

483 A Examples of proteins and projections

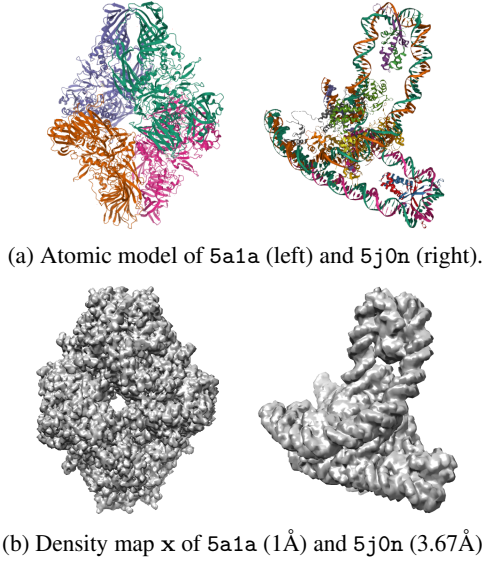


Figure 10: Proteins with different symmetries.

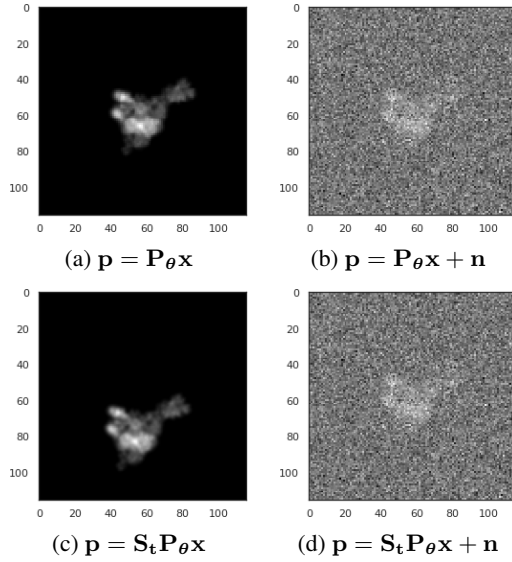


Figure 11: Example projections ($n \sim \mathcal{N}(0, 16I)$).

484 B Sampling of orientations

485 Figure 12 shows four distributions of orientations and the distributions of distances they induce.
 486 As shorter distances are under-sampled, we uniformly resampled the distances to avoid biasing the
 487 training of our distance estimator.

488 While we control the distributions of orientations and distances to facilitate distance learning, we
 489 cannot control them when recovering orientations of a given set of projections. Comparing Figure 13a
 490 with 8a, and 13b with 9e, shows that the recovered orientations and the reconstructed density are
 491 barely affected by a non-uniform sampling of orientations—a condition that might happen in real
 492 cryo-EM acquisitions.

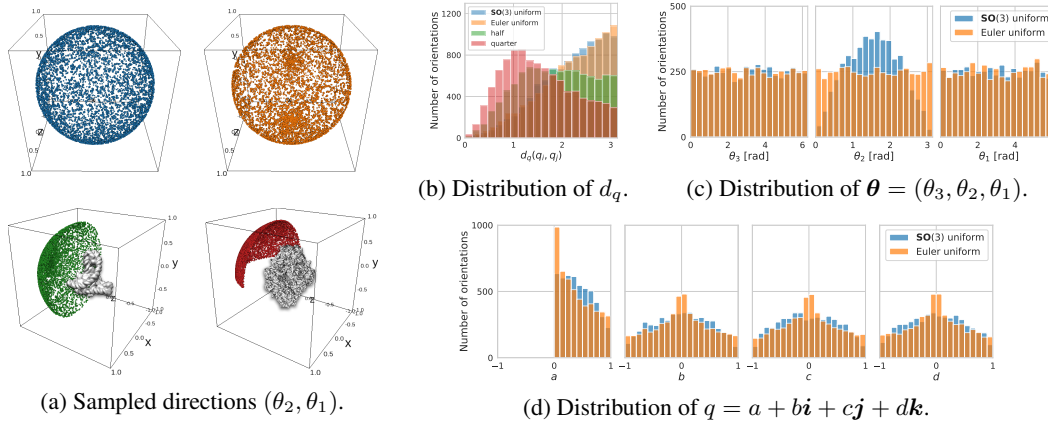


Figure 12: Sampling of orientations from four distributions: (blue) uniform on $SO(3)$, (orange) uniform on Euler angles, (green) Euler uniform restricted to half the directions $(\theta_2, \theta_1) \in [0, \frac{\pi}{2}] \times [0, 2\pi]$, and (red) $SO(3)$ uniform restricted to a quarter of the directions $(\theta_2, \theta_1) \in [0, \frac{\pi}{2}] \times [0, \pi]$.

