AQUAMAM: AN AUTOREGRESSIVE, QUATERNION MANIFOLD MODEL FOR RAPIDLY ESTIMATING COMPLEX $\mathbf{SO}(3)$ DISTRIBUTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Accurately modeling complex, multimodal distributions is necessary for optimal decision-making, but doing so for rotations in three-dimensions, i.e., the $\mathbf{SO}(3)$ group, is challenging due to the curvature of the rotation manifold. The recently described implicit-PDF (IPDF) is a simple, elegant, and effective approach for learning arbitrary distributions on $\mathbf{SO}(3)$ up to a given precision. However, inference with IPDF requires N forward passes through the network's final multi-layer perceptron—where N places an upper bound on the likelihood that can be calculated by the model—which is prohibitively slow for those without the computational resources necessary to parallelize the queries. In this paper, I introduce AQuaMaM,[1][2] a neural network capable of both learning complex distributions on the rotation manifold *and* calculating exact likelihoods for query rotations in a single forward pass. Specifically, AQuaMaM *autoregressively* models the projected components of unit quaternions as a mixture of uniform distributions that partition their geometrically-restricted domain of values. On an "infinite" toy dataset with ambiguous viewpoints, AQuaMaM rapidly converges to a sampling distribution closely matching the true data distribution. In contrast, the sampling distribution for IPDF dramatically diverges from the true data distribution, despite IPDF approaching its theoretical minimum evaluation loss during training. On a constructed dataset of 500,000 renders of a die in different rotations, an AQuaMaM model trained from scratch reaches a log-likelihood 14% higher than an IPDF model using a pretrained ResNet-50. Further, compared to IPDF, AQuaMaM uses 24% fewer parameters, has a prediction throughput 52× faster on a single GPU, and converges in a similar amount of time during training.

1 INTRODUCTION AND RELATED WORK

In many robotics applications, e.g., robotic weed control (Wu et al., 2020), the ability to accurately estimate the poses of objects is a prerequisite for successful deployment. However, compared to other automation tasks, which primarily involve either classification or regression in $\mathbb{R}^n$, pose estimation is particularly challenging because the 3D rotation group $\mathbf{SO}(3)$[3] lies on a curved manifold. As a result, standard probability distributions (e.g., the multivariate Gaussian) are not well-suited for modeling elements of the $\mathbf{SO}(3)$ set. Further, because the steps for interacting with an object in the "mean" pose *between* two possible poses (Figure 1) will often fail when applied to the object when it is in one of the *non-mean* poses, accounting for multimodality in the context of rotations is essential.

The recently described implicit-PDF (IPDF) (Murphy et al., 2021) is a simple,

[1] Pronounced "aqua ma'am".

[2] All code to generate the datasets, train and evaluate the models, and generate the figures can be found at: <anonymized for review>.

[3] $\mathbf{SO}(3)$ stands for "special orthogonal group in three dimensions", with the "special" referring to the fact that all rotation matrices have a determinant of one. See: `https://blogs.scientificamerican.com/roots-of-unity/a-few-of-my-favorite-spaces-so-3/` for a popular science introduction to $\mathbf{SO}(3)$.

elegant, and effective approach for modeling distributions on $\mathbf{SO}(3)$ that both respects the curvature of the rotation manifold and is inherently multimodal. The IPDF model f is trained through negative sampling where, for each ground truth image/rotation pair $(\mathbf{X}, \boldsymbol{R}_1)$, a set of $N-1$ negative rotations $\{\boldsymbol{R}_i\}_2^N$ are sampled and a score is assigned to each rotation matrix as $s_i = f(\boldsymbol{X}, \boldsymbol{R}_i)$. The final density $p(\boldsymbol{R}_1|\mathbf{X})$ is then approximated as $p(\boldsymbol{R}_1|\mathbf{X}) \approx \mathrm{softmax}(\boldsymbol{s})[1]/V$ where $\boldsymbol{s}$ is a vector containing the scores with $\boldsymbol{s}[1] = s_1$, and $V = \pi^2/N$, i.e., the volume of $\mathbf{SO}(3)$ split into N pieces. While effective, the N hyperparameter induces a non-trivial trade-off between model precision (in terms of the maximum likelihood that can be assigned to a rotation) and inference speed in environments that do not have the computational resources necessary to extensively parallelize the task. Further, again due to the precision/speed trade-off, IPDF is trained with $N_{\text{train}} \ll N_{\text{test}}$,[4] which can make it difficult to reason about how the model will behave in the wild.

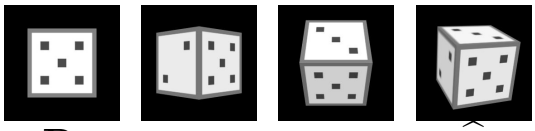


Figure 1: When minimizing the unimodal Bingham loss for the two rotations $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$, the maximum likelihood estimate $\widehat{\boldsymbol{R}}$ is a rotation that was never observed in the dataset. Note, the die images are for demonstration purposes only, i.e., no images were used during optimization. $\boldsymbol{R}_0$ is the identity rotation.

The alternative to *implicitly* modeling distributions on $\mathbf{SO}(3)$ is *explicitly* modeling them. Here, I briefly describe the three baselines used in Murphy et al. (2021), which are representative of explicit distribution modeling approaches. Notably, IPDF outperformed each of these models by a wide margin in a distribution modeling task (see Table 1 in Murphy et al. (2021)). First, Prokudin et al. (2018) described a biternion network (Beyer et al., 2015) trained to output Euler angles by optimizing a loss derived from the von Mises distribution. The two multimodal variants of the model consist of: (1) a variant that outputs mixture components for a von Mises mixture distribution, and (2) an "infinite mixture model" variant, which is implemented as a conditional (Sohn et al., 2015) variational autoencoder (Kingma & Welling, 2014). Second, Gilitschenski et al. (2020) described a procedure for training a network to directly model a distribution of rotations by optimizing a loss derived from the Bingham distribution (Bingham, 1974), with the multimodal variant outputting the mixture components for a Bingham mixture distribution. Lastly, Deng et al. (2020) extended the work of Gilitschenski et al. (2020) by optimizing a "winner takes all" loss (Guzmán-rivera et al., 2012; Rupprecht et al., 2017) in an attempt to overcome the difficulties (Makansi et al., 2019) commonly encountered when training Mixture Density Networks (Bishop, 1994).

