1. Why perform affine transformation on quaternion?

It seems straightforward to perform affine transformation on the rotation matrix, however, we find that these methods concentrate the distribution in an undesired way. The simplest way is to left or right multiply a $3 \times 3$ invertible matrix $W$ to rotation matrix $R$, and then use SVD decomposition or the Gram-Schmidt orthogonalization to project the result $WR$ or $RW$ into a rotation matrix, i.e., $\text{SVD}(WR)$, $\text{SVD}(RW)$, $\text{GS}(WR)$, $\text{GS}(RW)$.

However, if we use SVD decomposition ($\text{SVD}(WR)$ or $\text{SVD}(RW)$) or right Gram-Schmidt orthogonalization ($\text{GS}(RW)$), it can be proved that it only functions as a rotation and thus has poor expressivity. If we use left Gram-Schmidt orthogonalization ($\text{GS}(WR)$), we concentrate the distribution to 4 different modes where the relative angle between each pair of them is fixed to be $180^\circ$ as shown in Figure 1, which is usually undesired.

The 4-modal distribution can be understood as multiplying $W$ to the first column and then normalizing it can be interpreted as scaling $S_2$ to an ellipsoid and then squeezing it to make it concentrated similar to quaternion affine transformation. Multiplying $W$ to the second column and using the Gram-Schmidt orthogonalization can be viewed as scaling a circle $S_1$ to an oval and then squeezing it to make it concentrated. Those two operations both have the property of antipodal symmetry similar to quaternion affine transformation. However, keeping antipodal symmetry is needed in quaternion affine transformation as $q$ and $-q$ represent the same rotation, while in rotation matrix affine transformation, if $(c_1, c_2, c_3)$ is one mode where $c_i$ is the $i$-th column of the rotation matrix, then $(-c_1, -c_2, c_3)$, $(-c_1, c_2, -c_3)$ and $(c_1, -c_2, -c_3)$ will also be the mode due to this symmetry.

One can also treat the first two or all three columns of the rotation matrix as a vector and then perform affine transformation on it, but these methods usually lead to even more modes and the inverse of those transformations are much harder to solve.

On the contrary, as shown in Figure 5, the affine transformation on quaternion has elegant geometric interpretations and it can concentrate the distribution to only one mode, so we choose to perform affine transformation on quaternion.

Table 1 reports results of ablation study of quaternion affine transformation and rotation matrix affine transformation on ModelNet10-SO3 datasets. It shows that the overall performance of quaternion affine transformation is better than rotation matrix affine transformation. Note that our average median error is higher because the result is highly affected by bathtub category, whereas bathtub exhibits symmetry while there is only one ground truth annotation for each image.

2. How to obtain the $4 \times 4$ Invertible Matrix?

We present two methods to parameterize the invertible matrix $W$ in quaternion affine transformation. One is to output an unconstrained matrix, and the other is to obtain the matrix by LU decomposition $W = PL(U+S)$, as in [3]. For this method, we follow [3] to construct $P$ as a fixed permutation matrix, $L$ as a lower triangular matrix with ones.

---

*Equal contribution
†He Wang and Baoquan Chen are the corresponding authors ({hewang, baoquan}@pku.edu.cn).
on the diagonal, \( U \) as an upper triangular matrix with zeros on the diagonal, and \( S \) as a diagonal matrix with positive entries.

We conduct an ablation study on different strategies in both unconditional and conditional experiments, and the results are shown in Table 2. We find that both unconditional and conditional experiments, and the entries. We also report results on ModelNet10-SO3 dataset and adopt SYMSOL-I dataset with log-likelihood (\( \log \)-likelihood) as the evaluation metrics. We use uniform distribution in \( \text{SO}(3) \) as base distribution. The best performance is shown in bold and the second best is with underlined.

![Table 1. Ablation of different affine transformation on ModelNet10-SO3 dataset. We adopt 15° accuracy, 30° accuracy and median error as the evaluation metrics. We use uniform distribution in SO(3) as base distribution. The best performance is shown in bold and the second best is with underlined.](image)


### 3. Ablation \( \frac{\sqrt{2}}{2} \) trick

We present a \( \sqrt{2} \) trick to alleviate discontinuity encountered in linear combination of Mobius transformations. We have to confess that the expressivity of single Mobius transformation are reduced due to restriction of \( \|w\| \), however, as it enables linear combination of Mobius transformations, the general expressivity are gained.

