Delving into Discrete Normalizing Flows on SO(3) Manifold for Probabilistic Rotation Modeling

Yulin Liu* Haoran Liu* Yingda Yin* Yang Wang Baoquan Chen† He Wang†
Peking University

Abstract

Normalizing flows (NFs) provide a powerful tool to construct an expressive distribution by a sequence of trackable transformations of a base distribution and form a probabilistic model of underlying data. Rotation, as an important quantity in computer vision, graphics, and robotics, can exhibit many ambiguities when occlusion and symmetry occur and thus demands such probabilistic models. Though much progress has been made for NFs in Euclidean space, there are no effective normalizing flows without discontinuity or many-to-one mapping tailored for $\mathrm{SO}(3)$ *manifold. Given the unique non-Euclidean properties of the rotation manifold, adapting the existing NFs to* $\mathrm{SO}(3)$ *manifold is non-trivial. In this paper, we propose a novel normalizing flow on* $\mathrm{SO}(3)$ *by combining a Mobius transformation-based coupling layer and a quaternion affine transformation. With our proposed rotation normalizing flows, one can not only effectively express arbitrary distributions on* $\mathrm{SO}(3)$*, but also conditionally build the target distribution given input observations. Extensive experiments show that our rotation normalizing flows significantly outperform the baselines on both unconditional and conditional tasks.*

1. Introduction

Endowing a neural network with the ability to express uncertainty along with the prediction is of crucial influence to safety and interpretability-critical systems and provides valuable information for downstream tasks [4, 19, 32]. As a widely used technique in computer vision and robotics, rotation regression can also benefit from such uncertainty-aware predictions and enable many applications [5, 14, 31].

To this end, recent years have witnessed much effort in modeling the uncertainty of rotation via probabilistic modeling of the $\mathrm{SO}(3)$ space, including von Mises distribution for Euler angles [27], Bingham distribution for quaternions [6, 13], matrix Fisher distribution for rotation matrices [24], etc. Those distributions are all single-modal, which fall short on modeling objects with continuous symmetry, which are ubiquitous in our daily life. Taking *cup* as an example, it exhibits rotational symmetry for which modeling with the unimodal or the mixture of distributions is clearly insufficient. How to model an *arbitrary* distribution on $\mathrm{SO}(3)$ manifold is still a challenging open problem.

Normalizing flows [28], which maps samples from a simple base distribution to the target distributions via invertible transformations, provides a flexible way to express complex distributions and has been widely used in expressing arbitrary distributions in Euclidean space [1, 2, 7, 8, 16, 17]. However, developing normalizing flows on $\mathrm{SO}(3)$ manifold is still highly under-explored.

Some works rely on normalizing flows in Euclidean space and adapt them to handle rotations. ReLie [9] proposes normalizing flows for general Lie group via Lie algebra in Euclidean space. However, it suffers from discontinuous rotation representations [37] and leads to inferior performance. ProHMR [18] considers rotations as 6D vectors in Euclidean space and leverages Glow [17] to model distributions, where a many-to-one Gram-Schmidt projection is needed to close the gap between the two spaces. Although composed of bijective transformations in Euclidean space, the many-to-one mapping from Euclidean space to $\mathrm{SO}(3)$ breaks the one-to-one regulation of normalizing flows [28].

Other works propose general normalizing flows for non-Euclidean spaces. Mathieu et al. [23], Lou et al. [21] and Falorsi et al. [10] propose continuous normalizing flows for general Riemannian manifold, without considering any property of $\mathrm{SO}(3)$ space, which leads to unsatisfactory performance for probabilistic rotation modeling. Rezende et al. [29] introduce normalizing flows on tori and spheres. Note that despite unit quaternions lying on $\mathcal{S}^3$ space, [29] does not exhibit the antipodal symmetry property of quaternions and thus is not suitable for modeling rotations in $\mathrm{SO}(3)$.

In this work, we introduce novel discrete normalizing flows for rotations on the $\mathrm{SO}(3)$ manifold. The core building block of our discrete rotation normalizing flow consists of a Mobius coupling layer with rotation matrix representa-

*Equal contribution
†He Wang and Baoquan Chen are the corresponding authors ({hewang, baoquan}@pku.edu.cn).

tion and an affine transformation with quaternion representation, linked by conversions between rotations and quaternions. In the Mobius coupling layer, one column of the rotation matrix acts as the *conditioner*, remaining unchanged. Another column serves as the *transformer*, undergoing Mobius transformation conditioned on the conditioner, while the remaining column is determined by the cross-product. By combining multiple Mobius coupling layers, we enhance the capacity of our vanilla design for rotation normalizing flows.

To further increase expressivity, we propose an affine transformation in quaternion space. This quaternion affine transformation complements the Mobius coupling layer, functioning as both a global rotation and a means for condensing or dilating the local likelihood. Despite quaternions being a double coverage of the $\mathrm{SO}(3)$ manifold, this transformation remains bijective and diffeomorphic to $\mathrm{SO}(3)$.

We conduct extensive experiments to validate the expressivity and stability of our proposed rotation normalizing flows. The results show that our rotation normalizing flows are able to either effectively fit the target distributions on $\mathrm{SO}(3)$ with distinct shapes, or regress the target distribution given input image conditions. Our method achieves superior performance on both tasks over all the baselines.

2. Related Work

Normalizing flows on Euclidean space Most of the Normalizing flows are constructed in Euclidean space. Many of them are constructed using coupling layers [7, 8], and Glow [17] improves it using 1×1 convolution for flexible permutation. Flow++ [16] combines multiple cumulative distribution functions to make the transformation in the coupling layer more expressive. Invertible ResNet [1] and Residual Flow [2] propose residual flow which is more flexible and is also possible to be extended to $\mathrm{SO}(3)$. Neural ODE [3] and FFJORD [15] treat the transformation as a continuous movement of the vectors with ordinary differential equation (ODE), and RNODE [11] further improves it by adding constraints to smooth the dynamics.