One additional approach that needs to be introduced is the direct classification of an individual rotation from a set of rotations, which, in some sense, is the explicit version of IPDF. Unfortunately, the computational complexity of this strategy quickly becomes prohibitive as more precision is required. For example, Murphy et al. (2021) used an evaluation grid of $\sim$2.4 million equivolumetric cells (Yershova et al., 2010), which would not only require an extraordinary number of parameters in the final classification layer of a model, but would also require an extremely large dataset to reasonably "fill in" the grid due to the curse of dimensionality.[5]

In this paper, I introduce **AQuaMaM**, an **A**utoregressive, **Qua**ternion **Ma**nifold **M**odel that learns arbitrary distributions on $\mathbf{SO}(3)$ with a high level of precision. Specifically, AQuaMaM models the projected components of unit quaternions as a mixture of uniform distributions that partition their geometrically-restricted domain of values. Architecturally, AQuaMaM is a Transformer (Vaswani et al., 2017). Although Transformers were originally motivated by language tasks, their flexibility and expressivity have allowed them to be successfully applied to a wide range of data types, including: images (Parmar et al., 2018; Dosovitskiy et al., 2021), tabular data (Padhi et al., 2021; Fakoor et al., 2020; Alcorn & Nguyen, 2021c), multi-agent trajectories (Alcorn & Nguyen, 2021a), and proteins (Rao et al., 2021). Recently, multimodal Transformers (Ramesh et al., 2021; Reed et al., 2022), i.e., Transformers that *jointly* process different modalities of input (e.g., text and images), have revealed additional fascinating capabilities of this class of neural networks. AQuaMaM is multimodal in both senses of the word—it processes inputs from different modalities (images and unit quaternions) to model distributions on the rotation manifold with multiple modes.

[4]In Murphy et al. (2021), $N_{\text{train}} = 4{,}096$ and $N_{\text{test}} = 2{,}359{,}296$, i.e., $N_{\text{train}}/N_{\text{test}} = 0.2\%$.

[5]Notably, Mahendran et al. (2018) used a maximum of 200 rotations in their classification approach.

To summarize my contributions, I find that:

1. AQuaMaM is highly effective at modeling arbitrary distributions on the rotation manifold. On an "infinite" toy dataset with ambiguous viewpoints, AQuaMaM rapidly converges to a sampling distribution closely matching the true data distribution (Section 4.1). In contrast, the sampling distribution for IPDF dramatically diverges from the true data distribution, despite IPDF approaching its theoretical minimum evaluation loss during training. Additionally, on a constructed dataset of 500,000 renders of a die in different rotations, an AQuaMaM model trained from scratch reaches a log-likelihood 14% higher than an IPDF model using a pretrained ResNet-50 (Section 4.2).
2. AQuaMaM is *fast*, reaching a prediction throughput 52× faster than IPDF on a single GPU.

2 THEORY

2.1 THE QUATERNION FORMALISM OF ROTATIONS

Several different formalisms exists for rotations in three dimensions, likely the most widely encountered of which is the rotation matrix—a 3×3 orthonormal matrix that has determinant of one. Another formalism, particularly popular in computer graphics software, is the unit quaternion. Quaternions (typically denoted as the algebra $\mathbb{H}$) are extensions of the complex numbers, where the quaternion $\boldsymbol{q}$ is defined as $\boldsymbol{q} = q_x\boldsymbol{i} + q_y\boldsymbol{j} + q_z\boldsymbol{k} + q_w$ with $q_x, q_y, q_z, q_q \in \mathbb{R}$ and $\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -1$. Remarkably, the axis and angle of every 3D rotation can be encoded as a unit quaternion $\boldsymbol{q} \in \mathbb{H}_1 = \{\boldsymbol{q} \in \mathbb{H} : \|\boldsymbol{q}\| = 1\}$, i.e., $q_w = \cos(\theta/2)$, $q_x = e_x \sin(\theta/2)$, $q_y = e_y \sin(\theta/2)$, and $q_z = e_z \sin(\theta/2)$ where $\boldsymbol{e} = [e_x, e_y, e_z]$ is a unit vector indicating the axis of rotation and the angle θ describes the magnitude of the rotation about the axis. Additionally, the unit quaternions $\boldsymbol{q}$ and $-\boldsymbol{q}$ encode the same rotation, i.e., $\mathbb{H}_1$ double covers $\mathbf{SO}(3)$.

2.2 MODELING SO(3) DISTRIBUTIONS AS MIXTURES OF UNIFORM DISTRIBUTIONS ON PROJECTED UNIT QUATERNIONS

Due to the curvature of the 3-sphere S^3, directly modeling distributions on $\mathbb{H}_1$ is difficult. However, because $\mathbb{H}_1$ double covers $\mathbf{SO}(3)$, we can narrow our focus to the subset of unit quaternions with $q_w \geq 0$, i.e., $\widetilde{\mathbb{H}}_1 = \{\boldsymbol{q} \in \mathbb{H}_1 : q_w \geq 0\}$. Practically speaking, we can convert any $\boldsymbol{q}$ in a dataset with $q_w < 0$ to $-\boldsymbol{q}$. Note that q_w is now fully determined because $\forall \boldsymbol{q} \in \widetilde{\mathbb{H}}_1, q_w = \sqrt{1 - q_x^2 - q_y^2 - q_z^2}$. Therefore, there is a one-to-one mapping $f : B^3 \to \widetilde{\mathbb{H}}_1$ (where $B^3 = \{\boldsymbol{v} \in \mathbb{R}^3 : \|\boldsymbol{v}\| \leq 1\}$ is the unit 3-ball), which allows us to estimate $p(q_x, q_y, q_z)$ on the non-curved B^3 using standard distributions.

Using the chain rule of probability, we can factor the joint probability $p(q_x, q_y, q_z)$ as $p(q_x, q_y, q_z) = p(q_x)p(q_y|q_x)p(q_z|q_y, q_x)$. To model this distribution, desiderata for each factor include: (1) the flexibility to handle complex multimodal distributions and (2) the ability to satisfy the unit norm constraint of $\boldsymbol{q} \in \widetilde{\mathbb{H}}_1$. By partitioning $[-1, 1]$ into N bins, each with a width of $\frac{2}{N}$, we can model the distributions for q_x, q_y, and q_z as mixtures of uniform distributions where N controls the maximum precision of the model. Recall that for the uniform distribution $\mathcal{U}_{[a,b)}$ with $-\infty < a < b < \infty$ the density is $\mathcal{U}(x; a, b) = \begin{cases} \frac{1}{b-a} & \text{for } x \in [a, b) \\ 0 & \text{otherwise} \end{cases}$. Therefore, the density for q_x is:

$$p(q_x) = \sum_{i=1}^{N} \pi_i \mathcal{U}(q_x; a_i, b_i) = \pi_k \mathcal{U}(q_x; a_k, b_k) = \pi_k \frac{1}{b_k - a_k} = \pi_k \frac{1}{2/N} = \pi_k \frac{N}{2} \tag{1}$$

where π_i is the mixture proportion for the mixture component indexed by i, and k is the index for the bin that contains q_x.

