm 13. Each of eight blocks applies BTA then HBTA; the enclosed two-partition sequence repeats four times. Pair features and the semantic voxel representation are reused across blocks. LayerNorm and a linear projection give the probabilities of four map-frame error classes, summed into $r_{a,1}$, $r_{a,3}$, and $r_{a,5}$. Colors follow Figure 7.

Training data. The atom error head and the AlphaFold-style confidence head are trained last, with every earlier stage frozen: the map encoder, the point and voxel encoders, EMTrunk, and the atomic diffusion module. For each training crop, the frozen sampler draws two samples $\boldsymbol{X}^{\mathrm{gen}}$ from noise with fresh seeds, no reference input, and five Euler steps, using the same sampler step count as evaluation (Algorithm 12). Each sample is matched to the deposited structure with the chain and atom-symmetry permutation used for the diffusion loss, without rigid alignment. Both heads are trained with AdamW (Table 7).

AlphaFold-style confidence head. The four-block confidence head of the AlphaFold-style model is initialized from the pretrained weights of the MSA version of Protenix-Mini (Gong et al., 2025)

and fine-tuned on the generated samples with its native losses. It reads the EMTrunk single and pair representations together with $\boldsymbol{X}^{\mathrm{gen}}$ and predicts pLDDT, PAE, PDE, and a per-atom resolved probability (Abramson et al., 2024); it sees the map only through EMTrunk.

Error classes and objective. A linear layer maps each atom state to probabilities $\pi_{a,k}$ over four classes $k = 0, \ldots, 3$ of the map-frame error:

$$e_a = \|\boldsymbol{x}_a^{\mathrm{gen}} - \boldsymbol{x}_a\| \in [0,1),\ [1,3),\ [3,5),\ [5,\infty)\ \text{Å},$$
$$r_{a,1} = 1 - \pi_{a,0}, \qquad r_{a,3} = \pi_{a,2} + \pi_{a,3}, \qquad r_{a,5} = \pi_{a,3}, \tag{4}$$

where $\boldsymbol{x}_a$ is the matched deposited atom; summing classes makes the three risks monotone by construction. $\mathcal{L}_{\mathrm{err}}$ is the cross-entropy of the four classes over matched, resolved atoms; unmatched, unresolved, and padding atoms do not contribute. The confidence objective $\mathcal{L}_{\mathrm{conf}}$ adds to $\mathcal{L}_{\mathrm{err}}$ the equally weighted mean of the available AlphaFold-style losses (pLDDT, PDE, resolved, and PAE on tokens with valid frames), averaged over samples. The outputs of the two heads are reported separately, and cross-entropy training alone does not make $r_{a,t}$ a calibrated probability.

Algorithm. Algorithm 13 scores one generated sample. It reuses the frozen reference-embedding and pair-feature builders of the atomic diffusion module, conditioned on the EMTrunk single and pair representations. The coordinate input to the reference embedding is zero; the head does not add the noisy-coordinate or Fourier-coordinate embeddings of the atomic diffusion module. Generated coordinates instead determine the tree partitions, 3D rotary embeddings, and atom–voxel distance biases. The partitions and frozen pair features for both partitions are computed once per sample and reused across the eight blocks. The input projection, all attention blocks (including each block's pair normalization and bias projection), and the output classifier are independently trainable.

Algorithm 13 Atom error head

Require: generated coordinates $\{\boldsymbol{x}_a^{\mathrm{gen}}\}$ in Å; EMTrunk single and pair $(\boldsymbol{s}, \boldsymbol{z})$; atom reference features; semantic voxel representation $\{\boldsymbol{e}_v, \boldsymbol{u}_v\}$

$\mathcal{T}_{\mathrm{A}}^a, \mathcal{T}_{\mathrm{B}}^a \leftarrow$ BUILDPARTITIONS$(\{\boldsymbol{x}_a^{\mathrm{gen}}\},\ \ell_a = 128,\ 45^\circ)$ ▷ atom trees, BallTree attention

$\mathcal{T}_{\mathrm{A}}^{av}, \mathcal{T}_{\mathrm{B}}^{av} \leftarrow$ BUILDPARTITIONS$(\{\boldsymbol{x}_a^{\mathrm{gen}}\} \cup \{\boldsymbol{u}_v\},\ \ell_{av} = 256,\ 45^\circ)$; drop leaves without atoms ▷ mixed Overlapping BallTrees

$\boldsymbol{h}_a^{\mathrm{ref}} \leftarrow \mathrm{RefEmbed}_{\mathrm{frozen}}(a) + \boldsymbol{W}_{\mathrm{frozen}} \mathrm{LN}_{\mathrm{frozen}}(\boldsymbol{s}_{\mathrm{tok}(a)})$ ▷ coordinate input is zero

$\boldsymbol{a}_a \leftarrow \boldsymbol{W}_{\mathrm{in}} \boldsymbol{h}_a^{\mathrm{ref}}$ ▷ trainable input projection

$\boldsymbol{f}_j^{aa'} \leftarrow$ frozen pair features from reference features and $\boldsymbol{z}$, for leaves of $\mathcal{T}_j^a$, $j \in \{\mathrm{A}, \mathrm{B}\}$

for $k = 1, \ldots, 8$ **do** ▷ independent weights per block

  $j \leftarrow$ A for odd k, B for even k ▷ same partition for both attentions

  $b_k^{aa'} \leftarrow \boldsymbol{W}_{b,k} \mathrm{LN}_k(\boldsymbol{f}_j^{aa'})$ ▷ trainable per-block, per-head bias

  $\boldsymbol{a} \leftarrow$ BALLTREEATTN$(\boldsymbol{a}, \mathcal{T}_j^a,\ b_k^{aa'}, \boldsymbol{X}^{\mathrm{gen}})$

  $\boldsymbol{a} \leftarrow$ HETBALLTREE$(\boldsymbol{a}, \boldsymbol{e}, \mathcal{T}_j^{av}, \boldsymbol{X}^{\mathrm{gen}}, \boldsymbol{U})$ ▷ atoms query voxels

end for

$(\pi_{a,0}, \ldots, \pi_{a,3}) \leftarrow \mathrm{softmax}\big(\boldsymbol{W}_{\mathrm{out}}\, \mathrm{LN}(\boldsymbol{a}_a)\big)$ ▷ classes $[0,1), [1,3), [3,5), [5,\infty)$ Å

return $r_{a,1} = 1 - \pi_{a,0}, r_{a,3} = \pi_{a,2} + \pi_{a,3}, r_{a,5} = \pi_{a,3}$ ▷ Eq. 4

F TRAINING SET AND INPUT PREPARATION

Training set. The training set consists of paired cryo-EM maps and deposited atomic structures released through 2025. Entries have a reported resolution of 4 Å or better, a one-to-one PDB–EMDB pairing, EMDB cell edges no longer than 500 Å, a resolved polymer fraction above 0.8 in biological assembly 1, and no unknown (UNK) polymer or ligand components. Deposited structures supply the segmentation labels and coordinate targets; subsequent stages consume predictions from the preceding stages. The evaluation set is prepared separately from later releases, with sequence-similarity exclusion against both this training set and Cryo2StructData (Appendix I.1).