Without \( \frac{\sqrt{2}}{2} \) trick, we can not compute the inverse process of linear combination of Mobius transformations due
to the ambiguity of Møbius combination. So we have to compare the gain from using linear combination and the cost of this $\frac{\sqrt{2}}{2}$ trick.

We conduct ablation study of $\frac{\sqrt{2}}{2}$ trick on Cube dataset in Fig. 2. Seen from curves [1-2], the introduction of $\frac{\sqrt{2}}{2}$ trick indeed limits the performance; however, from [3] vs. [1] and [5] vs. [4], the accommodation of linear combination in return provides larger expressivity.

4. More Discussions on using other Normalizing Flows on SO(3)

4.1. ReLie

ReLie [2] performs normalizing flows on the Lie algebra of Lie groups, where for rotations in SO(3), the Lie algebra is the axis-angle representation ($\theta e$) in $\mathbb{R}^3$. ReLie applies normalizing flows in Euclidean space and then maps the Euclidean samples back to SO(3) space through the exponential map. Noticing that the axis-angle representation periodically covers SO(3) space, i.e., $(\theta + 2i\pi)e$, $i \in \mathbb{Z}$ represent the same rotation, ReLie restricts the output of the Euclidean normalizing flows in a sphere with the radius $r$ by an $r \cdot \tanh(\cdot)$ operation to resolve the infinity-to-one issue. However, ReLie still exhibits several drawbacks. Firstly, even if limited in a sphere with the radius of $\pi$, axis-angle representation is not a diffeomorphism to SO(3) [11], thus the normalizing flows cannot build diffeomorphic mappings between Lie algebra and Lie group and suffer from discontinuous rotation representation. Secondly, with the non-linear $\tanh(\cdot)$ operation, at the inverse stage, $\tanh^{-1}(\cdot)$ can yield infinitely large values, resulting in numerical instability (see Figure 3). Note that this issue can not be solved by replacing $\tanh(\cdot)$ by other functions or varying the value $r$, since a non-linear mapping function is always needed to restrict $\mathbb{R}^3$ into a sphere.

In our experiment, ReLie fails to fit the peak distribution (see Table 1 in the main paper). This is because the normalizing flows learn to push mostly all the points to the peak, and in the inverse process, points that are not close to the peak are pushed back near to the surface of the $r$-sphere in Lie algebra, which yields NaN in training and breaks the process.

4.2. ProHMR

ProHMR [4] leverages 6D rotation representation and considers the first two columns of the rotation matrix as a 6D Euclidean vector. Thus, it applies Euclidean normalizing flows on the 6D vectors and maps the samples back to SO(3) by Gram-Schmidt orthogonalization. Without any constraint, the infinity-to-one mapping clearly makes the probability density of a given rotation intractable. This is because a rotation in SO(3) corresponds to infinity points in Euclidean space and the PDF of the rotation should be the integration of all the corresponding Euclidean points. Due to the unavailability of the probability densities, we do not incorporate it as our baseline in experiments.

5. Application: Entropy Estimation of Arbitrary Distribution

One outstanding feature of our Normalizing Flows compared to other probability inference methods on SO(3) (like [8]) is its ability for efficient samples. [8] can only sample by querying a large number of rotations and calculating the probability density function. For highly peaked distribution, this method may fail as it is hard to have queries that are enough close to the peak such that the probability is not close to zero. However, we can sample by transforming $z$ sampled from a base distribution through our flows. Efficient sampling makes it possible to estimate properties of data $x$, for example, entropy can be estimated via Monte Carlo:

$$S = E[\log p(x)].$$  (1)

In this experiment, we compare our rotation NFs with Implicit-PDF in approximating the entropy of the target distributions. In order to obtain the ground truth entropy for evaluation, we adopt multiple matrix Fisher distributions (whose entropy can be analytically computed) with different parameters as the target distributions. We sample 600k points from each target distribution as the training data and evaluate both our method and Implicit-PDF by randomly sampling N (N=5, 10, 100, 1k, 10k) points from the learned distributions. The results are shown in Figure 4. We can see that even when the sampling size is small, our rotation normalizing flows still achieve accurate estimation of entropy for different target distributions, while Implicit PDF fails to do so.