Normalizing flows on non-Euclidean, SO(3)-relevant manifolds ReLie [9] and ProHMR [18] both perform normalizing flow on the Euclidean space and then use a map from $\mathbb{R}^N$ to $\mathrm{SO}(3)$, however, they suffer from either discontinuity or infinite-to-one mapping due to $\mathrm{SO}(3)$'s special topological structure as shown in Supplementary Material. Rezende et.al. [29] propose three methods to construct flow on tori and sphere: Mobius transformation, circular splines, and non-compact projection, and here we use Mobius transformation to build our flow on $\mathrm{SO}(3)$. Also, Mathieu et al. [23], Lou et al. [21] and Falorsi et al. [10] extend continuous normalizing flow to Riemann manifold by parameterizing the "velocity" in the tangent space and this can also be applied to $\mathrm{SO}(3)$. Moser Flow [30] further improves it by parametrizing the density as the prior density minus the divergence of a learned neural network, however, the model's effectiveness is limited when density is too high or too low, as it models probability instead of log probability.

Distributions for rotation Several works leverage probabilistic distributions on $\mathrm{SO}(3)$ for the purpose of rotation regression. Prokudin et al. [27] use the mixture of von Mises distributions over Euler angles. Gilitschenski et al. [13] and Deng et al. [6] utilize Bingham distribution over quaternion to jointly estimate a distribution over all axes.

Mohlin et al. [24] leverage matrix Fisher distribution for deep rotation regression with unconstrained Euclidean parameters. Different from the parametric distributions above, Murphy et al. [25] represents distributions implicitly by neural networks, where it predicts unnormalized log probability first and

then normalize it by discretization sampling on $\mathrm{SO}(3)$. In this work, we use normalizing flows to directly generate normalized distributions.

3. Normalizing Flows on Riemannian Manifold

Normalizing flows (NFs) provide a flexible way to construct complex distributions in high-dimensional Euclidean space by transforming a base distribution through an invertible and differential mapping. Base distributions are often chosen to be easily evaluated and sampled from, like gaussian distribution in Euclidean space. NFs can be extended to Riemannian manifolds embedded in a higher dimensional space [12, 26]. Formally, normalizing flows transform base distributions $\pi(u), u \in \mathcal{M}$ to target distributions $p(x), x \in \mathcal{N}$, where $\mathcal{M}, \mathcal{N}$ are Riemannian manifold and have the same topology, via diffeomorphisms $T : \mathcal{M} \to \mathcal{N}$. The probability density function(pdf) of x can be calculated by change of variable formulas:

$$p(x) = \pi(T^{-1}(x))|\det J_{T^{-1}}(x)|, \tag{1}$$

where Jacobian matrix $J_{T^{-1}}(x) = \frac{\partial(T^{-1}(x))}{\partial x}$ is the $D \times D$ partial derivatives of T^{-1} at x.

As diffeomorphisms are composable, in practice, the transformation T is often implemented via a sequence of simple transformations $T = T_K \circ \cdots \circ T_2 \circ T_1$, whose Jacobian determinants are easy to evaluate. The determinant of the composed transformation is given by:

$$\det(J_{T^{-1}}(x)) = \Pi_{i=1}^{k} \det(J_{T_i^{-1}}(T_K^{-1} \circ \cdots \circ T_{i+1}^{-1}(x))) \tag{2}$$

Normalizing flows enable both forward and inverse processes and one can calculate $p(x)$ through the process. We can fit the target distribution by minimizing the negative log-likelihood (NLL) of training data in the inverse process and sample via mapping samples from the base distribution in the forward process.

4. Method

Our normalizing flows comprise two key building modules that operate on the $\mathrm{SO}(3)$ manifold. First, we utilize a Mobius coupling layer that interprets the rotation matrix as three orthonormal 3-vectors, with one vector held fixed and the remaining part transformed using the Mobius transformation with the fixed vector as condition.

Second, we employ a quaternion affine transformation that uses the quaternion representation while retaining antipodal symmetry, making it a diffeomorphism on real projective space $\mathbb{RP}^3$. Affine transformation resembles the 1×1 convolution in Glow [17], but is applied to the quaternion representations rather than the channels.

Mobius coupling layers are generally effective at modeling various distributions, while affine transformations are more suitable for uni-modal distributions. By combining these two building blocks, we can construct more powerful normalizing flows that perform better and converge faster. Our model consists of multiple blocks of layers, each of which comprises a Mobius coupling layer and a quaternion affine transformation. Figure 1 illustrates our model.

4.1. Mobius Coupling Layer

Mobius Transformation is defined on a D-dimensional sphere $\mathcal{S}^D$. Rezende et al. [29] have applied it to build expressive normalizing flows on the 2D circle $\mathcal{S}^1$. However, with the unique topology of $\mathrm{SO}(3)$, it's non-trivial to apply Mobius transformation to $\mathrm{SO}(3)$ manifold. We present a coupling layer method that fully utilizes the geometry of $\mathrm{SO}(3)$ and provides a $\frac{\sqrt{2}}{2}$ trick to solve the discontinuity encountered in combining multiple transformations.

Revisit Mobius transformation Mobius Transformation on $\mathcal{S}^D$ can be parameterized by an $\omega \in \mathbb{R}^{D+1}$ satisfying $\|\omega\| < 1$. For a point $c \in \mathcal{S}^D$, Mobius transformation f_ω is defined as:

$$f_\omega(c) = \frac{1 - \|\omega\|^2}{\|c - \omega\|}(c - \omega) - \omega \tag{3}$$

This transformation has a very straightforward geometric meaning: first extend the line $c\omega$ and find the intersection point between it and the sphere $\tilde{c}$, then output the point symmetric to $\tilde{c}$ about the origin $c' = -\tilde{c}$, as shown in Figure 3 Left. When ω is at the origin, f_ω becomes identity transformation; when ω is not at the origin, f_ω concentrates part away from ω; and when ω is near to surface of unit sphere $\mathcal{S}^D$, f_ω maps almost all points on $\mathcal{S}^D$ to nearing of $-\omega$.

Mobius coupling layer A 3×3 rotation matrix $R \in \mathrm{SO}(3)$ satisfies $RR^T = I$ and $\det R = +1$. It thus can be expressed as three orthonormal vectors $[c_1, c_2, c_3]$ that satisfy $||c_i|| = 1$ (meaning $c_i \in \mathcal{S}^2$) and $c_i \cdot c_j = 0$ for all $i \neq j$.