For q_y and q_z, the conditional densities are similar to the density for q_x, but the unit norm constraint on $\boldsymbol{q}$ presents opportunities for introducing inductive biases into a model (Figure 2). For example, consider the bins that are fully contained within $\mathbb{R}_{\geq 0}$, i.e., where $a, b \geq 0$. For any such bin where

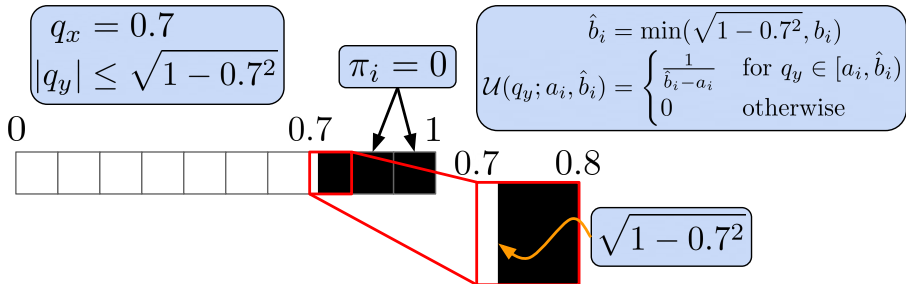


Figure 2: When modeling the conditional distribution $p(q_y|q_x)$ as a mixture of uniform distributions, the geometric constraints of the unit quaternion are easily enforced. Here, I focus on non-negative bins for clarity, i.e., intervals $[a_i, b_i)$ where $0 \leq a < b \leq 1$, but the same logic applies to negative bins. Given $q_x = 0.7$, we know that $|q_y| \leq \sqrt{1-0.7^2}$ because $\boldsymbol{q}$ has a unit norm. As a result, the mixture proportion π_i for any bin where $\sqrt{1-0.7^2} < a_i$ *must* be zero. AQuaMaM enforces this constraint by assigning a value of $-\infty$ to the output scores for "strictly illegal bins" during training.[6]For the remaining bins, the corresponding uniform distribution is $\mathcal{U}(q_y; a_i, \hat{b}_i)$ where $\hat{b}_i = \min(\sqrt{1-0.7^2}, b_i)$, i.e., the upper bound of the uniform distribution for the partially legal bin is reduced to $\sqrt{1-0.7^2}$.

$\sqrt{1-q_x^2} < a_i$, we *know* that $\pi_i = 0$ for q_y because $\|\boldsymbol{q}\| = 1$. Further, again due to the unit norm constraint, the conditional density for q_y is in fact:

$$p(q_y|q_x) = \pi_k \frac{1}{\min(\sqrt{1-q_x^2}, b_k) - a_k} \tag{2}$$

i.e., when $a_k < \sqrt{1-q_x^2} < b_k$, the upper bound of the bin's uniform distribution is reduced. The full density for $p(q_x, q_y, q_z)$ is thus:

$$p(q_x, q_y, q_z) = \pi_{q_x} \frac{N}{2} \pi_{q_y} \frac{1}{\omega_{q_y}} \pi_{q_z} \frac{1}{\omega_{q_z}} = \pi_{q_x} \pi_{q_y} \pi_{q_z} \frac{N}{2\omega_{q_y}\omega_{q_z}} \tag{3}$$

where π_{q_c} is the mixture proportion for the bin containing component q_c and ω_c is the potentially reduced width of the bin for q_c used in Equation 2.

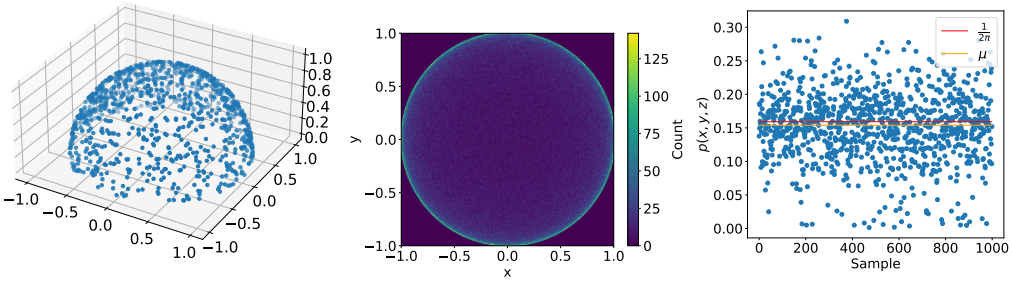
2.3 Calculating $p(\mathbf{q})$

While $p(q_x, q_y, q_z)$ can be used to generate samples on the rotation manifold, the density $p(q_x, q_y, q_z) \neq p(\boldsymbol{q})$ because the curvature of $\widetilde{\mathbb{H}}_1$ "expands" regions of B^3; therefore, $p(q_x, q_y, q_z)$ cannot be used for comparing likelihoods with models like IPDF. Intuitively, thinking about the probability *density* of a point as its infinitesimal probability *mass* divided by its infinitesimal *volume*, the density $p(q_x, q_y, q_z)$ needs to be "diluted" by however much its volume expands when transported to $\widetilde{\mathbb{H}}_1$.[7] Specifically, the diluting factor is $\frac{1}{s_{\boldsymbol{q}}}$ where $s_{\boldsymbol{q}}$ is the magnitude of the wedge product of the columns of the Jacobian matrix $\boldsymbol{J}$ for $f(q_x, q_y, q_z) = [q_x, q_y, q_z, q_w]$ where $q_w = \sqrt{1-q_x^2-q_y^2-q_z^2}$. Similar to how the magnitude of the cross product of two vectors equals the area of the parallelogram defined by the vectors, the magnitude of the wedge product for three vectors is the volume of the parallelepiped defined by the vectors.[8] Figure 3 visualizes an analogous density transformation in two dimensions.

[6]See Section A.1 for a subtlety regarding "strictly illegal bins".

[7]See the lecture notes here: `https://www.cs.cornell.edu/courses/cs6630/2015fa/notes/pdf-transform.pdf` for a survey of probability density transformations.

[8]See the lecture notes here: `https://sheelganatra.com/spring2013_math113/notes/cross_products.pdf` for additional connections between the cross product and the wedge product.



(a) (b) (c)

Figure 3: Because there is a one-to-one mapping between the unit disk B^2 and the unit hemisphere $\widetilde{S}^2$, the challenging task of estimating a distribution on the curved $\widetilde{S}^2$ manifold can be simplified to estimating a distribution on the non-curved B^2. (a) Here, the true distribution on $\widetilde{S}^2$ is a uniform distribution, i.e., each point has a density of $\frac{1}{2\pi}$ (because $\widetilde{S}^2$ has a surface area of 2π). (b) The (x, y) coordinates of points uniformly sampled from $\widetilde{S}^2$ are more concentrated towards the edges of B^2 due to the curvature of $\widetilde{S}^2$. If we model the distribution of (x, y) coordinates on B^2 as a mixture of uniform distributions, we can calculate $p(x, y, z)$ by dividing $p(x, y)$ by the area of the parallelogram at (x, y, z) on the hemisphere defined by the Jacobian. (c) The $p(x, y, z)$ calculated through this procedure are generally quite close to the expected density. The mean density μ of the 1,000 points shown in (c) is 0.154 (compared to 0.159 for the true density). A similar procedure is used by AQuaMaM to obtain the probability of a unit quaternion $p(\boldsymbol{q})$ while only modeling the first three components of $\boldsymbol{q}$: q_x, q_y, and q_z. See Section A.2 for additional details.