6.1. Determinate of Jacobian

Mobius Transformation The forward pass of Mobius transformation is defined as follows:
\[ c' = f_\omega(c) = \frac{1 - \|\omega\|^2}{\|c - \omega\|^2} (c - \omega) - \omega. \]  
(2)

Where \( c, c' \) are vectors \( \in S^2 \), \( \omega \) is a point \( \in \mathbb{R}^3 \) that satisfies \( \|\omega\| < \frac{\sqrt{2}}{2} \). The Jacobian matrix is given by partial derivatives of \( c' \) with respect to \( c \),
\[ J = \frac{dc'}{dc} = \frac{1 - \|\omega\|^2}{\|c - \omega\|^2} (I - 2(c - \omega)(c - \omega)^T) \frac{1}{\|c - \omega\|^2} \]  
(3)

where \( I \) is the \( 3 \times 3 \) identity matrix.

In our implementation, the Mobius transformation is a one-degree-of-freedom mapping, which maps unit vectors lying in the plane vertical to the unchanged condition column to unit vectors in the same plane. As any three-dimensional vector \( c \) in the plane can be parameterized by an angle \( \theta \) to reference vector \( c_2 \) where \( c_2 \) and \( c_3 \) are one pair of the orthogonal basis of the plane.
\[ c = \cos \theta c_2 + \sin \theta c_3 \]  
(4)
\[ c' = \cos \theta' c_2 + \sin \theta' c_3 \]  
(5)

Via a change of variable formula, the determinate of Jacobian can be calculated as:
\[ \frac{d\theta'}{d\theta} = \frac{d\theta' \, dc}{dc' \, d\theta} \]  
(6)

Given that \( c, c', c_2, c_3 \) are unit vectors, the absolute value of determinant of Jacobian is thus calculated as:
\[ |\det J| = \left| \frac{d\theta'}{d\theta} \right| = \left| \frac{dc'}{dc} \right| \]  
(7)

Where \( \frac{dc'}{dc} \) are given by:
\[ \frac{dc'}{d\theta} = \frac{dc'}{dc} \frac{dc}{d\theta} = \frac{dc'}{dc} \]  
(8)
\[ \frac{dc}{d\theta} = -\sin \theta c_2 + \cos \theta c_3 \]  
(9)

Affine Transformation The forward of quaternion affine transformation is given by:
\[ q' = g(q) = \frac{Wq}{\|Wq\|} \]  
(10)

The determinate of Jacobian of affine transformation is very straightforward and is given by:
\[ \det J(q) = \frac{\det W}{\|Wq\|^4} \]  
(11)

6.2. Inverse

Mobius Transformation The inverse of Mobius transformation is implemented by connecting \(-c'\) and parameters \( \omega \) with a straight line that intersects with \( S^3 \) at \( c \). The explicit expression for the inverse of Mobius transformation is given as follows:
\[ f_\omega^{-1}(c') = \frac{1 - \|\omega\|^2}{\|c' + \omega\|^2} (c' + \omega) + \omega \]  
(12)

The forward process and the inverse process have the same computational complexity.

However, as we utilize a linear combination, i.e. the weighted sum of angles to improve the expressivity of Mobius transformation, there is no analytical inverse for combined Mobius transformation, so we use binary search algorithms to find its inverse, as the combined angle \( \theta' \) are constrained within \((-\pi/2, \pi/2)\) and there’s no discontinuity around the boundary \( \pm \pi/2 \). The computational complexity for inverting the Mobius transformation is \( O(\log(1/\epsilon)) \), where \( \epsilon \) is the computational error.

Affine Transformation The inverse of affine transformation is implemented as:
\[ g^{-1}(q') = \frac{W^{-1}q'}{\|W^{-1}q'\|} \]  
(13)

Where \( W \) is a \( 4 \times 4 \) invertible matrix and \( q' \) is a quaternion. We find that the inverse and forward processes of
affine transformation have the same expression form while the invertible matrix $W$ in the forward path is replaced by its inverse matrix $W^{-1}$ in the inverse path. We attribute this feature to the geometry of affine transformation.