To build a normalizing flow on $\mathrm{SO}(3)$, we thus consider applying the idea of *coupling* layer introduced in [7, 8] to the orthonormal vectors. In each *coupling* layer, the input x is divided into the *conditioner* x_1 that remains unchanged after the flow and the *transformer* x_2 that changes according to the condition part, which can be written as $x'_1 = x_1, x'_2 = f(g(x_1), x_2)$. As the calculation of the Jacobian determinant only involves $\frac{\partial f}{\partial x_2}$, g can be arbitrarily complex which enables high expressivity and here we use a neural network to parameterize it.

We utilize a similar structure to build Mobius transformation for rotation matrices. We divide a rotation matrix into 2 parts, the *conditioner* is c_i $(i = 1, 2, 3)$ and the *transformer* is the rest two columns $\{c_j \mid j \neq i\}$. Taking $i = 1$ as an example: conditioning on c_1, we can transform c_2 to c'_2. Then c'_3 is already determined by $c'_3 = c'_1 \times c'_2$where $c'_1 = c_1$. The coupling layer needs to ensure that: 1) $||c'_2|| = 1$, *i.e.* $c'_2 \in \mathcal{S}^2$; and 2) c'_2 is orthogonal to c'_1.

Given condition 1), we thus consider using a Mobius transformation on $\mathcal{S}^1$ to transform c_2. To further meet condition 2), we notice that all valid c_2 and c_3 form a plane P that passes the origin and is perpendicular to c_1. After the transformation, c'_2 needs to stay in P. This can be achieved by constraining ω inside P. Therefore, we propose to learn a neural network that maps the condition c_1 to $\mathbb{R}^3$ and then projects it to P, as shown below:

$$\omega = \omega' - c_1(c_1 \cdot \omega') \tag{4}$$

where ω' is the unconstrained parameters generated by the neural network. The structure of our Mobius coupling layer is illustrated in Figure 2 and Figure 3 Middle.

Note that, given c_1, there is only 1 Degree of Freedom left for the rest two columns. So, our Mobius coupling layer is essentially rotating c_2 and c_3 about c_1 simultaneously by an angle θ conditioned on $c_1$and c_2.

Linear combination of multiple Mobius transformations To further increase the expressivity of the Mobius transformation, we leverage linear combination of Mobius transformation presented in [29]. It is done by first transforming c_2 into $\{c'_{2i}\}$ using a set of ωs, $\{\omega_i\}$, and then calculate the weighted sum $\theta' = \sum w_i\theta_i$, where w_i is the weight generated by a neural network conditioned on c_1 and normalized with softmax, and θ_i is the angle between c'_{2i} and c_2. The result of combined transformation c'_2 can then be calculated by rotating c_2 with θ'.

However, such naive implementation has the problem of discontinuity. Take two combination points with weights [0.5, 0.5] for example. Assume θ_1 is $30°$, θ_2 is $-178°$, the combined angle θ is $-74°$. However, when θ_2 slightly changed to $-182°$ that is $178°$ as $\theta \in [-\pi, \pi]$, the combined angle θ becomes $104°$. This discontinuity of slight change of θ_i resulting in a huge jump in combined θ can reduce the networks' performance and add difficulties in learning.

$\frac{\sqrt{2}}{2}$ trick We present a $\frac{\sqrt{2}}{2}$ trick to alleviate this discontinuity. If $\|\omega\|$ is small, c'_2 will be close to c_2. And if all $\{c'_{2i}\}$

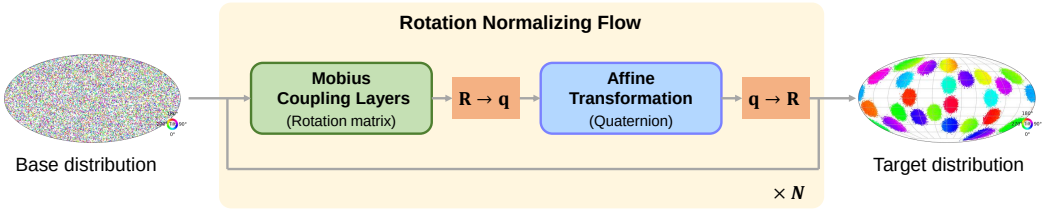

Figure 1. **Pipeline overview.** Our flow model takes rotations as input and outputs transformed rotations and log determinants of Jacobian, transforming a base distribution to a target one. Our flow is done by iteratively alternating Mobius coupling on Rotation matrix representation and affine transformation on quaternion for N times. For probability inference, data are fed into the flow to the corresponding rotation which is of base distribution and predicts log-likelihood; while in the sampling process, our flow runs inversely, generating new data by transforming samples from the base distribution. The distribution visualization is borrowed from [25] where $\mathrm{SO}(3)$ is projected to a 2D sphere by Hopf fibration, points on the 2D sphere indicate the direction of a canonical z-axis, the colors represent the tilt angle about that axis, the direction of a canonical z-axis and the sizes of points show the probability density.

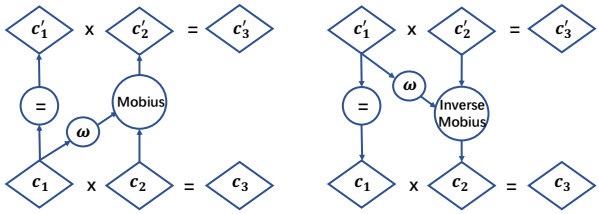

Figure 2. **Computational graphs of Mobius coupling layer.** c_1, c_2, c_3 indicate different columns for rotation matrix and c_1', c_2', c_3' indicate the columns for transformed rotation matrix. Mobius coupling layer applies the Mobius transformation to column c_2 conditioned on the unchanged column c_1.