2.4 Optimizing $p(\mathbf{q})$ as a "Quaternion Language Model"

After incorporating the diluting factor, the final density for $\boldsymbol{q} \in \widetilde{\mathbb{H}}_1$ is:

$$p(\boldsymbol{q}) = \pi_{q_x} \pi_{q_y} \pi_{q_z} \frac{N q_w}{2 \omega_{q_y} \omega_{q_z}} \tag{4}$$

As previously demonstrated by Alcorn & Nguyen (2021b), models using mixtures of uniform distributions that partition some domain can be optimized solely using a "language model loss". Here, the negative log-likelihood (NLL) for a dataset $\mathcal{X}$ is:

$$\mathcal{L} = -\sum_{d=1}^{|\mathcal{X}|} \ln \pi_{q_{d,x}} + \ln \pi_{q_{d,y}} + \ln \pi_{q_{d,z}} + \ln \frac{N q_{d,w}}{2 \omega_{q_{d,y}} \omega_{q_{d,z}}} \tag{5}$$

Notice that the last term is constant for a given dataset, i.e., it can be ignored during optimization. The final loss is then $\widehat{\mathcal{L}} = -\sum_{d=1}^{|\mathcal{X}|} \ln \pi_{q_{d,x}} + \ln \pi_{q_{d,y}} + \ln \pi_{q_{d,z}}$, which is exactly the loss of a three token autoregressive language model.

As a final point, it is worth noting that the last term in Equation 4 is bounded below such that $\frac{N q_w}{2 \omega_{q_y} \omega_{q_z}} \geq \frac{N^3 q_w}{8}$, i.e., for a given $\pi_{q_x} \pi_{q_y} \pi_{q_z}$, the likelihood increases *at least* cubically with the number of bins. For 50,257 bins (i.e., the size of GPT-2/3's vocabulary (Radford et al., 2019; Brown et al., 2020)), $N^3 = 1.26 \times 10^{14}$. In contrast, the likelihood for IPDF only scales linearly with the number of equivolumetric grid cells (the equivalent user-defined precision hyperparameter).

3 AQuaMaM

3.1 Architecture

Here, I describe AQuaMaM (Figure 4), a Transformer trained to perform pose estimation using the autoregressive quaternion framework. Specifically, AQuaMaM consists of a Vision Transformer

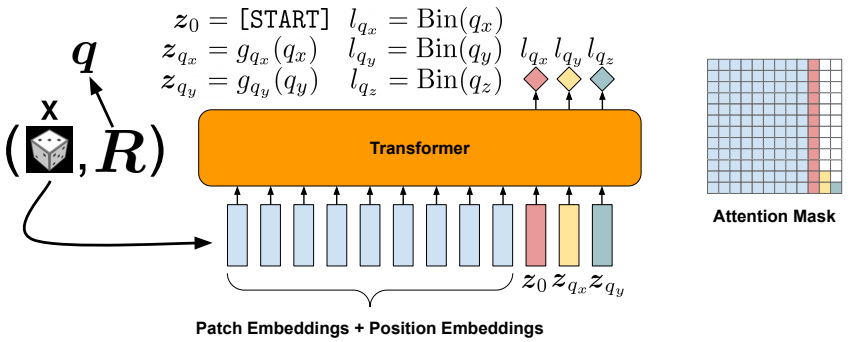


Figure 4: An overview of the AQuaMaM architecture. Given an image/rotation matrix pair $(\mathbf{X}, \boldsymbol{R})$, the image is first converted into a sequence of P patch embeddings while the rotation matrix is converted into its unit quaternion representation $\boldsymbol{q} = [q_x, q_y, q_z, q_w]$. By restricting the unit quaternions to those with positive real components (which is allowed because $\boldsymbol{q}$ and $-\boldsymbol{q}$ encode the same rotation), q_w becomes fully determined and does not need to be modeled. Next, each of the first two components q_c of $\boldsymbol{q}$ is mapped to an embedding z_{q_c} by a separate subnetwork g_{q_c}. The full input to the Transformer is thus a sequence consisting of the P patch embeddings, a special `[START]` embedding z_0, and the two unit quaternion embeddings. The labels l_{q_x}, l_{q_y}, and l_{q_z} are generated by assigning q_x, q_y, and q_z to one of N labels through a binning function Bin. Using a partially causal attention mask, AQuaMaM models the conditional distribution $p(q_x, q_y, q_z|\mathbf{X})$ *autoregressively*, i.e., $p(q_x, q_y, q_z|\mathbf{X}) = p(q_x|\mathbf{X})p(q_y|q_x, \mathbf{X})p(q_z|q_x, q_y, \mathbf{X})$ where each component is modeled as a mixture of uniform distributions that partition the component's geometrically constrained domain. Because minimizing the loss of a mixture of uniform distributions is equivalent (up to a constant) to minimizing the classification loss over the bins, AQuaMaM is trained as a "quaternion language model".

(ViT) (Dosovitskiy et al., 2021) with a few modifications. As with ViT, AQuaMaM tokenizes an input image $\mathbf{X}$ by splitting it into P patches that are then passed through a linear layer to obtain P d_{model}-dimensional embeddings. Three additional embeddings are then appended to this sequence: (1) a `[START]` embedding z_0 (similar to the `[START]` embedding used in language models) and (2) two embeddings z_{q_x} and z_{q_x} corresponding to the first two components of $\boldsymbol{q}$. Each z_{q_c} is obtained by passing q_c through a subnetwork g_{q_c} consisting of a NeRF-like positional encoding function (Vaswani et al., 2017; Mildenhall et al., 2020; Tancik et al., 2020) followed by a multilayer perceptron (MLP), i.e., the size of the input for the MLPs is $1 + 2 \times L$ where L is the number of positional encoding functions.

Once positional embeddings have been added to the input sequence, the full sequence is then fed into a Transformer along with a partially causal attention mask (Alcorn & Nguyen, 2021a;b), which allows AQuaMaM to autoregressively model the conditional distribution $p(q_x, q_y, q_z|\mathbf{X})$ using the density from Equation 4. The transformed z_0, z_{q_x}, and z_{q_y} tokens—$\hat{z}_0$, $\hat{z}_{q_x}$, and $\hat{z}_{q_y}$—are used to assign a probability to the labels for q_x, q_y, and q_z, e.g., $\pi_{q_x} = h(\hat{z}_0)[l_{q_x}]$ where h is the classification head and $l_{q_x} = \text{Bin}(q_x)$ where Bin is the binning function. Notably, AQuaMaM somewhat sidesteps the curse of dimensionality by effectively learning three different models with shared parameters: $p(q_x|\mathbf{X})$, $p(q_y|q_x, \mathbf{X})$, and $p(q_z|q_y, q_x, \mathbf{X})$, each of which only needs to classify N bins, as opposed to learning a single model that must classify on the order of N^3 bins.

3.2 Generating a single prediction

While AQuaMaM *could* generate a single prediction by optimizing $\boldsymbol{q}$ under the full density (Equation 4)—similar to what was done in Murphy et al. (2021)—predictions obtained via optimization are slow. An alternative strategy is to embrace AQuaMaM as a "quaternion language model" and use a greedy search to return a "quaternion sentence" as a prediction. While sentences corresponding to regions where $q_w \approx 0$ will contain a wider range of compatible rotations, with a sufficiently large N, the range of rotations can be kept reasonably small. For example, when $N = 500$ (as in the AQuaMaM die experiment; see Section 4.2), if $q_x \in [0.996, 1]$, then $q_w \in [0, \sqrt{1 - 0.996^2}]$, and the geodesic distance between the rotations corresponding to the quaternions $\boldsymbol{q}_1 = [0.996, 0, 0, \sqrt{1 - 0.996^2}]$ and $\boldsymbol{q}_2 = [1, 0, 0, 0]$ is 10.3°. Using the analogous numbers

for $N = 50{,}257$ (as in the AQuaMaM toy experiment; see Section 4.1), the geodesic distance between $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ is 1.0°.