Via SVD decomposition, the affine transformation is divided into two types of operations: 1) rotation, 2) scaling and normalization. The inverse of rotation operation $R$ takes place by multiplying $R^{-1}$ which is straightforward, whereas the inverse of scaling and normalization is also implemented by taking the inverse of $S$ which sounds non-trivial.

The feature is caused by a geometry coincidence, as illustrated in Figure 5. $q$ are scaled to $\tilde{q}$ by multiplying $s_1, s_2, s_3, s_4$ on its coordinates and then normalized to the unit sphere $q'$. Inversely, $q'$ are scaled to $\tilde{q}'$ by $1/s_1, 1/s_2, 1/s_3, 1/s_4$ and normalized to $q$. $\Delta O\tilde{q}'q'$ and $\Delta Oq\tilde{q}$ are similar triangles as:

$$\frac{O\tilde{q}'}{O\tilde{q}} = \frac{Oq'}{Oq} = \frac{\tilde{q}'q}{q\tilde{q}}$$  \hspace{1cm} (14)

As $O, q', \tilde{q}$ are on the same line and $\angle q'Oq' = \angle qO\tilde{q}$ due to the property of similar triangles, $O, q', \tilde{q}$ are on the same line, thus normalized points $\tilde{q}'$ are $q$, which is the inverse of affine transformation.

7. ModelNet10-SO(3) and Pascal3D+ detailed results

Table 3 and Table 4 show per-category metrics for ModelNet10-SO(3) and Pascal3D+ dataset respectively. Note that our method achieves significantly lower median errors in experiments on ModelNet10-SO3 in all of the categories except the batutub category, where batutub images are well-known to exhibit severe symmetry and all the methods have unreasonably poor performance.

8. Analysis of Evaluation Methods

8.1. Log Likelihood Evaluation

In case of main experiments (rotation distribution fitting) and SYMSOL I/II datasets, we exploit log likelihood averaged over test data or test set annotations as evaluation metrics. It is implemented by transforming data through the flow to its base distribution and predicting log-likelihood.

8.2. Sample Generation

Sometimes pose estimation tasks require single-value prediction. Estimated rotations are generated via flowing samples from base distribution inversely and obtained a set of samples following target distribution. We pick the sample with the highest probability likelihood as our single prediction.

In SYMSOL I datasets, we present spread following IPDF [8]. Given a complete set of equivalent ground truth rotations, spread is defined as expectation of angular deviation to any of the ground truth values: $\mathbb{E}_{R \sim p(R|x)}[\min_{R' \in \{R_{GT}\}} d(R, R')]$, where $x$ is the given image and $d(R, R')$ is the relative angles between $R$ and $R'$. This measures how close the samples are to the ground truth poses. It is calculated as the mean of the relative angle between each of the generated samples and the ground truth poses closest to it.

In ModelNet10-SO(3) and Pascal3D+ datasets, we calculate the angular error of our single-value rotation prediction with single ground truth value, and report error median, accuracy at threshold 15° (for ModelNet-SO(3) only), 30°, i.e. Acc@15, Acc@30.

8.3. Ablations of other sample generation methods

We compare our inference method to those used in IPDF [8], which is implemented via first evaluating a grid $\{R_i\}$ or samples randomly sampled on SO(3), and solving

$$R_x^* = \arg \max_{R \in SO(3)} p(R|x)$$  \hspace{1cm} (15)

to pick the single pose prediction (We called the method PDF for short). The prediction can be made more accurate with gradient ascent of Equation (15) (Grad for short). The equi volumetric grids are generated first generating equal area grids on the 2-sphere with HEALPix method, and cover $SO(3)$ with Hopf fibration by sampling equi volumetric points on a circle with each point on the 2-sphere.

Table 5 shows that with few numbers of samples ($\sim 5$), our method is capable of capturing accurate results with little computational cost ($\sim 1-2$ minutes for evaluating the whole test sets). While for PDF and Grad, they require high-resolution grids to obtain high accuracy and the computation cost is not affordable when evaluating a large dataset.
9. Implementation Details

We use Adam as our optimizer with a learning rate of 1e-4.