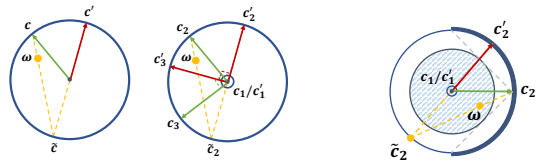

Figure 3. **Left**: Geometric illustration of Mobius transformation. The transformation is implemented by first connecting a straight line between c and ω which intersects the sphere with $\tilde{c}$, and the result c' is the opposite point of $\tilde{c}$. **Middle**: Geometric illustration of Mobius coupling layer. This sketch is viewed parallel to the unchanged c_1. An MLP conditioned on c_1 gives output ω' which is then projected to ω orthogonal to c_1. c_2 is then transformed into c_2' via Mobius transformation with parameter ω. c_3' is then computed by cross product of $c_1' = c_1$ and c_2'. **Right**: Illustration of $\frac{\sqrt{2}}{2}$ trick. ω are restricted in the blue circle and resulting in $\{c_{2i}'\}$ on the semi-circle of blue lines. Therefore the combined c_2' are restricted on the semi-circle of blue lines. Thus slight change of $\{\theta_i\}$ won't cause jump (about π) in θ, solving the discontinuity.

lie on a small neighborhood of c_2, there won't be discontinuity issues as circle is locally linear, whereas the expressiveness will be limited if $\{c_{2i}'\}$ are too close to c_2. It can be shown that the biggest range of c_2' with no discontinuity in combination is a semi-circle whose center is c_2 and the corresponding range of $\|\omega\|$ is $[0, \frac{\sqrt{2}}{2})$ as shown in Figure 3 Right.

By linear combination, the inverse of transformation can't be calculated analytically, we alleviate binary search presented in [29] to find the inverse as $\theta \in (-\pi/2, \pi/2)$. Though the restriction of $\|\omega\|$ may reduce the expressivity of our flows, the avoidance of discontinuity stabilizes our network so in general it is beneficial. (See Supplementary Material for details)

4.2. Quaternion Affine Transformation

Bijective transformation on $\mathrm{SO}(3)$ manifold Quaternion is another representation of rotation, defined as a unit vector on the 4D sphere. Let θ be the angle of the rotation and (x, y, z) be the axis of rotation, then $\mathbf{q}$ can be computed as $(\cos\frac{\theta}{2}, x\sin\frac{\theta}{2}, y\sin\frac{\theta}{2}, z\sin\frac{\theta}{2})$.

However, quaternion representation of $\mathrm{SO}(3)$ has the topology of antipodal symmetry, meaning that $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation R. To be bijective on $\mathrm{SO}(3)$, transformation on quaternion should keep antipodal symmetry, i.e. transform $-\mathbf{q}$ to $-f(\mathbf{q})$. Our proposed affine transformation satisfies such requirement and is thus a bijective transformation on $\mathrm{SO}(3)$.

Affine transformation

Our proposed quaternion affine transformation consists of a linear transformation of 4D vectors $\mathbf{q}$ followed by a projection to the unit sphere $\mathcal{S}^3$. The explicit expression for this transformation is as follows:

$$g(\mathbf{q}) = \frac{W\mathbf{q}}{\|W\mathbf{q}\|} \tag{5}$$

where W is a 4×4 invertible matrix, $\mathbf{q}$ is the quaternion representation of a rotation. The inverse of the transformation can be calculated by simply replacing W by its inverse

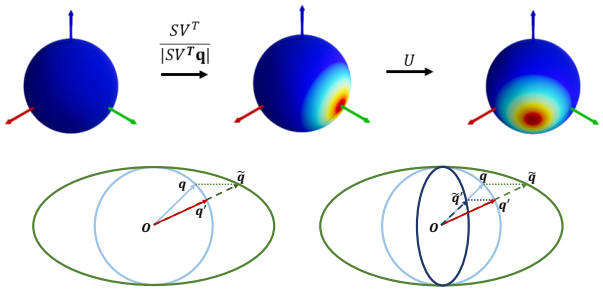

Figure 4. **Illustration of quaternion affine transformation. Top**: Effect of quaternion affine transformation. Let e_1, e_2, e_3, e_4 correspond to the standard basis of $\mathcal{S}^3$, i.e. w, x, y, z unit vectors. The affine transformation first rotates $\mathbf{q}$ by V^T, followed by the scaling and normalizing part which concentrates points near the axis $V^T e_i$ with large s_i and expands the others, and followed by a rotation U. **Bottom**: Forward and inverse sketch of scaling and normalization part in affine transformation. The sketch is viewed in 2D $w-x$ section. $\mathbf{q}$ is transformed to $\tilde{\mathbf{q}}$ by multiplying scaling factors $s_1 = 2, s_2 = 1, s_3 = 1, s_4 = 1$, and then normalized to $\mathbf{q}'$. The inverse transformation is similar to forward process with $s'_1 = 1/s_1, s'_2 = 1/s_2, s'_3 = 1/s_3, s'_4 = 1/s_4$. This feature is closely related to the geometry of affine transformation of an ellipse and more discussions are given in Supplementary Material.

W^{-1}. The transformation looks similar to the 1×1 convolution in Glow [17], in which an invertible matrix is multiplied on picture channels. We present a geometric explanation for its effect and explain the meaning of its name.

Geometric explanation of affine transformation We name this transformation *affine*, since it resembles the affine transformation in Eucliean space $f = ax+b$, where a is the scaling parameter, and b is a displacement term. By SVD decomposition $W = USV^T$, the 4×4 invertible matrix can be decomposed into an orthogonal matrix U, multiplied by a diagonal matrix S and another orthogonal matrix V^T. Multiplying U, V^T globally rotates the distribution pattern and acts as a displacement on $\mathrm{SO}(3)$ manifold, while the diagonal matrix S serves as the scaling term.

As multiplying an orthogonal matrix will not change the length of a vector, the term $\|W\mathbf{q}\|$ is equal to $\|S(V^T\mathbf{q})\|$, so affine transformation can be decomposed into:

$$g(\mathbf{q}) = U\frac{S}{\|S(V^T\mathbf{q})\|}(V^T\mathbf{q}) \tag{6}$$

This can be seen as a 4-step transformation, first rotate $\mathbf{q}$ to $V^T\mathbf{q}$, then multiply each coordinate by scaling factors $s_i, (i = 1, \cdots, 4)$ and normalize it to a unit vector, and finally rotate the quaternion with U. An illustration of quaternion affine transformation is shown in Figure 4 Top.