Generating a quaternion sentence with AQuaMaM requires three passes through the network. However, the last two passes can be considerably optimized by noting that the first $P + 1$ tokens do not depend on the last two tokens due to the partially causal attention mask. A naive decoding strategy where a full (padded) sequence is passed through AQuaMaM three times has an attention complexity of $O(3(P + 3)^2)$. By only passing the first $P + 1$ tokens through AQuaMaM and caching the attention outputs, the complexity of the attention operations when decoding reduces to $O((P + 1)^2 + (P + 2) + (P + 3))$. Empirically, I found that using this caching strategy increased the prediction throughput by approximately twofold.

4 Experiments and Results

Compared to the standard datasets used for computer vision tasks like image classification (Deng et al., 2009) and object detection (Lin et al., 2014)—which contain millions of labeled samples—many commonly used pose estimation datasets are relatively small, with PASCAL3D+ (Xiang et al., 2014) consisting of ∼20,000 training images[9] and T-LESS (Hodan et al., 2017) consisting of ∼38,000 training images. The SYMSOL I and II datasets created by Murphy et al. (2021) consisted of 100,000 renders for each of eight different objects, but the publicly released version only contains 50,000 renders per object.[10] In addition to their relatively small sizes, these pose datasets have the additional complication of including multiple categories of objects. The "canonical" pose for a category of objects is arbitrary, so it is not entirely clear whether combining categories of objects makes pose estimation harder or easier for a learning algorithm (a jointly trained model essentially needs to do *both* classification and pose estimation).[11] Therefore, to investigate the properties of AQuaMaM in moderately large data regimes where Transformers excel, I trained AQuaMaM on two constructed datasets for pose estimation, an "infinite" toy dataset (Section 4.1) and a dataset of 500,000 renders of a die (Section 4.2). Because Murphy et al. (2021) found that IPDF outperformed other multimodal approaches by a wide margin in their distribution estimation task, I only compare to an IPDF baseline here.

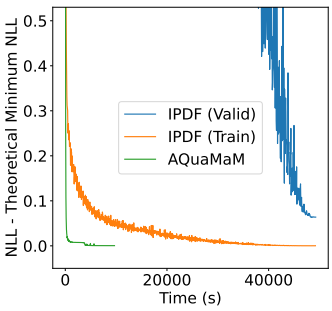


Figure 5: On the infinite toy dataset, AQuaMaM rapidly reached its theoretical minimum (classification) average negative log-likelihood (NLL). In contrast, IPDF never reached its theoretical minimum validation NLL, despite converging to its training theoretical minimum.

4.1 Toy experiment

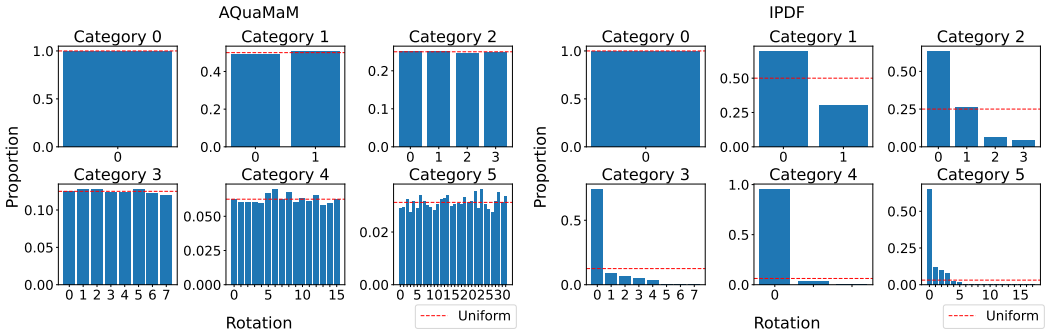
The infinite toy dataset consisted of six categories of viewpoints indexed $i = 0, 1, \ldots, 5$. I associated each viewpoint i with 2^i randomly sampled rotation matrices $\mathcal{R}_i = \{\boldsymbol{R}_j\}_1^{2^i}$, i.e., each viewpoint had a different number of rotation modes reflecting its level of ambiguity. I defined the true data-generating process as a hierarchical model such that a sample was generated by first randomly selecting a viewpoint from a uniform distribution over i, and then randomly selecting a rotation from a uniform distribution over $\mathcal{R}_i$. Given this hierarchical model, calculating the average log-likelihood (LL) in the limit of infinite evaluation test samples simply involves replicating each $(i, \boldsymbol{R}_j)$ pair 2^{5-i} times and then averaging the LLs for each sample. Importantly, because the infinite dataset has identical training and test distributions, it is impossible for a model to overfit the dataset.

[9] Using the standard filtering pipeline where occluded and truncated images are excluded.

[10] The publicly released dataset can be found at: `https://www.tensorflow.org/datasets/catalog/symmetric_solids`.

[11] Further confusing the issue is the fact that many authors augment the PASCAL3D+ dataset with synthetic data, which makes it difficult to separate the influence of the architecture vs. the data augmentations when evaluating model performance.

Using this toy dataset, I trained an AQuaMaM model with $N = 50{,}257$.[12] Because I was only interested in the distribution learning aspects of the model for the toy dataset, the category for each training sample was mapped to a Pd_{model}-dimensional embedding that was converted to a sequence of P d_{model}-dimensional patch embeddings. I also trained an IPDF baseline using the same hyperparameters and learning rate schedule from Murphy et al. (2021) for the SYMSOL I dataset except that I increased the number of iterations from 100,000 to 300,000 after observing that the model was still improving at the end of 100,000 updates. Like the AQuaMaM model, I replaced the image processing component of IPDF, a ResNet-50 (He et al., 2015), with $d_{\text{ResNet-50}}$-dimensional embeddings where $d_{\text{ResNet-50}} = 2{,}048$. In total, the AQuaMaM model had 3,510,609 parameters and the IPDF model had 748,545 parameters; however, 93% of AQuaMaM's parameters were contained in the final classification layer, with only 243,904 being found in the rest of the model.



(a) (b)

Figure 6: (a) The proportions of sampled rotations from the AQuaMaM model trained on the infinite toy dataset closely approximate the expected uniform distributions. (b) In contrast, despite approaching its theoretical minimum log-likelihood during training (Figure 5), the proportions of sampled rotations from the IPDF model drastically diverge from the expected uniform distributions.

The final average LL for AQuaMaM on the toy dataset was 27.12. In comparison, IPDF reached an average LL of 12.32 using the 2.4 million equivolumetric grid from Murphy et al. (2021). Note that the *theoretical maximum* LL for IPDF when using a 2.4 million grid is 12.38, which AQuaMaM comfortably surpassed. In fact, IPDF would need to use a grid of nearly six *trillion* cells for it to even be *theoretically* possible for it to match the LL of AQuaMaM.