Stage-specific training eligibility. Training examples for the sparse U-Nets are retained when the predicted macromolecule mask has intersection-over-union above 0.5 and recall at least 0.6 against

its reference mask. For atomic diffusion training, an additional quality filter retains maps whose total number of protein, nucleic-acid, and ligand EM points divided by the number of resolved polymer disto atoms within the preprocessed map crop lies in $[0.7, 1.2]$. Against these crop-clipped targets, protein disto-atom recall within 3 Å must be at least 0.75 when protein is present, and nucleic-acid disto-atom recall must be at least 0.60 when nucleic acid is present. These reference-dependent checks select training examples; they do not determine the map representation at inference or select targets from the separately prepared evaluation set.

Map resampling and normalization. Maps are resampled to an isotropic 1 Å grid by trilinear interpolation with zero-valued sampling outside the original grid, preserving the physical map origin. Negative values are set to zero. Let m be the minimum of this nonnegative grid and $q_{0.98}$ the 98th percentile of its strictly positive values. The normalized density is $(\rho - m)/(q_{0.98} - m)$; values above $q_{0.98}$ are not clipped. The author-recommended contour is transformed to the same intensity scale. Spatial cropping uses the bounding box of denoised contour support, extended by 8 Å, with a minimum extent of 64 voxels per axis. If denoising removes all support, the raw contour support is used; an empty support is rejected. This procedure uses the map and its contour, without reference coordinates.

Macromolecule mask and active voxels. The contour-cropped map is fitted into a 128^3 cube while preserving its aspect ratio. The macromolecule network receives two channels: normalized density with values below the author contour set to zero, and the unfiltered normalized density. Its predicted mask is mapped back to the 1 Å grid, smoothed with a Gaussian of $\sigma = 3$ Å, and Li-thresholded. The sparse U-Nets read the normalized density on these macromolecule-gated voxels. Per-channel Li thresholds select the active-voxel union, while the downstream encoders retain the continuous probabilities. Whole-map inference processes this complete predicted support; the contour crop removes surrounding map volume rather than dividing the assembly into sliding windows.

Protein MSAs and templates. Protein MSA features are read from paired and unpaired A3M alignments indexed by the exact chain sequence, so identical sequences reuse the same alignment archive. Species identifiers in the paired alignments support pairing across chains. The input embedder and MSA module inherited from the pretrained AlphaFold-style model are fine-tuned jointly with EMTrunk. RNA MSA features are disabled, and templates are disabled for all molecule types. The evaluation sequences use MSA sequence databases released before the evaluation window (section 4.1).

Shared coordinate frame. Grid indices are ordered as (d, h, w), whereas positions are stored as (x, y, z) in angstroms. The centroid of the selected predicted EM points defines a common translation for the EM points, active voxels, and atomic training targets. A single random rotation is applied to all three during training; inference uses the identity rotation. The atomic diffusion module therefore operates in one shared centered frame, and the inverse rotation and translation restore its predictions to the original map frame before export and evaluation. Reference coordinates are used for training crops and supervision, but do not define the inference translation or conditioning.

G HYPERPARAMETERS

Tables 4–7 summarize the architecture and optimization settings. Symbols follow section 3: R tokens, P EM points, A atoms, V active voxels, S noise samples, N sampler steps.

Training procedure. The macromolecule network is trained from scratch on the 128^3 map inputs. The three sparse U-Nets are then trained from scratch in BF16 on predicted macromolecule-gated voxels with a per-rank voxel budget; EM points and cluster summaries are then computed for every training map with the trained sparse U-Nets. EMTrunk starts from the released weights of the MSA version of Protenix-Mini (Gong et al., 2025) (`protenix_mini_default_v0.5.0`, `https://protenix.tos-cn-beijing.volces.com/checkpoint/protenix_mini_default_v0.5.0.pt`), a compact model of the Protenix family (ByteDance AML AI4Science Team et al., 2025) (`https://github.com/bytedance/Protenix`); its official checkpoint documentation reports a training-data cutoff of 30 September 2021 (`https://github.com/bytedance/Protenix/blob/main/`

`docs/supported_models.md`). Its input embedder, one-block MSA module, and Pairformer read protein MSAs, with templates disabled, and are fine-tuned together with the EM additions; all EM additions are freshly initialized, and EMTrunk is trained on these predicted EM points. The atomic diffusion module starts from the trained EMTrunk and is trained jointly with it, with voxels restricted to those near the crop (Table 6). Pretrained Pairformer parameters and new EM parameters use separate learning-rate groups under the hybrid Muon/AdamW optimizer. A shared random rotation is applied to structure, EM points, and voxels during training. Attention and point encoding run in BF16, while EDM preconditioning, the coordinate head, and the losses stay in FP32. Kernels are listed in Appendix H.

Table 4: Macromolecule network and sparse U-Nets.