We present 2 methods to parameterize 4×4 invertible matrix W. The first one is to use an unconstrained 4×4 matrix as the probability for a non-invertible matrix is near zero. It's numerically stable and safe in our implementation. We also tried LU decomposition $W = PL(U + S)$ as presented in [17], where P is a fixed permutation matrix, L is a lower triangular matrix with ones on the diagonal, U is an upper triangular matrix with zeros on the diagonal, and S is a vector with non-zero coordinate. See more discussions in Supplementary Material.

Why rotation U, V is needed? Our Mobius coupling layer allows distribution to flow in the vertical plane of c_i, however, it is very difficult to learn a global rotation of distributions on $\mathrm{SO}(3)$. The introduced rotation operation in quaternion space exactly alleviates this problem. Rotating quaternions also serve as a generalization of permutation in splitting condition or transformed columns of rotation matrix, which has similar effects to the 1×1 Convolution introduced in Glow [17].

Why scaling and normalization? Multiplying diagonal matrix S results in multiplying coordinate of a quaternion $\mathbf{q} = (w, x, y, z)$ by scaling s_1, s_2, s_3, s_4,

$$(w, x, y, z) \rightarrow (s_1 w, s_2 x, s_3 y, s_4 z) \tag{7}$$

transforming the unit sphere to an ellipsoid, as shown in Figure 4 Bottom. A point $\mathbf{q}$ on the sphere $\mathcal{S}^3$ is transformed to a point on the oval $\tilde{\mathbf{q}}$. It is followed by a projection to sphere: term $1/\|S\tilde{\mathbf{q}}\|$ normalizes the transformed vector. The final point $\mathbf{q}'$ is the intersected point of $O - \tilde{\mathbf{q}}$ on the sphere $\mathcal{S}^3$.

The explicit expression for scaling and normalization is:

$$f_{(s_1,s_2,s_3,s_4)}(w, x, y, z) = \frac{(s_1 w, s_2 x, s_3 y, s_4 z)}{||(s_1 w, s_2 x, s_3 y, s_4 z)||} \tag{8}$$

When $s_1 = s_2 = s_3 = s_4$, the scaling and normalization transformation is identity, otherwise it concentrates probability to the axis with large s. Scaling and normalization can create a high peak by transforming $\mathcal{S}^3$ to an elongated ellipsoid and then projecting back to $\mathcal{S}^3$.

4.3. Interchange of Rotation Matrix and Quaternion Representation

We iteratively compose Mobius coupling layer and quaternion affine transformation to build our flow model and switch between rotation matrix representation and quaternion representation in the process.

The composed flows perform better and learn faster, as the quaternion affine transformation makes up for some problems in Mobius coupling layer, and increases its expressiveness. The rotation operation in quaternion affine transformation allows a global rotation of distributions on $\mathrm{SO}(3)$, and serves as a generalization of permutation in splitting condition or transformed columns of rotation matrix. The scaling and normalization operation makes it possible to quickly create high peak distributions. This allows for quick concentrate distribution to target predictions and accelerates the convergence concerning rotation regression.

4.4. Conditional normalizing flow

There are cases when we need to infer a distribution depending on condition, for example when we need to infer the rotation of a symmetric or occluded object in an image. Our flow can be easily extended to be conditional using methods in [34] as we can simply concatenate the condition with fixed columns to generate parameters for Mobius flow, and use a neural network to output the matrix W in quaternion affine transformation. Moreover, apart from using uniform distribution as base distribution, we can pretrain a neural network to fit the target distribution using commonly-used unimodal distributions (e.g. Matrix Fisher distribution [24]) as the base distribution. We show examples of our conditional normalizing flow on learning multi-mode rotation distribution from symmetric object images and on predicting single pose given images(see Experiments for detail). The conditional features can be computed by a ResNet or other CNN network in this task.

5. Experiments

In this section, we conduct multiple experiments to validate the capacity of our proposed normalizing flows to model distributions on $\mathrm{SO}(3)$. We train all experiments with negative log-likelihood (NLL) loss. The invertible matrix W for quaternion affine transformation is parameterized as a 4×4 unconstrained matrix. Implementation details and results of LU decomposition parameterization for W are reported in Supplementary Material.

5.1. Learning to Model Various Distributions

As in common, we first evaluate and compare our normalizing flows and baseline methods by learning to fit distributions on $\mathrm{SO}(3)$ with distinct properties.

Datasets We design four challenging distributions on $\mathrm{SO}(3)$: a very sharp single modal distribution, a 24-peaked multi-modal distribution, a cyclic distribution, and a 3-line distribution. The 24-peaked distribution and the cyclic distribution are designed to simulate the symmetry property of *cube* and *cone* solids. We adopt the visualization tool of [25] and show the target distributions as follows.

Baselines [9] introduces the reparameterization trick for Lie groups and allows for constructing flows on the Lie algebra of $\mathrm{SO}(3)$. [23] proposes continuous normalizing flows on Riemannian manifold, and we apply it to $\mathrm{SO}(3)$ manifold. [25] models the distribution implicit by the neural networks, where the $\mathrm{SO}(3)$ space is uniformly discretized. Finally, we compare the mixture of matrix Fisher distribution with 500 components.

Results The results are reported in Table 1, where our model(Mobius + Affine) consistently achieves state-of-the-art performance among all baselines, demonstrating the ability of our method to fit arbitrary distributions in various shapes. We also report results of flows composed of single transformation, i.e. Mobius coupling layer or quaternion affine transformation. Results demonstrate that Mobius coupling layers can generally perform well while quaternion affine transformations are more suited for uni-modal distributions and infinite-modal distributions. Improvement by composing Mobius coupling layers and quaternion affine transformation is more demonstrated in conditional tasks, as shown in Sec. 5.4.

In our experiment, baseline [9] fails to fit the sharp distribution due to numerical unstable. For detailed discussions, see Supplementary Material.