Figure 6 shows the results of generating 40,000 samples from the hierarchical model while using AQuaMaM and IPDF to specify the rotation distributions for each sampled viewpoint.[13] For each sampled rotation, the rotation was assigned to the ground truth rotation mode from the sampled viewpoint with the smallest geodesic distance. For AQuaMaM, the average distance was 0.04°, while the average distance for IPDF was 0.84°. Further, the sampled rotations for AQuaMaM closely matched the true uniform distributions for each viewpoint. In contrast, IPDF's sampled distribution drastically diverged from the true data distribution despite its seemingly strong evaluation LL relative to its theoretical maximum. Notably, of the 40,000 quaternion sentences generated by AQuaMaM, only 23 (0.06%) were not from the true distribution.

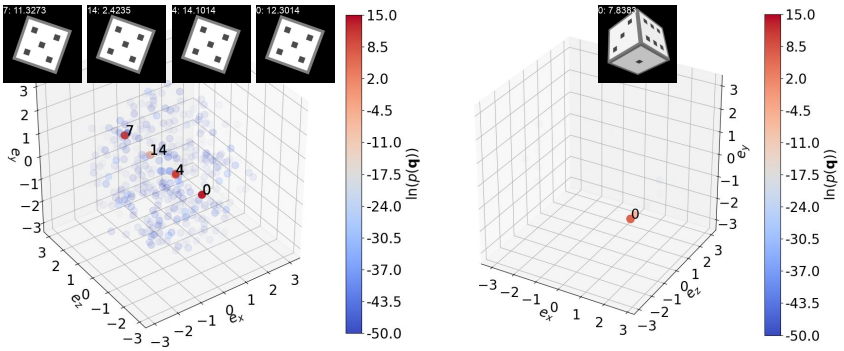
4.2 Die Experiment

The die dataset consisted of 520,000 renders[14] of a die (the same seen in Figure 1) where the die was randomly rotated to generate each image, and the dataset was split into training/validation/test set sizes of 500,000/10,000/10,000. Similar to the SYMSOL II dataset from Murphy et al. (2021), different viewpoints of the die have differing levels of ambiguity, which makes the die dataset appropriate for exploring how AQuaMaM handles uncertainty. Note that, while 500,000 samples is a moderately large training set, for $N = 500$, there are over 65 million unique token sequences

[12] Full training details for both experiments can be found in Section A.4.

[13] See Section A.3 for additional details.

[14] The die renders were generated using the renderer from Alcorn et al. (2019) and the 3D die model from: `https://github.com/garykac/3d-cubes`.



(a) (b)

Figure 7: AQuaMaM effectively models the uncertainty of different viewpoints for the die object. In the plots, each point corresponds to a rotation vector $\boldsymbol{\theta} = \theta \boldsymbol{e}$, i.e., all rotation vectors are contained within a ball of radius π. Both the color and the transparency of each point are a function of the log density $\ln(p(\boldsymbol{q}))$ with redder, more opaque points representing more likely rotations and bluer, more transparent points representing less likely rotations. 500 rotations are depicted in each plot—one rotation (labeled 0) corresponds to the ground truth, 32 rotations were obtained via a beam search with AQuaMaM, and the remaining 467 rotations were randomly selected from the uniform distribution on $\mathbf{SO}(3)$. Each image of a die corresponds to one of the points in its respective plot. The number before the colon indicates the associated annotated point, and the number after the colon is the $\ln(p(\boldsymbol{q}))$ for that pose. (a) Because the five side of the die is facing the camera in the die's default pose, all rotations showing only the five occur along the $(0, 0, 1)$ axis, which is reflected by the locations of the red points in the plot. Additionally, there are four high probability regions along the axis corresponding to the four rotations that could produce an identical image. (b) For this unambiguous viewpoint, almost all of the probability is concentrated at the correct rotation, which causes the other points to fade entirely from view (the *maximum* $\ln(p(\boldsymbol{q}))$ among the random rotations was -82.7). See Section A.6 for a plot corresponding to an object with continuous symmetry.

covering $\mathbf{SO}(3)$,[15] and only 135 of the 10,000 token sequences in the test set were also found in the training set, i.e., AQuaMaM must still be able to generalize. I trained a $\sim$20 million parameter AQuaMaM with $N = 500$ on the die dataset in addition to a $\sim$26 million parameter IPDF baseline that was identical to the model trained on the SYMSOL I dataset in Murphy et al. (2021).

The average LL for AQuaMaM on the die dataset was 14.01 compared to 12.29 for IPDF, i.e., IPDF would need to use a grid of over 12 million cells for it to even be theoretically possible for it to match the average LL of AQuaMaM. When generating single predictions, AQuaMaM had a mean prediction error of 4.32° compared to 4.57° for IPDF, i.e., a 5.5% improvement.[16] In addition to reaching a higher average LL and a lower average prediction error, both the evaluation and prediction throughput for AQuaMaM were over 52$\times$ faster than IPDF on a single GPU. Lastly, as can be seen in Figure 7, AQuaMaM effectively models the uncertainty of viewpoints with differing levels of ambiguity. Figure 7 uses a novel visualization method where low probability rotations are more transparent, which allows all of the dimensions describing the rotation to be displayed, in contrast to the visualization method introduced by Murphy et al. (2021).

5 Conclusion

In this paper, I have shown that a Transformer trained using an autoregressive "quaternion language model" framework is highly effective for learning complex distributions on the $\mathbf{SO}(3)$ manifold. Compared to IPDF, AQuaMaM makes more accurate predictions at a much faster throughput and produces far more accurate sampling distributions. The autoregressive factoring of the quaternion could be applied to other multidimensional datasets where high levels of precision are desirable, e.g., modeling the trajectory of a basketball as in Alcorn & Nguyen (2021a), which was partial inspiration for this work.

[15] $500^3 \times (\frac{4\pi}{3} \div 2^3) \approx$ 65,000,000, i.e., the fraction of grid cells in a cube with a side of two that fall inside the unit sphere.

[16] See Section A.5 for additional details.

REFERENCES

Michael A Alcorn and Anh Nguyen. baller2vec: A multi-entity transformer for multi-agent spatiotemporal modeling. *arXiv preprint arXiv:2102.03291*, 2021a.

Michael A Alcorn and Anh Nguyen. baller2vec++: A look-ahead multi-entity transformer for modeling coordinated agents. *arXiv preprint arXiv:2104.11980*, 2021b.

Michael A. Alcorn and Anh Nguyen. The deformer: An order-agnostic distribution estimating transformer. *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models (INNF+)*, 2021c.

Michael A Alcorn, Qi Li, Zhitao Gong, Chengfei Wang, Long Mai, Wei-Shinn Ku, and Anh Nguyen. Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4845–4854, 2019.

Lucas Beyer, Alexander Hermans, and Bastian Leibe. Biternion nets: Continuous head pose regression from discrete training labels. In *German Conference on Pattern Recognition*, pp. 157–168. Springer, 2015.

Christopher Bingham. An antipodally symmetric distribution on the sphere. *The Annals of Statistics*, pp. 1201–1225, 1974.

Christopher M Bishop. Mixture density networks. 1994.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Haowen Deng, Mai Bui, Nassir Navab, Leonidas Guibas, Slobodan Ilic, and Tolga Birdal. Deep bingham networks: Dealing with uncertainty and ambiguity in pose estimation, 2020.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Rasool Fakoor, Pratik Chaudhari, Jonas Mueller, and Alexander J Smola. Trade: Transformers for density estimation. *Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models Workshop*, 2020.