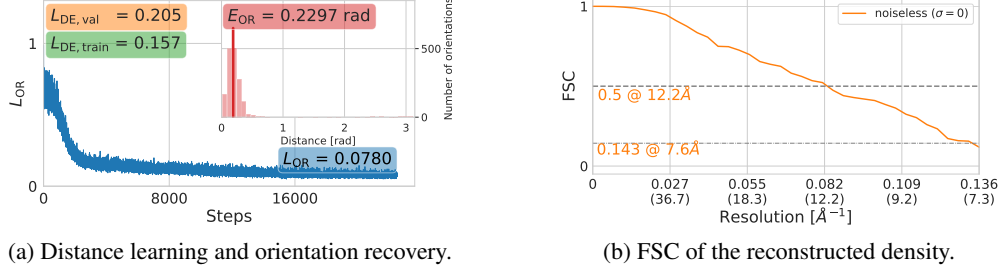


Figure 13: Orientation recovery and density reconstruction from noiseless projections of 5j0n acquired from non-uniformly sampled orientations (Euler uniform, Figure 12 (orange)).

C Optimization settings

We optimized (3) with the RMSProp optimizer [50] and a learning rate of 10^{-3} for 150 epochs. Batches of 256 pairs resulted in 247 steps per epoch for the training sets and 28 for the validation sets (Table 1). It took about 3.3 hours of a single Tesla T4 or 8.75 hours of a single Tesla K40c. Our code supports training on multiple GPUs. We optimized (4) with the Adam optimizer [51] and a learning rate of 0.5 until convergence on batches of 256 pairs sampled from the test sets (Table 1). It took about 3.75 hours of a single Tesla K40c (without early stopping). We optimized (5) with the FTRL optimizer [52], a learning rate of 2, and a learning rate power of -2 on batches of 256 orientations sampled from the test sets (Table 1). We reported the lowest of 6 runs (3 per value of m) of 300 steps each. This took about 50 minutes on a Xeon Silver 4114 CPU.

D Orientation recovery from exact distances

To verify that the lack of a convexity guarantee for (4) and the sampling of the sum are non-issues in practice, we attempted orientation recovery under exact distance estimation $\hat{d}_p(\mathbf{p}_i, \mathbf{p}_j) = d_q(q_i, q_j)$. Orientations were perfectly recovered; Figure 14 shows the convergence of L_{OR} to zero. Figure 15 shows how (5) could then perfectly align the recovered and true orientations—leading to $E_{OR} = 0$. It illustrates how alignment is necessary to evaluate the performance of orientation recovery.

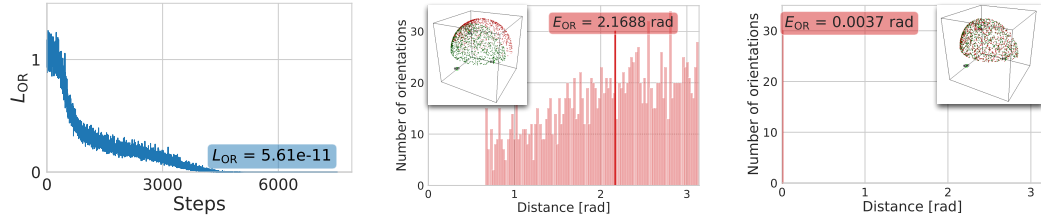


Figure 14: Example of perfect orientation recovery (for 5a1a). The loss L_{OR} (4) converges to zero when the distance estimation is perfect, i.e., $\hat{d}_p(\mathbf{p}_i, \mathbf{p}_j) = d_q(q_i, q_j)$.

(a) Orientations before alignment. (b) Orientations after alignment.

Figure 15: Example of perfect alignment (5) after a perfect recovery (4). The red histogram shows the errors (a) $\{d_q(q_i, \hat{q}_i)\}$ and (b) $\{d_q(q_i, \mathbf{T}\hat{q}_i)\}$, with the mean E_{OR} highlighted. True (green) and recovered (green) directions are shown in the insert. While both colors are seen in (b), they are superimposed.

E Euclidean distance between projections

We evaluate $\hat{d}_p(\mathbf{p}_i, \mathbf{p}_j) = \|\mathbf{p}_i - \mathbf{p}_j\|_2$ (i.e., the Euclidean distance) as a baseline distance estimator. Figure 16 shows the relationship between \hat{d}_p and d_q . Two main observations can be made from this experiment. First, as suspected, \hat{d}_p fails to be a consistent predictor of d_q , even in the simple imaging conditions considered here (no noise, no shift, no PSF). In particular, the larger the orientation distance d_q , the poorer the predictive ability of \hat{d}_p (the plot plateaus). Second, because 5a1a has