Macromolecule network	3D residual U-Net, 2 input channels, base width 32, multipliers $(1, 2, 4, 8, 16)$, 2 residual blocks per level, 128^3 input
Macromolecule input	map contour-cropped to the macromolecular region and resampled so its longest edge spans 128 voxels; channels: density with values below the author-recommended contour set to zero, and unfiltered normalized density
Post-processing	map back to 1 Å; Gaussian $\sigma = 3$ Å; Li threshold
Sparse U-Net widths	$(32, 32, 64, 128, 64, 32, 32)$
Sparse U-Net blocks per level	$(4, 6, 8, 12, 4, 4, 4)$, residual, LayerNorm
Output channels	general 6, protein 24, nucleic acid 9 ($C = 39$)
Label radius	1.414 Å around deposited atoms of each class
Macromolecule loss	binary cross-entropy with logits
Sparse U-Net loss	multi-label soft Dice
Precision	BF16 (mixed-dtype LayerNorm)
Training voxels per rank	1.6×10^7
Whole-map inference budget	1.2×10^8 input voxels in a single pass
MeanShift++ bandwidth	1 Å; ligand components < 8 voxels removed

Map-encoder optimization. The macromolecule network uses AdamW with learning rate 10^{-4}, weight decay 10^{-4}, and cosine decay to 10^{-6}. The sparse U-Nets use AdamW with learning rate 5×10^{-4}, a five-epoch linear warm-up from one tenth of that rate, and cosine decay to 10^{-6}. Their gradients are clipped at norm 10. Macromolecule checkpoints are monitored by mask intersection-over-union, and sparse U-Net checkpoints by mean channel intersection-over-union, on held-out training-set entries. These model-selection entries are distinct from the separately prepared evaluation set. Augmentation includes axis flips and permutations and density perturbations; the sparse U-Nets additionally drop 1% of input voxels at random.

Table 5: EMTrunk.

Architecture	AlphaFold-style Pairformer: input embedder, 1-block MSA module, 16 Pairformer blocks, distogram head
Widths	$c_s = 384$, $c_z = 128$, $c_{\mathrm{rp}} = 128$, $c_p = 128$; protein MSA features, no templates
Recycling	1 cycle
Point encoder	seed $121 \to 128 \to 128$ (SiLU); 4 BallTree attention blocks, 8 heads, $\ell = 256$, partitions $\mathcal{T}_{\mathrm{A}}, \mathcal{T}_{\mathrm{B}}$ at 45°
Point seed	centered position (3), Fourier features (24), kind embedding (16), $\bar{\boldsymbol{p}}_p$ (39), $\hat{\boldsymbol{p}}_p$ (39)
Fourier frequencies	$\pi 2^k/128$, $k = 0..3$, sine and cosine per axis
Point–point bias	32 Gaussian RBFs on $[0, 128]$ Å + rank-8 zero-gated low-rank term
Token–point attention	8 heads $\times$ 16; 3D RoPE; mixed triangle hidden 64; outer-product-mean hidden 32
Dropout	0.25 on pair and token–point paths
Distogram	token–token 64 bins on $[2.3125, 21.6875]$ Å; token–point 10 bins with boundaries $0.5, 1, 2, 3, 5, 7.5, 10, 12.5, 15$ Å
Loss weights	$\mathcal{L}_{\mathrm{rr}}: 1$, $\mathcal{L}_{\mathrm{rp}}: 1$, $\mathcal{L}_{\mathrm{res}}: 0.1$, $\mathcal{L}_{\mathrm{pt}}: 0.1$
Training crop	$R \leq 640$ tokens, $P \leq 768$ EM points within 5 Å of crop disto atoms
Optimizer	hybrid Muon/AdamW; peak 5×10^{-4} (AdamW), 1×10^{-3} (Muon); betas $(0.9, 0.95)$; weight decay 10^{-8}
Schedule	linear warm-up 2000 steps; $\times 0.95$ every 5000 steps
Batch	1 crop $\times$ 32 accumulation $\times$ 8 GPUs; BF16 mixed

Table 6: Atomic diffusion module and sampler.

Widths	$c_a = 128$, atom-pair 16, $c_t = 768$
Voxel encoder	$40 \to 128$; 1 sparse conv block + 3 sparse residual blocks, 3^3 kernels, no downsampling
Atom stream	encoder 4 Heterogeneous BallTree Attention + 4 BallTree attention; decoder 4 BallTree attention + 4 Heterogeneous BallTree Attention; 4 heads; residual outputs of blocks beyond the first two in each family zero-initialized
Overlapping BallTrees	atom leaves $\ell_a = 128$; heterogeneous leaves $\ell_{av} = 256$; partitions at 45°; heterogeneous RBF 32 on $[0, 32]$ Å
Token stream	8 blocks; token–token attention 16 heads; point-to-token attention 8 heads $\times$ 16; conditioned transition $n = 2$
EDM	$\sigma_{\text{data}} = 16$; $\log(\sigma/\sigma_{\text{data}}) \sim \mathcal{N}(-1.2, 1.5^2)$; $S = 16$
Loss	$\mathcal{L}_{\text{trunk}} + \frac{1}{3}\mathcal{L}_{\text{mse}} + \mathcal{L}_{\text{lddt}}$; atom weights protein 1, NA 6, ligand 11; smooth lDDT on pairs < 15 Å (protein) and < 30 Å (nucleic acid)
Training crop	$R \leq 512$, $P \leq 640$; voxels within 8 Å of crop disto and ligand atoms
Evaluation	complete assemblies without token, EM-point, or voxel caps; all P, all V
Sampler	Karras schedule, $N = 5$ Euler steps, $\gamma_0 = 0$ (no churn), $\eta = 1$; $\boldsymbol{X}_0 \sim \mathcal{N}(\mathbf{0}, \sigma_0^2 \boldsymbol{I})$ centered at the EM-point centroid; 5 samples
Optimizer	as trunk; warm-up 500; $\times 0.95$ every 2000; EMTrunk parameters on base LR, diffusion on new LR

Table 7: Confidence heads.

Frozen modules	macromolecule network, sparse U-Nets, point and voxel encoders, EMTrunk, atomic diffusion module
Training samples	frozen sampler from noise, no reference input, 5 Euler steps; 2 samples per structure with fresh seeds; fixed seed for validation
Matching	chain and atom-symmetry permutation in the map frame, no rigid alignment; unmatched and unresolved atoms excluded
AlphaFold-style confidence head	4 Pairformer blocks, initialized from pretrained weights; pLDDT 50 bins on $[0, 1]$; PAE and PDE 64 bins on $[0, 32]$ Å; resolved 2 classes
Atom error head	$c_a = 128$, atom-pair 16, 4 heads; 8 blocks of BallTree attention ($\ell_a = 128$) then Heterogeneous BallTree Attention ($\ell_{av} = 256$), partitions alternating per block; 4 error classes $[0, 1), [1, 3), [3, 5), [5, \infty)$ Å
Loss	$\mathcal{L}_{\text{conf}} = \mathcal{L}_{\text{err}} +$ equal-weight mean of pLDDT, PDE, resolved, and PAE (valid frames only) losses; averaged over samples
Training crop	as the atomic diffusion module (Table 6)
Optimizer	AdamW, LR 10^{-4}, weight decay 10^{-8}, no schedule; 30 epochs; checkpoint selected by validation $\mathcal{L}_{\text{conf}}$

H ATTENTION KERNELS

EMTrunk and the atomic diffusion module use the following kernels and memory controls. They change speed and memory only, not the operators of Algorithms 8–11.