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*, 2020. URL https://openreview.net/forum?id=ryloogSKDS.

Abner Guzmán-rivera, Dhruv Batra, and Pushmeet Kohli. Multiple choice learning: Learning to produce multiple structured outputs. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/cfbce4c1d7c425baf21d6b6f2babe6be-Paper.pdf.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Tomáš Hodan, Pavel Haluza, Štepán Obdržálek, Jiri Matas, Manolis Lourakis, and Xenophon Zabulis. T-less: An rgb-d dataset for 6d pose estimation of texture-less objects. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 880–888. IEEE, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (eds.), *Computer Vision – ECCV 2014*, pp. 740–755, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1.

Siddharth Mahendran, Haider Ali, and René Vidal. A mixed classification-regression framework for 3d pose estimation from 2d images. In *British Machine Vision Conference 2018, BMVC 2018, Newcastle, UK, September 3-6, 2018*, pp. 72. BMVA Press, 2018. URL `http://bmvc2018.org/contents/papers/0238.pdf`.

Osama Makansi, Eddy Ilg, Ozgun Cicek, and Thomas Brox. Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7144–7153, 2019.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

Kieran A Murphy, Carlos Esteves, Varun Jampani, Srikumar Ramalingam, and Ameesh Makadia. Implicit-pdf: Non-parametric representation of probability distributions on the rotation manifold. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 7882–7893, 2021.

Inkit Padhi, Yair Schiff, Igor Melnyk, Mattia Rigotti, Youssef Mroueh, Pierre Dognin, Jerret Ross, Ravi Nair, and Erik Altman. Tabular transformers for modeling multivariate time series. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3565–3569. IEEE, 2021. URL `https://ieeexplore.ieee.org/document/9414142`.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4055–4064. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/parmar18a.html`.

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)*, pp. 534–551, 2018.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8821–8831. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/ramesh21a.html`.

Roshan M Rao, Jason Liu, Robert Verkuil, Joshua Meier, John Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. Msa transformer. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8844–8856. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/rao21a.html.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

Christian Rupprecht, Iro Laina, Robert DiPietro, Maximilian Baust, Federico Tombari, Nassir Navab, and Gregory D Hager. Learning in an uncertain world: Representing ambiguity through multiple hypotheses. In *Proceedings of the IEEE international conference on computer vision*, pp. 3591–3600, 2017.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Xiaolong Wu, Stéphanie Aravecchia, Philipp Lottes, Cyrill Stachniss, and Cédric Pradalier. Robotic weed control using automated weed and crop classification. *Journal of Field Robotics*, 37(2): 322–340, 2020.

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*, pp. 75–82. IEEE, 2014.

Anna Yershova, Swati Jain, Steven M Lavalle, and Julie C Mitchell. Generating uniform incremental grids on so (3) using the hopf fibration. *The International journal of robotics research*, 29(7):801–812, 2010.

A Appendix

A.1 Defining "strictly illegal bins"

The definition of "strictly illegal bins" used by AQuaMaM is best motivated by an example. Consider a model that was trained on a dataset consisting of a single rotation. Python pseudocode for a naive sampling algorithm can be found in Listing 1 where the `get_probs_naive` function assigns a probability of zero to any bins where all of the values would violate the unit norm constraint given the components of $\boldsymbol{q}$ generated up to that point. This naive sampling procedure will generate sequences of bins that were not found in the training dataset. For example, assume the $\boldsymbol{q}$ for the single rotation had $q_x = 0.42$ and $q_y = 0.901$ and that $N = 20$ for the model. Given a globally optimal model, $\hat{q}_x$ will be sampled from $[0.4, 0.5]$; however, if $\hat{q}_x > \sqrt{1 - 0.9^2} \approx 0.436$, then `get_probs_naive` will assign zero probability to the bin $[0.9, 1.0]$.

To avoid this issue, AQuaMaM defines "strictly illegal bins" as those that would violate the unit norm constraint given the components of $\boldsymbol{q}$ generated up to that point *when using the minimum magnitudes of their corresponding bins*. Effectively, during training AQuaMaM must *learn* to assign zero probability to bins bordering "edge bins" rather than being *told* that they have zero probability (which is the case for strictly illegal bins). Python pseudocode for the final sampling algorithm can

```
def naive_sample_q(AQuaMaM):
    q_cs = []
    for i in range(3):
        bin_probs = AQuaMaM.get_probs_naive(q_cs)
        l_q_c = categorical(bin_probs)
        (lower, upper) = AQuaMaM.bin_edges[l_q_c]
        q_cs.append(constrained_uniform(lower, upper, q_cs))

    (q_x, q_y, q_z) = q_cs
    q_w = sqrt(1 - q_x**2 - q_y**2 - q_z**2)
    return (q_x, q_y, q_z, q_w)
```

Listing 1: Python pseudocode for naively generating a unit quaternion sample with AQuaMaM.

be found in Listing 2. In summary, the sampling procedure involves first generating a sequence of bins, and then sampling from the cell defined by the edges of those bins using rejection sampling. When viewed as a generative model, AQuaMaM can be interpreted as modeling noisy measurements of unit quaternions.

```
def sample_q(AQuaMaM):
    # The q_c_hats are used for generating the bins, but are not the
    final q_cs.
    q_c_hats = []
    cell_edges = []
    for i in range(3):
        bin_probs = AQuaMaM.get_probs(q_c_hats)
        l_q_c = categorical(bin_probs)
        (lower, upper) = AQuaMaM.bin_edges[l_q_c]
        q_c_hats.append(uniform(lower, upper))
        cell_edges.append((lower, upper))

    (q_x, q_y, q_z) = rejection_sample(cell_edges)
    q_w = sqrt(1 - q_x**2 - q_y**2 - q_z**2)
    return (q_x, q_y, q_z, q_w)
```

Listing 2: Python pseudocode for generating a unit quaternion sample with AQuaMaM.

A.2 Transforming a density on the unit disk to a density on the unit hemisphere

Here, I briefly walk through the diluting procedure for a 2D example that is directly analogous to $\widetilde{\mathbb{H}}_1$ and B^3. Consider the task of approximating a uniform distribution on the unit hemisphere $\widetilde{S}^2 = \{\boldsymbol{v} \in \mathbb{R}^3 : \|\boldsymbol{v}\| = 1 \text{ and } v_z \geq 0\}$ by modeling the projected coordinates of $\boldsymbol{v} \in \widetilde{S}^2$ as a mixture of uniform distributions on the unit disk B^2 (Figure 3). Note that $f(x, y) = [x, y, z]$ and $\boldsymbol{J}^\top = \begin{bmatrix} 1 & 0 & \frac{-x}{z} \\ 0 & 1 & \frac{-y}{z} \end{bmatrix}$ where $z = \sqrt{1 - x^2 - y^2}$. The rows of $\boldsymbol{J}^\top$ define a parallelogram tangent to the hemisphere at (x, y, z) with an area equal to the square root of the sum of the squared 2×2 minors in $\boldsymbol{J}^\top$,[17][18] i.e.:

$$a = \sqrt{\begin{vmatrix} 0 & 1 \\ \frac{-x}{z} & \frac{-y}{z} \end{vmatrix}^2 + \begin{vmatrix} 1 & 0 \\ \frac{-x}{z} & \frac{-y}{z} \end{vmatrix}^2 + \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix}^2} = \sqrt{\frac{x^2}{z^2} + \frac{y^2}{z^2} + 1} = \sqrt{\frac{x^2 + y^2 + z^2}{z^2}} = \frac{1}{z}$$

We can now calculate $p(x, y, z)$ as $p(x, y, z) = \frac{p(x,y)}{a} = p(x, y)z$. Intuitively, $z = 1$ corresponds to the top of the hemisphere where the tangent parallelogram is parallel to B^2 and there is no ex-

[17] This procedure resembles the "formal determinant" method for computing the cross product: https://en.wikipedia.org/wiki/Cross_product#Matrix_notation.