**Details of $\sqrt{\frac{2}{7}}$ trick** In Mobius coupling layers, each $\omega$ is predicted by first predicting $\omega'$ by a [64, 64, 64, 64] multi-layer perceptron (MLP), ReLU as activation function and a residual connection between the first and the last layer with a layer perceptron (MLP), ReLU as activation function and a residual connection between the first and the last layer with a layer perceptron (MLP). It is then reparameterized by $\omega = \sqrt{\frac{2}{7}} \omega'$ to constrain it within $\sqrt{\frac{2}{7}}$ sphere.

**Unconditional Experiments** We use 24 layers of blocks (24 Mobius + 24 Affine) with the combination of 64 Mobius transformations at a batch size of 64 for 50k steps.

**SYMPSOL** In experiments on SYMSOL, we add a conditional Affine transformation at the beginning of our flow (near the target distribution) and use 21 layers of blocks (64-combination conditional Mobius coupling layers + unconditional affine transformation) at a batch size of 128 for 900k steps. Conditional features with dimension 512 are captured by an ImageNet pre-trained ResNet50 following settings in IPDF [8]. We trained a single model for SYMSOL I for 5 categories and trained a single for each category in SYMSOL also following IPDF.

**ModelNet10-SO3** For experiments on ModelNet10-SO3, 24 layers of blocks (64-combination conditional Mobius coupling layers + conditional affine transformation) at a batch size of 128 for 900k steps.

---

<table>
<thead>
<tr>
<th>Acc@15°</th>
<th>Deng et al. [1]</th>
<th>0.562</th>
<th>0.140</th>
<th>0.788</th>
<th>0.800</th>
<th>0.345</th>
<th>0.563</th>
<th>0.708</th>
<th>0.279</th>
<th>0.733</th>
<th>0.440</th>
<th>0.832</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prokudin et al. [9]</td>
<td>0.456</td>
<td>0.114</td>
<td>0.822</td>
<td>0.662</td>
<td>0.023</td>
<td>0.406</td>
<td>0.704</td>
<td>0.187</td>
<td>0.590</td>
<td>0.108</td>
<td>0.946</td>
</tr>
<tr>
<td></td>
<td>Mohlin et al. [7]</td>
<td>0.693</td>
<td>0.322</td>
<td>0.882</td>
<td>0.881</td>
<td>0.536</td>
<td>0.682</td>
<td>0.790</td>
<td>0.516</td>
<td>0.919</td>
<td>0.446</td>
<td>0.957</td>
</tr>
<tr>
<td></td>
<td>IPDF [8]</td>
<td>0.719</td>
<td>0.392</td>
<td>0.877</td>
<td>0.874</td>
<td>0.615</td>
<td>0.687</td>
<td>0.799</td>
<td>0.567</td>
<td>0.914</td>
<td>0.523</td>
<td>0.945</td>
</tr>
<tr>
<td></td>
<td>Ours(Uni.)</td>
<td>0.760</td>
<td>0.402</td>
<td>0.896</td>
<td>0.927</td>
<td>0.704</td>
<td>0.753</td>
<td>0.843</td>
<td>0.602</td>
<td>0.939</td>
<td>0.561</td>
<td>0.975</td>
</tr>
<tr>
<td></td>
<td>Ours(Fisher)</td>
<td>0.744</td>
<td>0.439</td>
<td>0.890</td>
<td>0.909</td>
<td>0.638</td>
<td>0.715</td>
<td>0.810</td>
<td>0.585</td>
<td>0.938</td>
<td>0.535</td>
<td>0.978</td>
</tr>
</tbody>
</table>