- **Triangle attention** runs as a fused Triton kernel, with cuEquivariance as the fallback.
- **Token–point attention** runs with FlexAttention (Dong et al., 2025) in training and with cuEquivariance at evaluation.
- **Heterogeneous BallTree Attention** runs with FlexAttention (Dong et al., 2025), computing only the atom-query, voxel-key quadrant of each leaf (section 3.6).
- **Gated projections** run in BF16.
- **Activation checkpointing** is applied to the Heterogeneous BallTree Attention blocks and to the voxel encoder.
- **Pair transitions** are evaluated in chunks of 128 rows at inference, which bounds their transient memory on large assemblies.

I ADDITIONAL RESULTS AND SUB-COMPONENT EVALUATIONS

I.1 EVALUATION PROTOCOL

Separately prepared evaluation set. We prepare a prospective, sequence-novel evaluation set independently of the training set. Candidates are single-particle cryo-EM entries first released between 1 May and 10 September 2026, after the training-set cutoff, and satisfy the entry-quality criteria in Appendix F. The candidates are clustered with MMseqs2 (Steinegger & Söding, 2017) together with the training set and the complete Cryo2StructData release (Giri et al., 2024), at 40% protein and 80% nucleic-acid sequence identity. A candidate is retained only if its connected sequence cluster contains no entry from either reference collection. Exclusion against Cryo2StructData additionally removes sequence similarity to that widely used model-building dataset. One entry is drawn at random per novel cluster. The final evaluation set contains 103 maps from 103 distinct clusters. The 103 evaluation targets have reported resolutions of 1.52–3.90 Å (median 2.93 Å; 17 below 2.5 Å, 39 in $[2.5, 3.0)$, 39 in $[3.0, 3.5)$, and 8 in $[3.5, 3.9]$ Å) and up to 6,110 polymer residues; 101 contain protein, 14 nucleic acids, and 66 ligands. Every target is evaluated as a complete assembly using the complete map support retained by preprocessing, without structure-based crops or caps on tokens, EM points, or active voxels.

Sequence and atom correspondence. Reference and predicted polymer chains are aligned separately for protein, RNA, and DNA using sequence-alignment scores of 3 for a match, -2 for a mismatch, -2 for gap opening, and -1 for gap extension. Candidate chain pairs require identical aligned residues numbering at least half the length of the shorter chain. A Hungarian assignment first maximizes the number of identical aligned residues and then minimizes the sum of squared matched-atom distances in the map frame. Remaining predicted fragments can recover unmatched reference residues by deterministic greedy assignment, prioritizing sequence identity, then matched-atom count, then coordinate error, without reusing a residue. Atoms are matched by name and element, allowing chemically equivalent atom permutations for supported symmetric side chains, nucleotide phosphate oxygens, and ligands. Ligand instances are assigned within the same chemical-component identity. No rigid-body fitting is used in this correspondence procedure.

Metric calculations. All metrics except the TM-score are evaluated directly in the experimental map frame without rigid-body alignment or Kabsch superposition, measuring absolute coordinate placement; the TM-score is the only metric computed after structural alignment. Each metric is computed per target and macro-averaged with equal weight across the targets for which it is defined; the number of valid targets is given with each result. Accuracy at 3 Å uses all eligible observed reference atoms in that component as its denominator, so any unmatched reference atom is a miss. RMSD is evaluated over identity-aware matched atoms only and is read together with coverage, the fraction of reference atoms that are matched. Completeness is the percentage of reference residues, or of ligand instances for the ligand component, with modeled coordinates; a nucleotide counts as modeled when its input-sequence position has an accepted P or C4′ atom. For methods that return several samples, every per-target metric is averaged over the available samples before the target average; metrics are averaged, not coordinates. CryoASTRA generates five samples per map; CryoACE returns zero, one, or five samples depending on the target; the other baselines return one model. Ligand instances that CryoASTRA omits as crystallization aids count as unmodeled in ligand completeness, and ligand RMSD is undefined for the one target whose only ligand is such an aid, leaving 65 targets. DNA and RNA are reported together as nucleic acid (NA). For the full-assembly builders, NA metrics, including the TM-score, are computed jointly over all RNA and DNA chains of a target. Every TM-score is computed with US-align (Zhang et al., 2022) in sequence-dependent mode (`-TMscore 7`) for every method.

The nucleic-acid-only builders take one molecule type per run, so CryoREAD and EM2NA are run separately on the 18 RNA and DNA subtargets of the 14 NA-containing targets. For each target, their atom-level counts are summed over its subtargets before the per-target metric is formed, RMSD pools matched-atom sums of squares, and the result is macro-averaged over the 14 targets. These pooled scores are reconstructed from separately evaluated predictions rather than from one joint evaluation. For the TM-score, the separately built RNA and DNA models of each of the four targets that contain both are merged into one model and scored against the joint reference. In addition, for two targets the subtarget references hold slightly fewer atoms than the full-target reference (`13FN`:

1,496 versus 1,528; 23VV: 2,027 versus 2,091), so their comparison with the full-assembly builders is close to, but not exactly, matched.