[18] The area of the parallelogram can also be computed by taking the square root of the determinant of the Gram matrix $\boldsymbol{G} = \boldsymbol{J}^\top \boldsymbol{J}$.

pansion/dilution. In contrast, as $z \to 0$, the tangent parallelogram becomes increasingly "stretched" leading to greater dilution of $p(x, y)$.

A.3 Generating samples

Generating samples with IPDF simply involves sampling from the probability distribution defined by IPDF over the equivolumetric grid. For AQuaMaM, the sampling procedure uses a conditional version of the algorithm found in Listing 2. Python pseudocode for the conditional AQuaMaM sampling procedure can be found in Listing 3.

```
def sample_q(AQuaMaM, image):
    # The q_c_hats are used for generating the bins, but are not the
    final q_cs.
    q_c_hats = []
    cell_edges = []
    for i in range(3):
        bin_probs = AQuaMaM.get_probs(image, q_c_hats)
        l_q_c = categorical(bin_probs)
        (lower, upper) = AQuaMaM.bin_edges[l_q_c]
        q_c_hats.append(uniform(lower, upper))
        cell_edges.append((lower, upper))

    (q_x, q_y, q_z) = rejection_sample(cell_edges)
    q_w = sqrt(1 - q_x**2 - q_y**2 - q_z**2)
    return (q_x, q_y, q_z, q_w)
```

Listing 3: Python pseudocode for generating a unit quaternion sample conditioned on an image with AQuaMaM.

A.4 AQuaMaM training details

A.4.1 Toy experiment

For the toy dataset, the hyperparameters for the AQuaMaM model were $d_{\text{model}} = 64$, $P = 196$ (i.e., the number of 16×16 patches in a 224×224 image), eight attention heads, $d_{\text{ff}} = 256$ (the dimension of the inner feedforward layers of the Transformer), three Transformer layers, $L = 6$, g_{q_c} MLPs with 16, 32, and 64 output nodes and Gaussian Error Linear Unit (GELU) nonlinearities (Hendrycks & Gimpel, 2016), and $N = 50{,}257$. I used the Adam optimizer (Kingma & Ba, 2015) with an initial learning rate of 10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-9}$ to update the model's parameters. The learning rate was reduced by half every time the validation loss did not improve for five consecutive epochs. The model was implemented in PyTorch and trained on a single Tesla V100 GPU for $\sim$2.5 hours (624 epochs) where each epoch consisted of 40,000 training samples, and the evaluation loss was used for early stopping.

A.4.2 Die experiment

For the die dataset, the hyperparameters for the AQuaMaM model were $d_{\text{model}} = 512$, $P = 196$, eight attention heads, $d_{\text{ff}} = 2{,}048$, six Transformer layers, $L = 6$, g_{q_c} MLPs with 128, 256, and 512 output nodes and GELU nonlinearities, and $N = 500$. In total, this AQuaMaM model had 20,000,756 parameters. The model was trained using an identical optimization strategy to the toy experiment, and was trained for $\sim$2 days (121 epochs).

A.5 Generating predictions

Generating a prediction with IPDF simply involves taking the highest probability element from the probability distribution defined by IPDF over the equivolumetric grid. For AQuaMaM, generating a deterministic prediction is similar to the steps for generating a sample (See Listing 3), except the maximum probability bin is always selected and q_c is set as the midpoint of the selected bin rather than sampled from it. Python pseudocode for the conditional AQuaMaM sampling procedure can be found in Listing 4.

```python
def predict_q(AQuaMaM, image):
    # The q_c_hats are used for selecting the bins, but are not the final
     q_cs.
    q_c_hats = []
    for i in range(3):
        l_q_c = AQuaMaM.get_probs(image, q_c_hats).argmax()
        (lower, upper) = AQuaMaM.bin_edges[l_q_c]
        q_c_hats.append((lower + upper) / 2)

    if norm(q_c_hats) <= 1:
        q_cs = q_c_hats
    else:
        q_cs = normalize(q_c_hats)

    q_w = sqrt(1 - q_x**2 - q_y**2 - q_z**2)
    return (q_x, q_y, q_z, q_w)
```

Listing 4: Python pseudocode for deterministically predicting a unit quaternion given an image with AQuaMaM.

A.6 Visualizing a distribution for an object with continuous symmetry

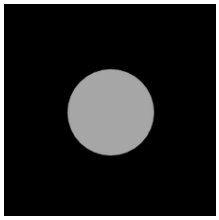


Figure 8: An example image from the "infinite" cylinder dataset.

Using the same hyperparameters from the toy experiment (Section A.4), I trained an AQuaMaM model on an "infinite" cylinder dataset. Specifically, the cylinder[19] was oriented such that, in the canonical view, one of the circular bases was facing directly towards the camera, i.e., the captured image was simply a circle (see Figure 8). Each training rotation was generated by randomly sampling a roll angle from $[-\pi, \pi]$, i.e., all associated images featured the same circle seen in Figure 8. Figure 9 visualizes the distribution learned by AQuaMaM using the same visualization technique described in Figure 7.

[19]The 3D cylinder model can be found at: `https://github.com/jlamarche/Old-Blog-Code/blob/master/Wavefront OBJ Loader/Models/Cylinder.obj`.

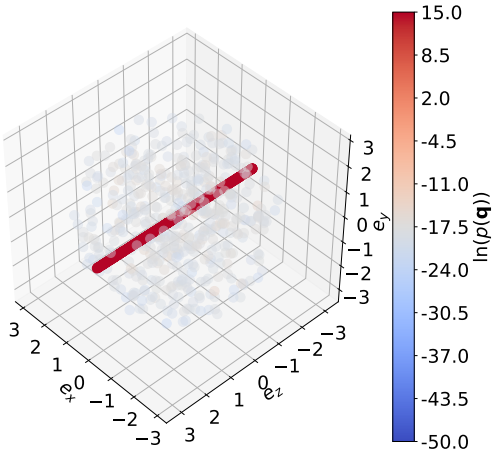


Figure 9: AQuaMaM is also capable of learning distributions for continuous symmetries, as shown here with the distribution learned by AQuaMaM on the "infinite" cylinder dataset. 1,000 rotations are depicted in the plot, 500 of which were obtained by randomly sampling 500 roll angles from $[-\pi, \pi]$ and another 500 of which were randomly selected from the uniform distribution on $\mathbf{SO}(3)$.