Table 1. **Comparisons on learning to fit various distributions.** We adopt log-likelihood as the evaluation metric and use uniform distribution in $\mathrm{SO}(3)$ as base distribution.

log likelihood ↑	avg.	peak	cone	cube	line
Riemannian [23]	5.82	13.47	8.82	1.02	-0.026
ReLie [9]	-	-	5.32	3.27	-6.97
IPDF [25]	4.38	7.30	4.75	4.33	1.12
Mixture MF [24]	6.04	10.52	8.36	4.52	0.77
Moser Flow [30]	6.28	11.15	8.22	4.42	**1.38**
Ours(Mobius)	**7.28**	**13.93**	8.98	**4.81**	**1.38**
Ours(Affine)	5.59	13.50	8.84	0.00	0.00
Ours(Mobius+Affine)	**7.28**	**13.93**	**8.99**	**4.81**	**1.38**

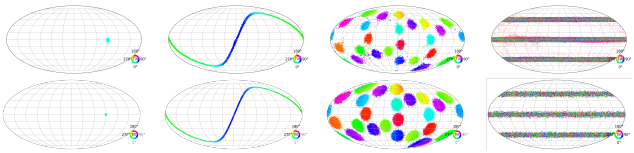

Figure 5. **Visualization of learned distributions for synthetic dataset**. **Top**: our learned distributions. **Bottom**: ground truth of density distributions. From the left to right are respectively peak, cone-like, cube-like, and line distributions. The distribution visualization is the same as Figure 1.

5.2. Rotation Regression with Conditional Normalizing Flows

In this experiment, we leverage our normalizing flows on conditional rotation regression given a single image.

5.2.1 SYMSOL I/II

Datasets We experiment on SYMSOL dataset introduced by [25]. SYMSOL I dataset contains images with solids with high order of symmetry, e.g., tetrahedron, cube, cone, cylinder, which challenges probabilistic approaches to learn complex pose distributions. SYMSOL II dataset includes solids with small markers to break the regular symmetries. We follow the experiment settings of [25].

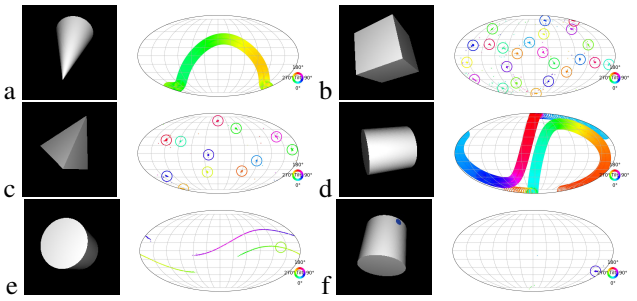

Figure 6. **Visualization of learned distributions for SYMSOL**. The ground truth is shown as a big circle around the point on the sphere which corresponds to the first axis and is colored according to the rotation of the axis. **a**: The cone possesses rotational symmetry and has a circle of equivalent poses. **b,c**: 24-modal distribution of cube and 12-modal distribution of tetrahedron are well fitted by our flow model. **d,e,f**: (d)Cylinder without a mark has 2 circles of equivalent rotations under 2-fold rotation symmetry. (e,f)The cylinder is marked with a dot. When the dot is invisible, orientations with the mark on the hidden side are possible, thus our model predicts 2 half circles of rotations. When dot is visible, our model gives a single and accurate prediction.

Baselines We compare our method to Implicit-PDF [25] as well as several works which parameterize multimodal distributions on $\mathrm{SO}(3)$ for the purpose of pose estimation, including von-Mises distribution [27] and Bingham distribution [6, 13]. We quote numbers of baselines from [25].
Results The log-likelihood scores are reported in Table 2. We can see that on both SYMSOL I and II datasets, our proposed rotation normalizing flows obtain a significant and consistent performance improvement over all the baselines. We further evaluate our method under *spread* metric in Table 3. *Spread*, also referred to as the Mean Absolute Angular Deviation (MAAD) [13, 25, 27], measures the minimum expected angular deviation to the equivalent ground truth $\mathbb{E}_{R\sim p(R|x)}[\min_{R'\in\{R_{GT}\}} d(R,R')]$ and $d(R,R')$ is the geodesic distance between rotations. Results show that our model learns to predict accurate and concentrated rotation pose, with an average MAAD less than 1°.

5.2.2 ModelNet10-SO3

Dataset ModelNet10-SO3 dataset [20] is a popular dataset widely used in the task of regressing rotations from single images. It is synthesized by rendering the CAD models of ModelNet10 dataset [35] that are uniformly rotated.
Baselines We have established our baselines using recent works on probabilistic rotation regression, including von Mises distribution [27], Bingham distribution [6], matrix Fisher distribution [24] and Implicit-PDF [25].
Results We leverage the common acc@15°, acc@30° and median error metrics to evaluate the performance, and the detailed results are shown in Table 4. We show results of using uniform distribution as base distribution (Ours(Uni.)) and a pre-trained fisher distribution as base distribution (Ours(Fisher)). As shown in the table, our method yields superior in the task of unimodal rotation regression. Note that the main advantage of our normalizing flows over other parametric distributions is the superiority to model complex distributions on $\mathrm{SO}(3)$, however, the results of unimodal rotation regression further demonstrate the robustness and wide applications of our method.

5.3. Pascal3D+

Dataset Pascal3D+ dataset [36] is also a popular dataset in rotation regression with a single image as input and it consists of real images with no symmetry.
Baselines Our baselines consist of recent works for probabilistic rotation regression, including von Mises distribution [27], matrix Fisher distribution [24] and Implicit-PDF [25] and non-probabilistic rotation regression methods including [20], [33] and [22].
Results As shown in Table 5, by pretraining a neural network which uses Matrix Fisher distribution [24] to estimate rotation and then using the learned distribution as the base distribution in our flow, our method can achieve the highest acc@30° rate and the lowest median error. This is because the pre-trained conditional Matrix Fisher distribution can serve as a better initialization to our flow, and our flow can further improve the performance as it is able to model more complex distributions and can thus approximate the underlying distribution better. This result shows that our method can collaborate with other probabilistic methods to get better performance.

5.4. Ablation Study

In this experiment, we evaluate the effectiveness of each proposed component in our rotation normalizing flows. To show the 2 functions of affine transformation, we implement Mobius with Rotation only, where W of affine transformation is replaced by $R = UV^T$ of its SVD decomposition $W = USV^T$. The experiment settings are the same as Sec. 5.2 and the results are reported in Table 6.

As shown in Table 6, Mobius coupling layers are of crucial importance while only quaternion affine transformation is not capable to tackle such complex distributions. Quaternion affine transformation performs as an enhancement to Mobius coupling layers and accelerates learning.