Baselines. ModelAngelo (Jamali et al., 2024) extracts C_α/P candidate peaks and refines residue coordinates and sequences with a graph neural network reading local density crops. CryoAtom2 (Su et al., 2026a;b), the CryoAtom release that adds nucleic acids, detects C_α or phosphorus peaks with patch-wise 3D U-Nets and refines protein and nucleic-acid structures with local attention over 17^3-voxel density crops. E3-CryoFold (Wang et al., 2025), in the released version we run, detects C_α positions in the map and generates protein backbones conditioned on them; its published formulation instead encoded the map as patchified density tokens. EMProt (Li et al., 2026b) identifies C_α candidates in overlapping 60^3 voxel chunks (stride 30) and models protein coordinates with a three-track attention network conditioned on per-C_α density crops and pairwise cuboids. CryoACE (Li et al., 2026a), a diffusion model built on Boltz (Wohlwend et al., 2024) whose recommended pipeline starts from an initial Boltz prediction, fuses one token per non-overlapping density patch into its sequence tokens and, over three self-refinement rounds, adds raw density sampled at the atoms predicted in the previous round; its denoising network receives density only through these token-level features, while inference-time guidance also pulls coordinates toward the map. CryoREAD (Wang et al., 2023) segments the map into phosphate, sugar, and base voxels, connects sugars into a backbone, maps the sequence along it, and builds full atoms that are then refined. EM2NA (Li et al., 2024) segments nucleic acid with one Swin-Conv-UNet on overlapping 48^3 Å^3 boxes, predicts P, C4′, and N1/N9 probabilities with a second, converts them to points by mean shift, traces P and C4′ points into backbones as a vehicle routing problem, and places template nucleotides. ModelAngelo and CryoAtom2 build nucleic acids, CryoREAD and EM2NA build only nucleic acids, and none of the seven baselines builds ligands.

Baseline implementations and coverage. Baselines are run with their official implementations: ModelAngelo from `https://github.com/3dem/model-angelo`, E3-CryoFold from `https://github.com/A4Bio/E3-CryoFold`, EMProt from `https://github.com/huang-laboratory/EMProt`, CryoREAD from `https://github.com/kiharalab/CryoREAD`, and EM2NA from `https://github.com/emberslee/EM2NA`. Every baseline receives the complete deposited map and the input sequences, without cropping or a contour level, except that CryoACE also receives the reported resolution and the recommended contour and builds its own MSAs, and CryoREAD and EM2NA use the recommended contour. For CryoAtom, we evaluate the release from the official repository `https://github.com/YangLab-SDU/CryoAtom` (v2.1.0; nucleic-acid modeling was introduced in v2.0.0) and refer to it as CryoAtom2 (Su et al., 2026a;b). CryoAtom2 returned models for all 103 targets; ModelAngelo, E3-CryoFold, and EMProt for the 101 protein-containing targets; the other two contain only RNA, which E3-CryoFold and EMProt do not model and the evaluated ModelAngelo release does not accept without a protein sequence. CryoACE returned models for 50 targets: it does not accept the 14 nucleic-acid-containing targets and ran out of memory on 39 others. It completed no target above 2,050 polymer residues. CryoREAD and EM2NA have defined atom-level metrics on all 18 nucleic-acid subtargets.

I.2 Targets completed by CryoACE

Table 8 restricts every method to the 50 targets on which CryoACE returned a model. All of them have at most 2,050 polymer residues, and none contains nucleic acids.

Table 8: **The 50 targets completed by CryoACE** (map frame; all at most 2,050 polymer residues, none containing nucleic acids). Macro-averaged over the 50 protein targets and the 33 ligand-bearing targets among them; CryoASTRA and CryoACE are averaged over their samples, and ± is the standard deviation over the samples of a target, averaged over targets with at least two valid samples (CryoACE: 17 targets). Bold: best; '—': unsupported or undefined. Ligand completeness counts ligand instances.

Method	TM-score ↑	Accuracy@3 Å (%) ↑	RMSD (Å) ↓	Coverage ↑	Completeness (%) ↑
Protein *(50 targets)*					
CryoASTRA	**0.976** ±0.002	**94.05** ±0.24	2.79 ±0.12	**1.000**	**100.00**
CryoACE	0.937 ±0.038	85.52 ±4.97	4.72 ±1.70	**1.000**	**100.00**
CryoAtom2	0.846	93.54	**2.45**	0.977	91.81
E3-CryoFold	0.898	68.82	7.32	0.968	91.10
EMProt	0.895	86.47	2.66	0.909	83.73
ModelAngelo	0.820	86.32	3.19	0.905	83.84
Ligand *(33 targets)*					
CryoASTRA	—	51.35 ±4.63	11.42 ±1.24	**1.000**	**100.00**

On these targets CryoASTRA has the highest TM-score (0.976 versus 0.937 for CryoACE) and Accuracy@3 Å (94.05% versus 93.54% for CryoAtom2) at 100% completeness, which follows from sequence-conditioned generation and accounts for part of this lead. Its matched RMSD (2.79 Å) is higher than those of CryoAtom2 (2.45 Å) and EMProt (2.66 Å), which model 91.81% and 83.7% of the residues. On the 33 ligand-bearing targets, CryoASTRA models every ligand instance at 51.4% Accuracy@3 Å.

I.3 Sub-component breakdowns on the evaluation set

Table 9 reports protein backbone and side chains and nucleic-acid backbone, phosphate, sugar, and base groups on the 103 targets. On the 101 protein targets, CryoASTRA attains the lowest matched RMSD across both backbone (3.17 Å versus 4.95–9.19 Å, with CryoACE at 4.37 Å on 50 targets) and side chains (3.69 Å versus 5.45–10.54 Å, CryoACE: 5.00 Å) while maintaining full residue completeness.

Table 9: **Sub-component modeling accuracy on the evaluation set.** Macro-averaged over valid targets, with the number of valid targets in parentheses where it differs from the group size; multi-sample methods are averaged over their samples, and ± is their within-target standard deviation over samples (CryoACE: 17 targets with at least two samples). §: nucleic-acid-only builders pooled over RNA and DNA subtargets (Appendix I.1). Bold: best.