| Acc@30° | Deng et al. [1] | 0.694 | 0.325 | 0.880 | 0.908 | 0.556 | 0.649 | 0.807 | 0.466 | 0.902 | 0.485 | 0.958 |
|---------| Prokudin et al. [9] | 0.528 | 0.175 | 0.847 | 0.777 | 0.061 | 0.500 | 0.788 | 0.306 | 0.673 | 0.183 | 0.972 |
|         | Mohlin et al. [7] | 0.757 | 0.403 | 0.908 | 0.935 | 0.674 | 0.739 | 0.863 | 0.614 | 0.944 | 0.511 | 0.981 |
|         | IPDF [8] | 0.735 | 0.410 | 0.883 | 0.917 | 0.629 | 0.688 | 0.832 | 0.570 | 0.921 | 0.531 | 0.967 |
|         | Ours(Uni.) | 0.777 | 0.419 | 0.904 | 0.946 | 0.722 | 0.766 | 0.868 | 0.617 | 0.948 | 0.567 | 0.982 |
|         | Ours(Fisher) | 0.768 | 0.460 | 0.898 | 0.934 | 0.694 | 0.738 | 0.859 | 0.615 | 0.948 | 0.544 | 0.987 |

**Table 3.** Numerical results of rotation regression on ModelNet10-SO3 dataset. We adopt $15^\circ$ accuracy, $30^\circ$ accuracy and median error as the evaluation metrics. The best performance is shown in **bold** and the second best is with *underlined.*

<table>
<thead>
<tr>
<th>Avg.</th>
<th>Liao et al. [5]</th>
<th>0.819</th>
<th>0.82</th>
<th>0.77</th>
<th>0.55</th>
<th>0.93</th>
<th>0.95</th>
<th>0.94</th>
<th>0.85</th>
<th>0.61</th>
<th>0.80</th>
<th>0.95</th>
<th>0.83</th>
<th>0.82</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mohlin et al. [7]</td>
<td>0.825</td>
<td>0.90</td>
<td>0.85</td>
<td>0.57</td>
<td>0.94</td>
<td>0.95</td>
<td>0.96</td>
<td>0.78</td>
<td>0.62</td>
<td>0.87</td>
<td>0.85</td>
<td>0.77</td>
<td>0.84</td>
</tr>
<tr>
<td></td>
<td>Prokudin et al. [9]</td>
<td>0.838</td>
<td>0.89</td>
<td>0.83</td>
<td>0.46</td>
<td>0.96</td>
<td>0.93</td>
<td>0.90</td>
<td>0.80</td>
<td>0.76</td>
<td>0.90</td>
<td>0.90</td>
<td>0.82</td>
<td><strong>0.91</strong></td>
</tr>
<tr>
<td></td>
<td>Tulsiani &amp; Malik [10]</td>
<td>0.808</td>
<td>0.81</td>
<td>0.77</td>
<td>0.59</td>
<td>0.93</td>
<td>0.98</td>
<td>0.89</td>
<td>0.80</td>
<td>0.62</td>
<td>0.88</td>
<td>0.82</td>
<td>0.80</td>
<td>0.80</td>
</tr>
<tr>
<td></td>
<td>Mahendran et al. [6]</td>
<td>0.859</td>
<td>0.87</td>
<td>0.81</td>
<td><strong>0.64</strong></td>
<td>0.96</td>
<td>0.97</td>
<td>0.95</td>
<td>0.92</td>
<td>0.87</td>
<td>0.95</td>
<td>0.97</td>
<td>0.82</td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td>IPDF [8]</td>
<td>0.837</td>
<td>0.81</td>
<td>0.85</td>
<td>0.56</td>
<td>0.93</td>
<td>0.95</td>
<td>0.94</td>
<td>0.87</td>
<td><strong>0.78</strong></td>
<td>0.85</td>
<td>0.88</td>
<td>0.78</td>
<td>0.86</td>
</tr>
<tr>
<td></td>
<td>Ours(Uni.)</td>
<td>0.827</td>
<td>0.83</td>
<td>0.78</td>
<td>0.56</td>
<td>0.95</td>
<td>0.96</td>
<td>0.93</td>
<td>0.87</td>
<td>0.62</td>
<td>0.85</td>
<td>0.90</td>
<td>0.81</td>
<td>0.86</td>
</tr>
<tr>
<td></td>
<td>Ours(Fisher)</td>
<td><strong>0.863</strong></td>
<td><strong>0.89</strong></td>
<td><strong>0.89</strong></td>
<td>0.55</td>
<td><strong>0.96</strong></td>
<td><strong>0.98</strong></td>
<td><strong>0.95</strong></td>
<td><strong>0.94</strong></td>
<td><strong>0.97</strong></td>
<td><strong>0.95</strong></td>
<td><strong>0.92</strong></td>
<td><strong>0.95</strong></td>
<td><strong>0.85</strong></td>
</tr>
</tbody>
</table>