For *spread*, as illustrated in Figure 7, it takes Mobius with affine about 100k iterations for convergence, whereas Mobius without affine continued to decrease for 900k iterations. Comparison between Mobius with affine and Mobius with rotation shows the effects of scaling and normalization part to accelerate learning speed as it can quickly concentrate distributions and approximate high mode.

Though *spread* quickly converges, the log-likelihood

Table 2. **Results of rotation regression with conditional normalizing flows on SYMSOL I and II.** We adopt log-likelihood as the evaluation metric and use uniform distribution in $\mathrm{SO}(3)$ as base distribution. Note that we use the convention that a minimally informative uniform distribution has an average log likelihood of 0, which is different from IPDF's convention of -2.29.

	SYMSOL I (log likelihood ↑)						SYMSOL II (log likelihood ↑)			
	avg.	cone	cube	cyl.	ico.	tet.	avg.	sphX	cylO	tetX
Deng et al. [6]	0.81	2.45	-2.15	1.34	-0.16	2.56	4.86	3.41	5.28	5.90
Gilitschenski et al. [13]	1.86	6.13	0.00	3.17	0.00	0.00	5.99	5.61	7.17	5.19
Prokudin et al. [27]	0.42	-1.05	1.79	1.01	-0.10	0.43	2.77	-1.90	6.45	3.77
IPDF [25]	6.39	6.74	7.10	6.55	3.57	7.99	9.86	9.59	9.20	10.78
Ours	**10.38**	**10.05**	**11.64**	**9.54**	**8.26**	**12.43**	**12.94**	**12.37**	**12.92**	**13.53**

Table 3. ***Spread* estimation on SYMSOL.** This metric evaluates how close the probability mass is centered on any of the equivalent ground truths. We follow [25] to evaluate it on SYMSOL I, where all ground truths are known at test time. Values are in degrees.

spread↓	avg.	cone	cube	cyl.	ico.	tet.
Deng et al. [6]	22.4	10.1	40.7	15.2	29.5	16.7
IPDF [25]	4.0	1.4	4.0	1.4	8.4	4.6
Ours	**0.7**	**0.5**	**0.6**	**0.5**	**1.1**	**0.6**

Table 4. **Numerical results of rotation regression on ModelNet10-SO(3).** We adopt 15° and 30° accuaracy and median error as evaluation metric. Metrics are averaged over categories, and the best performance is shown in **bold** while the second best is underlined. See Supplementary Material for the complete table with per-category metrics.

	Acc@15°↑	Acc@30°↑	Med. (°) ↓
Deng et al. [6]	0.562	0.694	32.6
Prokudin et al. [27]	0.456	0.528	49.3
Mohlin et al. [24]	0.693	0.757	17.1
IPDF [25]	0.719	0.735	21.5
Ours (Uni.)	**0.760**	**0.774**	14.6
Ours (Fisher)	0.744	0.768	**12.2**

Table 5. **Numerical results of rotation regression on Pascal3D+ dataset.** We adopt 30° accuaracy and median error as the evaluation metrics. Metrics are averaged over categories, and the best performance is shown in **bold**. See Supplementary Material for the complete table with per-category metrics.

	Acc@30° ↑	Med.(°)↓
Liao et al. [20]	0.819	13.0
Mohlin et al. [24]	0.825	11.5
Prokudin et al. [27]	0.838	12.2
Tulsiani & Malik [33]	0.808	13.6
Mahendran et al. [22]	0.859	10.1
IPDF [25]	0.837	10.3
Ours (Uni.)	0.827	10.2
Ours (Fisher)	**0.863**	**9.9**

continues to increase, as the confidence of predictions is enhanced during training.

6. Conclusion

In this work, we show the capability of our proposed novel discrete normalizing flows for rotations to learn various kinds of distributions on $\mathrm{SO}(3)$. The proposed flow is numerically stable and very expressive, thanks to the complementary role of our proposed Mobius coupling layer and quaternion affine transformation. Our extensive experiments demonstrate that our flows are able to fit complex distributions on $\mathrm{SO}(3)$ and achieve the best performance in both unconditional and conditional tasks.

The Mobius coupling can be elegantly interpreted as a bundle isomorphism. It uses the fiber bundle with projection $\pi : \mathrm{SO}(3) \to S_2$ and fiber $S_1 = SO(2)$. Choosing the conditional dimension $i = 1, 2, 3$ defines one such projection, for each base space point, there is a diffeomorphism over its fiber, which together form a base-space preserving bundle isomorphism. Our Mobius coupling design and affine transformation for accelerating may shed light on normalizing flows for other fiber bundles.

Table 6. **Ablation study on Affine transformation on SYMSOL I dataset**. We adopt *log-likelihood* and *spread* as evaluation matrice.

(log likelihood ↑)	avg.	cone	cube	cyl.	ico.	tet.
Mobius	9.41	10.52	9.68	**10.00**	5.35	11.51
Affine	1.96	9.79	0.00	0.00	0.00	0.00
Mobius + Rotation	10.06	**10.65**	9.91	9.99	7.97	11.78
Mobius + Affine (Ours)	**10.38**	10.05	**11.64**	9.54	**8.26**	**12.43**

(*spread* ↓)	avg.	cone	cube	cyl.	ico.	tet.
Mobius	1.60	0.45	1.00	**0.42**	5.18	0.96
Affine	35.49	0.51	41.16	56.45	29.05	50.29
Mobius + Rotation	0.87	**0.40**	1.63	0.43	**1.06**	0.84
Mobius + Affine(Ours)	**0.67**	0.50	**0.60**	0.53	1.08	**0.64**

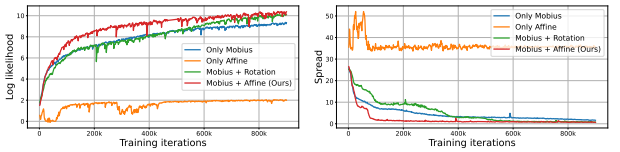

Figure 7. **Ablation study on Affine transformation on SYMSOL I dataset**. We plot *log likelihood* and *spread* evolving with training iterations.

Acknowledgement

This work is supported in part by National Key R&D Program of China 2022ZD0160801.