Method	Accuracy@3 Å (%) ↑	RMSD (Å) ↓	Coverage ↑
Protein backbone *(101 targets; CryoACE on 50)*			
CryoASTRA	**94.28** ±0.35	**3.17** ±0.19	**1.000**
CryoACE	87.60 ±5.08	4.37 ±1.69	**1.000**
CryoAtom2	91.62	6.40	0.982
E3-CryoFold	79.13	9.19	0.958
EMProt	83.80	5.95	0.903
ModelAngelo	86.53	4.95	0.911
Protein side chain *(101 targets; CryoACE on 50)*			
CryoASTRA	**90.64** ±0.44	**3.69** ±0.17	**1.000**
CryoACE	83.37 ±4.89	5.00 ±1.73	**1.000**
CryoAtom2	89.01	6.80	0.983
E3-CryoFold	54.47	10.54	0.959
EMProt	81.03	6.49	0.909
ModelAngelo	84.14	5.45	0.916
Nucleic acid backbone *(14 targets; ModelAngelo on 12)*			
CryoASTRA	**46.17** ±1.53	**18.30** ±1.07	**1.000**
CryoAtom2	41.78	21.21 (13)	0.814
ModelAngelo	26.88	31.13	0.638
EM2NA§	8.69	31.17	0.423
CryoREAD§	8.93	47.17	0.842
Nucleic acid phosphate *(14 targets; ModelAngelo on 12)*			
CryoASTRA	**44.35** ±1.71	**18.68** ±1.05	**1.000**
CryoAtom2	41.80	21.38 (13)	0.823
ModelAngelo	27.00	31.23	0.642
EM2NA§	8.05	31.39	0.425
CryoREAD§	10.79	47.65	0.842
Nucleic acid sugar *(14 targets; ModelAngelo on 12)*			
CryoASTRA	**46.80** ±1.55	**18.16** ±1.08	**1.000**
CryoAtom2	41.77	21.15 (13)	0.811
ModelAngelo	26.85	31.08	0.636
EM2NA§	8.92	31.09	0.423
CryoREAD§	8.28	47.00	0.841
Nucleic acid base *(14 targets; ModelAngelo on 12)*			
CryoASTRA	**48.90** ±1.44	**17.63** ±1.04	**1.000**
CryoAtom2	39.15	19.75 (13)	0.699
ModelAngelo	23.87	26.46	0.484
EM2NA§	7.90	30.63	0.401
CryoREAD§	4.27	45.31	0.747

I.4 Variability across targets and across samples

Table 10 gives, for every TM-score, Accuracy@3 Å, RMSD, and completeness cell of Table 1, the sample standard deviation across targets of the per-target values. The spread is large for every method because the evaluation set mixes maps of different resolution, size, and composition; it describes the benchmark, not the uncertainty of the mean. The five CryoASTRA samples of a target differ far less than the targets do (0.0016 versus 0.059 in protein TM-score, and 0.36 versus 10.7 points in protein Accuracy@3 Å) and less than the CryoACE samples (0.038 and 5.0 points). The sample spread of CryoASTRA is larger on nucleic acids (0.013 in TM-score, 1.3 points) and ligands (4.7 points in Accuracy@3 Å). Coverage and completeness do not vary between samples for either model.

Table 10: **Across-target variability of Table 1.** Mean ± sample standard deviation across valid targets; CryoACE protein metrics cover its 50 completed targets, CryoAtom2 NA RMSD 13 targets, and CryoASTRA ligand RMSD 65 targets. NA TM-scores are joint over RNA and DNA chains. §: atom counts pooled over RNA and DNA subtargets, and TM-score from the merged RNA and DNA models (Appendix I.1).

Method	TM-score	Accuracy@3 Å (%)	RMSD (Å)	Completeness (%)
Protein *(101 targets)*				
CryoASTRA	0.972 ± 0.059	92.49 ± 10.71	3.47 ± 5.58	100.00 ± 0.00
CryoACE	0.937 ± 0.131	85.52 ± 23.07	4.72 ± 7.27	100.00 ± 0.00
CryoAtom2	0.855 ± 0.178	90.34 ± 15.77	6.64 ± 14.90	92.48 ± 5.77
E3-CryoFold	0.891 ± 0.182	67.05 ± 22.52	9.96 ± 17.55	90.61 ± 9.07
EMProt	0.880 ± 0.172	82.44 ± 22.61	6.27 ± 13.28	83.65 ± 14.57
ModelAngelo	0.830 ± 0.218	85.36 ± 18.04	5.24 ± 9.68	84.66 ± 13.87
Nucleic acid (NA) *(14 targets; ModelAngelo on 12)*				
CryoASTRA	0.561 ± 0.336	47.39 ± 40.07	18.02 ± 19.20	100.00 ± 0.00
CryoAtom2	0.424 ± 0.287	40.60 ± 35.39	20.69 ± 16.20	66.62 ± 28.22
ModelAngelo	0.274 ± 0.237	25.54 ± 21.32	29.76 ± 14.73	50.49 ± 31.54
EM2NA§	0.133 ± 0.150	8.34 ± 10.13	30.96 ± 15.20	35.52 ± 27.60
CryoREAD§	0.143 ± 0.163	6.85 ± 8.01	46.44 ± 18.17	71.88 ± 17.37
Ligand *(66 targets)*				
CryoASTRA	—	47.35 ± 38.76	15.08 ± 16.29	97.60 ± 13.22

I.5 EM-point localization protocol

The probe of section 4.3 evaluates voxel-wise semantic probabilities and MeanShift++ before voxel encoding and structure generation, so an EM point is scored as an estimate of one atom and its matched error is not a finished-model RMSD. Protein and nucleic-acid EM points are extracted from the disto-atom channels, and each set is matched one-to-one, in the map frame and without rigid alignment, both to the deposited disto atoms and to the deposited residue centers (C_α or $C1'$); ligand EM points are matched to deposited ligand-instance centroids. Candidate pairs within 3 Å are sorted by distance and accepted greedily, closest first, only if neither EM point nor reference position has already been matched. Precision is the fraction of EM points that are matched, recall the fraction of deposited atoms or ligand instances that are matched, and matched RMSD and median matched error are computed over matched pairs; because the 3 Å threshold bounds these errors, they are read together with recall. RMSD and the median require at least one match. EM-point extraction is deterministic, so each map contributes one value per metric, and metrics are macro-averaged over the targets that contain the molecule type. Ligand EM points target one point per ligand atom but are matched one-to-one to instance centroids, so at most one EM point per instance can be matched. The probe counts every non-polymer entity, including ions, as a ligand instance, so it covers 67 targets, one more than Table 1, which classifies a free amino acid as protein; ligand precision is over the 66 targets with a ligand EM point, and ligand RMSD and median error are over the 59 targets with a match.