**Table 4.** Numerical results of rotation regression on Pascal3D+ dataset. We adopt $30^\circ$ accuracy and median error as the evaluation metrics. The best performance is shown in **bold** and the second best is with *underlined.*

---

avg. bathtb bed chair desk dresser tv n. stand sofa table toilet
Table 5. **Abalations on evaluation time.** We report results on conditional rotation regression tasks of ModelNet10-SO3. We compare the time required to generate estimated rotation via inversing the flow (Ours), inferring the probability of a grid (PDF), gradient descent of probability of a grid (Grad). The number of samples used in PDF and Grad correspond to the HEALPix-SO(3) grids of levels 2, 3 and 4 respectively.

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of samples</th>
<th>evaluation time (total) ↓</th>
<th>evaluation time (s/iter) ↓</th>
<th>Acc@15° ↑</th>
<th>Acc@30° ↑</th>
<th>Med. (°) ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours</td>
<td>1</td>
<td>1.2 min</td>
<td></td>
<td>0.508</td>
<td>0.740</td>
<td>0.760</td>
</tr>
<tr>
<td>Ours</td>
<td>5</td>
<td>1.3 min</td>
<td></td>
<td>0.525</td>
<td>0.759</td>
<td>0.773</td>
</tr>
<tr>
<td>Ours</td>
<td>50</td>
<td>1.4 min</td>
<td></td>
<td>0.555</td>
<td>0.759</td>
<td>0.773</td>
</tr>
<tr>
<td>Ours</td>
<td>500</td>
<td>4.8 min</td>
<td></td>
<td>1.97</td>
<td>0.760</td>
<td>0.773</td>
</tr>
<tr>
<td>Ours</td>
<td>5k</td>
<td>0.7 h</td>
<td></td>
<td>16.8</td>
<td>0.760</td>
<td>0.773</td>
</tr>
<tr>
<td>PDF</td>
<td>4.6k</td>
<td>0.6 h</td>
<td></td>
<td>0.122</td>
<td>0.480</td>
<td>0.611</td>
</tr>
<tr>
<td>PDF</td>
<td>37k</td>
<td>1.4 h</td>
<td></td>
<td>0.289</td>
<td>0.699</td>
<td>0.745</td>
</tr>
<tr>
<td>PDF</td>
<td>295k</td>
<td>9.5 h</td>
<td></td>
<td>1.88</td>
<td>0.755</td>
<td>0.771</td>
</tr>
<tr>
<td>Grad</td>
<td>4.6k</td>
<td>2.2h</td>
<td></td>
<td>0.446</td>
<td>0.496</td>
<td>0.612</td>
</tr>
<tr>
<td>Grad</td>
<td>37k</td>
<td>3.0h</td>
<td></td>
<td>0.610</td>
<td>0.704</td>
<td>0.748</td>
</tr>
<tr>
<td>Grad</td>
<td>295k</td>
<td>11.0h</td>
<td></td>
<td>2.20</td>
<td>0.758</td>
<td>0.772</td>
</tr>
</tbody>
</table>

batch size of 128 for 250k steps are used. We also use the strategy of learning rate decay and multiply the learning rate by 0.1 at 30 and 40 epochs. Conditional features with dimension 2048 are captured by an ImageNet pre-trained ResNet101 following settings in Mohlin et al. [7]. We train a single model for the whole dataset and use a 32-dimensional positional embedding for 10 categories.

**Pascal3D+** For Pascal3D+, 24 layers of blocks (64-combination conditional Mobius coupling layers + conditional affine transformation) at a batch size of 128 for 350k steps are used. We also use the strategy of learning rate decay and multiply the learning rate by 0.1 at 200k and 250k iterations when using uniform distribution as base distribution, 12 and 15 epochs when using pre-trained fisher as base distribution respectively. Conditional features with dimension 2048 are captured by an ImageNet pre-trained ResNet101. We train a single model for the whole dataset and use a 32-dimensional positional embedding for 10 categories.

**References**