References

[1] Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, pages 573–582. PMLR, 2019. 1, 2

[2] Ricky TQ Chen, Jens Behrmann, David K Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. *Advances in Neural Information Processing Systems*, 32, 2019. 1, 2

[3] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018. 2

[4] Travers Ching, Daniel S Himmelstein, Brett K Beaulieu-Jones, Alexandr A Kalinin, Brian T Do, Gregory P Way, Enrico Ferrero, Paul-Michael Agapow, Michael Zietz, Michael M Hoffman, et al. Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, 15(141):20170387, 2018. 1

[5] John L Crassidis and F Landis Markley. Unscented filtering for spacecraft attitude estimation. *Journal of guidance, control, and dynamics*, 26(4):536–542, 2003. 1

[6] Haowen Deng, Mai Bui, Nassir Navab, Leonidas Guibas, Slobodan Ilic, and Tolga Birdal. Deep bingham networks: Dealing with uncertainty and ambiguity in pose estimation. *International Journal of Computer Vision*, pages 1–28, 2022. 1, 2, 7, 8

[7] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014. 1, 2, 3

[8] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016. 1, 2, 3

[9] Luca Falorsi, Pim de Haan, Tim R Davidson, and Patrick Forré. Reparameterizing distributions on lie groups. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3244–3253. PMLR, 2019. 1, 2, 6

[10] Luca Falorsi and Patrick Forré. Neural ordinary differential equations on manifolds. *arXiv preprint arXiv:2006.06663*, 2020. 1, 2

[11] Chris Finlay, Jörn-Henrik Jacobsen, Levon Nurbekyan, and Adam M Oberman. How to train your neural ode. *arXiv preprint arXiv:2002.02798*, 2020. 2

[12] Mevlana C Gemici, Danilo Rezende, and Shakir Mohamed. Normalizing flows on riemannian manifolds. *arXiv preprint arXiv:1611.02304*, 2016. 2

[13] Igor Gilitschenski, Roshni Sahoo, Wilko Schwarting, Alexander Amini, Sertac Karaman, and Daniela Rus. Deep orientation uncertainty learning based on a bingham loss. In *International Conference on Learning Representations*, 2019. 1, 2, 7, 8

[14] Jared Glover, Gary Bradski, and Radu Bogdan Rusu. Monte carlo pose estimation with quaternion kernels and the bingham distribution. In *Robotics: science and systems*, volume 7, page 97, 2012. 1

[15] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018. 2

[16] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, pages 2722–2730. PMLR, 2019. 1, 2

[17] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018. 1, 2, 3, 5

[18] Nikos Kolotouros, Georgios Pavlakos, Dinesh Jayaraman, and Kostas Daniilidis. Probabilistic modeling for human mesh recovery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11605–11614, 2021. 1, 2

[19] Christian Leibig, Vaneeda Allken, Murat Seçkin Ayhan, Philipp Berens, and Siegfried Wahl. Leveraging uncertainty information from deep neural networks for disease detection. *Scientific reports*, 7(1):1–14, 2017. 1

[20] Shuai Liao, Efstratios Gavves, and Cees GM Snoek. Spherical regression: Learning viewpoints, surface normals and 3d rotations on n-spheres. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9759–9767, 2019. 7, 8

[21] Aaron Lou, Derek Lim, Isay Katsman, Leo Huang, Qingxuan Jiang, Ser Nam Lim, and Christopher M De Sa. Neural manifold ordinary differential equations. *Advances in Neural Information Processing Systems*, 33:17548–17558, 2020. 1, 2

[22] Siddharth Mahendran, Haider Ali, and Rene Vidal. A mixed classification-regression framework for 3d pose estimation from 2d images. *arXiv preprint arXiv:1805.03225*, 2018. 7, 8

[23] Emile Mathieu and Maximilian Nickel. Riemannian continuous normalizing flows. *Advances in Neural Information Processing Systems*, 33:2503–2515, 2020. 1, 2, 6

[24] David Mohlin, Josephine Sullivan, and Gérald Bianchi. Probabilistic orientation estimation with matrix fisher distributions. *Advances in Neural Information Processing Systems*, 33:4884–4893, 2020. 1, 2, 6, 7, 8

[25] Kieran Murphy, Carlos Esteves, Varun Jampani, Srikumar Ramalingam, and Ameesh Makadia. Implicit-pdf: Non-parametric representation of probability distributions on the rotation manifold. *arXiv preprint arXiv:2106.05965*, 2021. 2, 4, 6, 7, 8

[26] George Papamakarios, Eric T Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *J. Mach. Learn. Res.*, 22(57):1–64, 2021. 2

[27] Sergey Prokudin, Peter Gehler, and Sebastian Nowozin. Deep directional statistics: Pose estimation with uncertainty quantification. In *Proceedings of the European conference on computer vision (ECCV)*, pages 534–551, 2018. 1, 2, 7, 8

[28] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015. 1

[29] Danilo Jimenez Rezende, George Papamakarios, Sébastien Racaniere, Michael Albergo, Gurtej Kanwar, Phiala Shanahan, and Kyle Cranmer. Normalizing flows on tori and spheres. In *International Conference on Machine Learning*, pages 8083–8092. PMLR, 2020. 1, 2, 3, 4

[30] Noam Rozen, Aditya Grover, Maximilian Nickel, and Yaron Lipman. Moser flow: Divergence-based generative modeling on manifolds. *Advances in Neural Information Processing Systems*, 34:17669–17680, 2021. 2, 6

[31] Torsten Sattler, Qunjie Zhou, Marc Pollefeys, and Laura Leal-Taixe. Understanding the limitations of cnn-based absolute camera pose regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3302–3312, 2019. 1

[32] Gesina Schwalbe and Martin Schels. A survey on methods for the safety assurance of machine learning based systems. In *10th European Congress on Embedded Real Time Software and Systems (ERTS 2020)*, 2020. 1

[33] Shubham Tulsiani and Jitendra Malik. Viewpoints and keypoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1510–1519, 2015. 7, 8

[34] Christina Winkler, Daniel Worrall, Emiel Hoogeboom, and Max Welling. Learning likelihoods with conditional normalizing flows. *arXiv preprint arXiv:1912.00042*, 2019. 6

[35] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 7

[36] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE winter conference on applications of computer vision*, pages 75–82. IEEE, 2014. 7

[37] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5738–5746, 2019. 1