J Operator efficiency

We benchmark the three irregular operators in isolation on ten large cryo-EM entries from the upper quartile of operator workload (Figure 9); these are operator timings, not end-to-end inference or a speedup claim over a competing builder. The sparse U-Nets receive the macromolecule-gated voxels (7.2×10^5–2.4×10^6) and are compared with TorchSparse++ (Tang et al., 2023) and fVDB (Williams et al., 2024); at matched 32-bit precision they use 4.2–4.4% less peak memory than TorchSparse++ (Appendix J.1). Overlapping BallTree preparation receives batches of 16 mixed sets of deposited atoms and active voxels (2.1×10^5–6.5×10^5 rows per tree) and is compared with Erwin's CPU preparation, which parallelizes over the 16 trees with OpenMP, and with scikit-learn's plain BallTree as a non-equivalent reference (section 3.3). For batches of 16 atom–voxel sets of 2.1×10^5–6.5×10^5 rows each, one GPU prepares all partitions in 34–52 ms, whereas Erwin's multi-threaded CPU construction takes 0.32–1.47 s; Erwin's benchmarks contain at most 3.2×10^4 points per sample,

and its tabulated tree-construction timings stop at 1.6×10^4 points per tree (Zhdanov et al., 2025). MeanShift++ receives the Li-selected voxels (2.7×10^4–9.6×10^4) and is compared with the original CPU implementation (Jang & Jiang, 2021). The benchmark protocol, the training-capacity sweep, and batch-1 tree timings are in Appendices D and J.2.

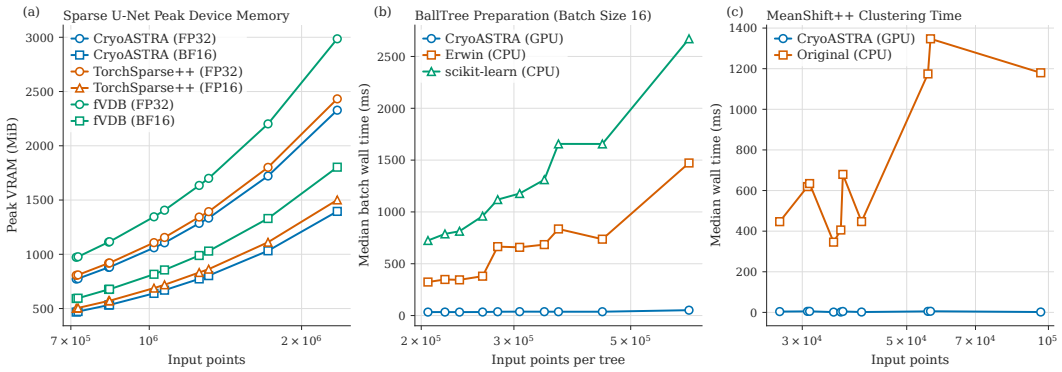

Figure 9: **Operator efficiency. Left:** Sparse U-Net peak device memory. **Middle:** Overlapping BallTree preparation wall time per batch of 16 atom–voxel sets. **Right:** MeanShift++ wall time. Input counts are on a logarithmic axis; parentheses in the legends give precision or device. Each point uses one warm-up and three timed repeats (Appendix J.2).

J.1 Sparse U-Net comparison

Figure 10 expands the U-Net panel of Figure 9 to show both forward latency and peak device allocation on the same ten macromolecule-gated cryo-EM maps of Appendix J (717,283–2,352,378 gated voxels). All models use the same encoder–decoder widths and residual-block counts, but our LayerNorm–SiLU U-Net and the TorchSparse++/fVDB BatchNorm–ReLU ports are not numerically identical models. fVDB is shown in FP32 and BF16 on the same ten maps; fVDB FP16 is not plotted.

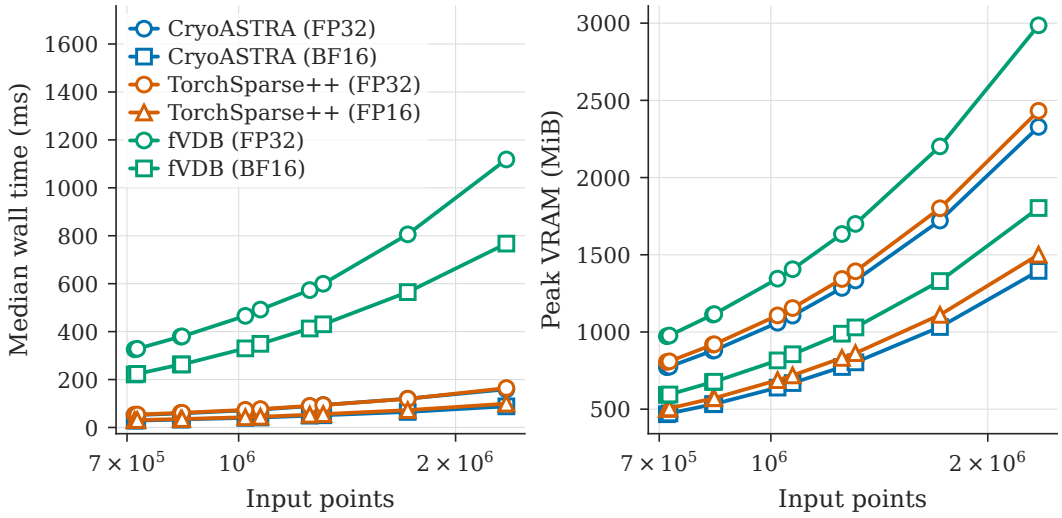

Figure 10: **Sparse U-Net forward cost on macromolecule-gated voxels. Left:** forward latency, one warm-up and three timed repetitions per map, with cached topology plans for fVDB; plan construction is excluded. **Right:** peak `torch.cuda.memory_allocated`, including the resident model and input, measured during one untimed forward with the allocator peak reset, not from process-wide GPU memory or a training step. Parentheses in the legend give precision.

At matched 32-bit precision, CryoASTRA uses 4.2–4.4% less peak allocated memory than TorchSparse++; CryoASTRA BF16 uses 6.5–7.1% less than TorchSparse++ FP16. BF16 and FP16 each occupy two bytes per scalar, so a precision-width difference cannot explain the latter result. Both runs use implicit GEMM; TorchSparse++ uses on-the-fly hashing with sorting disabled. Kernel-map construction and temporary-allocation lifetimes also differ between the implementations; this comparison does not isolate their individual contributions to peak memory.