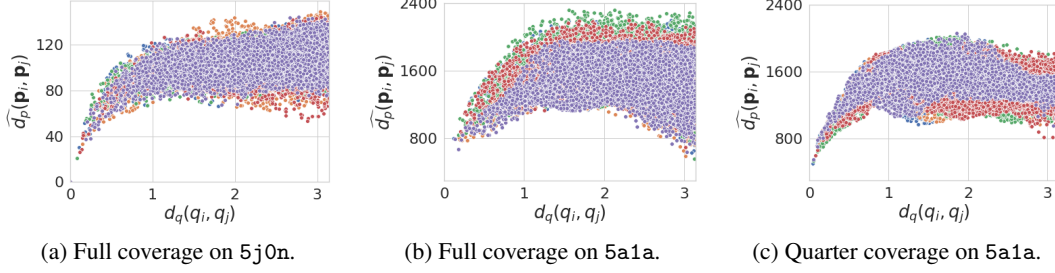
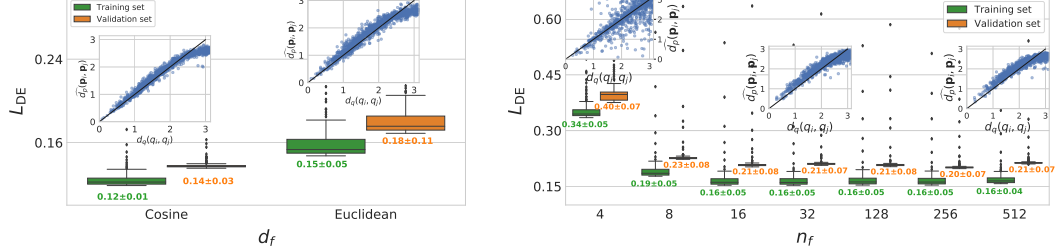


Figure 16: Euclidean distance between projections $\hat{d}_p(\mathbf{p}_i, \mathbf{p}_j) = \|\mathbf{p}_i - \mathbf{p}_j\|_2$ versus their actual relative orientation $d_q(q_i, q_j)$. We randomly selected 5 projections from $P = 5,000$: each color represents the distances between one of those and all the others.

D2 symmetries, two projections might be identical while not having been acquired from the same orientation. Restricting directions to a quarter captures only one of four identical projections, solving the issue.

F SNN: feature distance and embedding dimension

There are multiple options for a distance function d_f between two features $\mathbf{f}_i = \mathcal{G}_w(\mathbf{p}_i) \in \mathbb{R}^{n_f}$. Figure 17a compares the use of the Euclidean distance $d_f(\mathbf{f}_i, \mathbf{f}_j) = \|\mathbf{f}_i - \mathbf{f}_j\|_2$ and the cosine distance $d_f(\mathbf{f}_i, \mathbf{f}_j) = 2 \arccos\left(\frac{\langle \mathbf{f}_i, \mathbf{f}_j \rangle}{\|\mathbf{f}_i\| \|\mathbf{f}_j\|}\right)$. The cosine distance results in a lower L_{DE} , which makes \hat{d}_p a better estimator of d_q . This superiority of the cosine distance is likely due to its capacity to model the elliptic geometry of $\mathbf{SO}(3)$, a feat the Euclidean distance does not achieve, the Euclidean space being neither periodic nor curved.



(a) Performance w.r.t. the feature distance d_f . (b) Performance w.r.t. the embedding dimensionality n_f .

Figure 17: Performance of our distance estimator \hat{d}_p w.r.t. two design choices. The box plots show the distance learning loss L_{DE} (3). The inserted plots show the relationship between $d_p(\mathbf{p}_i, \mathbf{p}_j) = d_f(\mathcal{G}_w(\mathbf{p}_i), \mathcal{G}_w(\mathbf{p}_j))$ and $d_q(q_i, q_j)$ on 1,000 pairs sampled from 5j0n.

Figure 17b shows the performance of our distance estimator \hat{d}_p depending on the size n_f of the feature space. It clearly indicates that a space of $n_f = 4$ dimensions is insufficient to represent the variability of projections. That is a motivation to embed the projections in a space of higher dimensions that can represent more variations than the orientation, and can abstract that variation by solely considering the distances between the embedded projections $\mathbf{f}_i = \mathcal{G}_w(\mathbf{p}_i)$. While our choice of $n_f = 512$ might be overkill ($n_f = 16$ seems sufficient), it is not penalizing.

G Convolutional neural network architecture

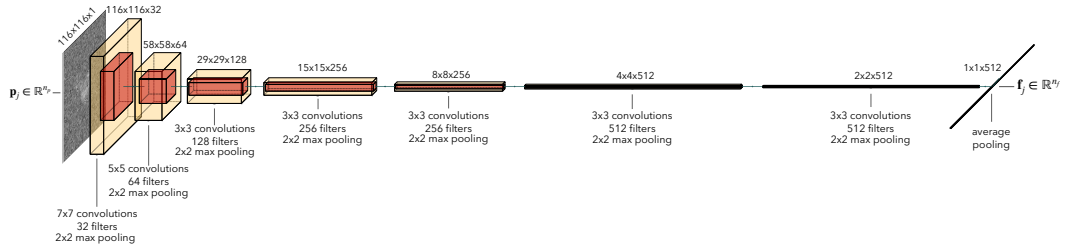


Figure 18: Architecture of \mathcal{G}_w , the convolutional neural network that extracts feature vectors $\mathbf{f}_j = \mathcal{G}_w(\mathbf{p}_j) \in \mathbb{R}^{n_f}$ from projections $\mathbf{p}_j \in \mathbb{R}^{n_p}$. While $n_f = 512$ and $n_p = 116 \times 116$ in our experiments, \mathcal{G}_w can accommodate any image size thanks to the global average pooling layer.