J.2 BallTree construction benchmarks

Construction benchmark. All three panels of Figure 9 use one warm-up and three timed repeats per point, with time limits of 240 s (sparse U-Nets), 120 s (tree preparation), and 300 s (MeanShift++). The tree panel is a construction microbenchmark, not an attention or end-to-end timing: it isolates preparation of the partitions consumed by BallTree attention and Heterogeneous BallTree Attention. Each workload is a mixed set of deposited atoms and the active voxels $\mathcal{V}$ (the Li-thresholded union over the sparse U-Nets), replicated 16 times. Erwin receives the concatenated batch and parallelizes over trees with OpenMP; scikit-learn builds 16 conventional trees with joblib threads (`n_jobs=16`); and CryoASTRA launches its 16 GPU constructions sequentially. All methods use leaf size 128 (the model's mixed partitions use $\ell_{av} = 256$). CryoASTRA and Erwin both prepare the primary tree and five rotated multiscale partitions, more than the single rotated partition $\mathcal{T}_{\mathrm{B}}$ that the model uses (section 3.3), whereas scikit-learn prepares only its plain query tree.

Batch-1 control. Table 11 reports a single-cloud measurement on the same hardware used for Figure 9. Each call receives one real mixed atom–voxel cloud, uses one warm-up and three timed repeats, and reports the median. The CryoASTRA and Erwin columns include the primary tree and five rotated multiscale partitions, more than the single rotated partition of the model; scikit-learn builds one plain tree with `leaf_size=128`. Without batching, Erwin cannot parallelize across trees.

Table 11: Batch-1 tree-preparation median wall time (ms). M is points in the single mixed cloud. CryoASTRA and Erwin prepare the primary tree and five rotated multiscale partitions; scikit-learn builds a non-equivalent plain tree.

M	CryoASTRA GPU	Erwin CPU	scikit-learn CPU
205,972	2.186	82.110	44.197
221,861	2.224	85.881	48.922
236,448	2.220	87.486	50.039
261,579	2.212	95.888	59.098
279,445	2.421	158.813	68.927
307,587	2.426	159.994	73.094
342,683	2.440	168.684	81.166
364,525	2.426	185.443	102.782
441,708	2.441	187.323	103.859
645,230	3.328	347.136	168.474

J.3 Maximum-workload benchmarks on `74415_9ZLY`

The largest workload evaluated across all benchmarked entries corresponds to target `74415_9ZLY`. Table 12 reports the raw timings and speedups across three timed repetitions following one warm-up. For batch-16 Overlapping BallTree preparation (645,230 points per tree, leaf size 128, constructing the primary partition and five rotated partitions without subsequent attention), CryoASTRA on a single GPU achieves a median latency of 51.88 ms, corresponding to a **28.4**× speedup over Erwin's OpenMP multi-threaded CPU implementation (1,472.31 ms) and **51.5**× over scikit-learn's standard query tree (2,670.45 ms, which builds single-scale trees without rotation). For candidate-atom localization via MeanShift++ on the 96,308 Li-selected voxels of `74415_9ZLY`, our custom CUDA kernel executes in 1.85 ms, delivering a **637.7**× speedup over the original CPU implementation (1,179.80 ms).

Table 12: **Operator benchmarks on the maximum-workload entry** (`74415_9ZLY`). BallTree preparation evaluates 16 trees without attention; scikit-learn builds standard query trees without rotation.

Operator & workload	Implementation	Median	Timed runs (ms)
BallTree	CryoASTRA GPU (`pointseg`)	**51.88 ms**	52.02, 51.80, 51.88
645,230 pts/tree	Erwin CPU (OpenMP, 16 th.)	1,472.31 ms	1,476.62, 1,472.31, 1,469.77
batch 16	scikit-learn CPU (joblib, 16 th.)	2,670.45 ms	2,671.27, 2,670.45, 2,659.23
MeanShift++	CryoASTRA CUDA	**1.85 ms**	1.84, 1.85, 1.87
96,308 candidate pts	Original CPU (Jang & Jiang, 2021)	1,179.80 ms	1,182.69, 1,179.80, 1,172.77

K ATOM ERROR ESTIMATION

The atom error head is evaluated on the five samples per map generated with $N = 5$ steps for section 4.2. It ran out of memory on five of the 103 evaluation-set entries, and every statistic below uses the remaining 98. The matched atoms of each sample are labeled by whether $e_a \geq t$ Å. AUROC, AUPRC, and the Brier score of $r_{a,t}$ are computed for each entry over the matched atoms of its five samples and averaged over the 98 entries with equal weight (Table 13); the AUPRC falls with t because large errors are rarer. The threshold statistics of section 4.5 instead pool the matched atoms of all samples of the 98 entries, 6,089,485 atom–sample pairs of which 671,711 have $e_a \geq 3$ Å: 118,566 pairs have $r_{a,3} \geq 0.8$, and 108,543 of them have $e_a \geq 3$ Å; 188,229 of the pairs with $e_a \geq 3$ Å have $r_{a,3} < 0.2$. As fractions, atoms with $r_{a,3} \geq 0.8$ make up 1.95% [1.31, 2.73] of the matched atoms, and atoms with $r_{a,3} < 0.2$ still hold 28.0% [19.5, 37.6] of those with $e_a \geq 3$ Å. All 95% intervals are percentile intervals from 5,000 bootstrap resamples of entries; the atoms and samples of one entry are never treated as independent. The evaluated checkpoint was selected by minimum validation loss ($\mathcal{L}_{\text{conf}}$, Table 7), not on the evaluation set; an earlier checkpoint of the same head had been scored on the evaluation set before, so these numbers do not come from a strictly single-use test set.

Table 13: **Atom error head on the evaluation set** (98 entries; per-entry metrics averaged with equal weight; 95% percentile intervals from 5,000 bootstrap resamples of entries).

Error threshold	AUROC ↑	AUPRC ↑	Brier ↓
$e_a \geq 1$ Å	0.912 [0.904, 0.920]	0.809 [0.793, 0.825]	0.102 [0.094, 0.111]
$e_a \geq 3$ Å	0.937 [0.922, 0.950]	0.606 [0.568, 0.644]	0.048 [0.037, 0.061]
$e_a \geq 5$ Å	0.937 [0.920, 0.953]	0.450 [0.397, 0.504]	0.039 [0.026, 0.053]