Adaptive Sampling for Continuous Group Equivariant Neural Networks

Berfin Inal [1 2] **Gabriele Cesa** [3 2]

Abstract

Steerable networks, which process data with intrinsic symmetries, often use Fourier-based nonlinearities that require sampling from the entire group, leading to a need for discretization in continuous groups. As the number of samples increases, both performance and equivariance improve, yet this also leads to higher computational costs. To address this, we introduce an adaptive sampling approach that dynamically adjusts the sampling process to the symmetries in the data, reducing the number of required group samples and lowering the computational demands. We explore various implementations and their effects on model performance, equivariance, and computational efficiency. Our findings demonstrate improved model performance, and a marginal increase in memory efficiency.

1. Introduction

Symmetry processing holds a significant importance in deep learning, since many real-world datasets inherently exhibit symmetrical properties. Research in integrating equivariance into deep architectures, such as steerable CNNs and Group Convolutional Neural Networks (GCNNs), has become a significant focus (Cohen & Welling, 2016a;b; Worrall et al., 2016; Thomas et al., 2018; Weiler et al., 2018a). While traditional CNNs offer translation equivariance, the first GCNNs (Cohen & Welling, 2016a) extended this to rotations and reflections, though limited to small discrete groups like C_4 and D_4 due to computational limits.

Steerable CNNs, introduced by Cohen & Welling (2016b), use steerable filters to achieve equivariance to larger groups with reduced computational demands. In the design of steerable CNNs, choosing the right activation function presents a key challenge. While pointwise nonlinearities have proven to be highly effective, they demand sampling from the entire group, requiring some form of discretization for continuous groups (Franzen & Wand, 2021; de Haan et al., 2021; Cesa et al., 2022). This introduces a trade-off between compute and performance: using more samples in the discretization improves model stability and equivariance, yet it also raises the computational and memory cost of the layer.

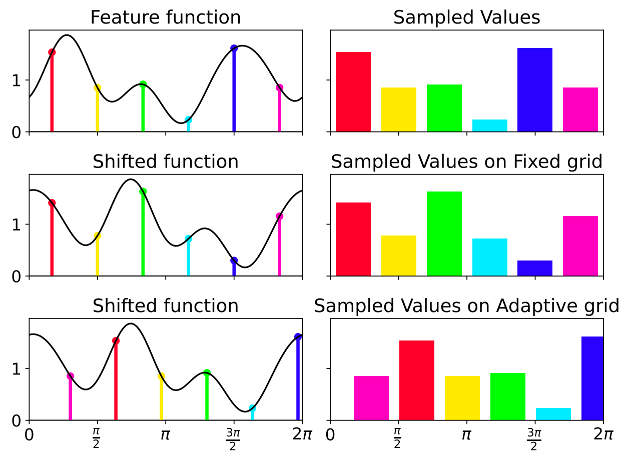

Figure 1. Behaviour of sampled values from a translated feature when using a fixed and an adaptive grid. *Left column*: the continuous feature function as well as the location of the grid samples (in colors). *Right column*: the measured values at the grid locations. When the function is shifted, the sampled values on the *fixed grid* change; these samples can not accurately capture translations smaller than the grid resolution. Instead, the *adaptive grid* is translated together with the input function, ensuring the measured values are constant (compare columns of the same color in the first and third row). Information about the phase is preserved by the adaptive grid itself (note the colored grid points translates too).

In our work, we aim to improve the sampling process within these pointwise non-linearities by using an adaptive grid approach. Traditionally, group sampling is done on a fixed grid, where samples are generated at model initialization and then cached for use throughout the network. However, our method dynamically *adjusts the sampling grid to align with the input data*. More precisely, we propose predicting the sampling grid from the input in an equivariant way: this ensures that transformed versions of the same input

[1]Work done during internship in QUVA-Lab. All datasets were downloaded/generated in QUVA-Lab [2]University of Amsterdam [3]Qualcomm AI Research, an initiative of Qualcomm Technologies, Inc. Correspondence to: Berfin Inal <s.berfininal@gmail.com>, Gabriele Cesa <gcesa@qti.qualcomm.com>.

Proceedings of the Geometry-grounded Representation Learning and Generative Modeling at 41st International Conference on Machine Learning, Vienna, Austria. PMLR Vol Number, 2024. Copyright 2024 by the author(s).

are processed in the non-linear layers using accordingly transformed versions of the same sampling grid. As proven in Sec. 3.3, this guarantees that the activation layer is always perfectly equivariant regardless of the number of samples employed. Fig 1 provides an intuitive visualization of this idea.

We present various implementations of the adaptive grid, focusing on strategies to share it efficiently and reduce the overall computational costs. Our empirical analysis examines how varying the number of samples, compared to a fixed grid, affects model performance, equivariance error, and computational efficiency. Our findings indicate improved model performance and a marginal increase in memory efficiency. Lastly, we explore the limitations, computational considerations, and potential advancements of our approach.

2. Steerable Convolutional Neural Networks

Steerable Features In steerable CNNs, feature spaces are defined as spaces of *steerable feature fields* $f : \mathbb{R}^n \to \mathbb{R}^{d_\rho}$, which assigns a d_ρ-dimensional vector $f(x) \in \mathbb{R}^{d_\rho}$ to each data point $x \in \mathbb{R}^n$. These feature fields are associated with a transformation law, which describes how they are transformed by the action of the group G. The transformation law of a d_ρ-dimensional vector is defined by a group representation $\rho : G \to \mathbb{R}^{d_\rho \times d_\rho}$:

$$[g.f](x) := \rho(g)f(g^{-1}.x) \tag{1}$$

where ρ specifies how the d_ρ channels of each feature vector $f(x)$ mix. In practice, the entire feature space can be defined as the *direct sum* $\bigoplus_i f_i$ of multiple individual feature fields f_i, which transforms according to the direct sum representation $\rho := \bigoplus_i \rho_i$, enabling each field to transform independently of others.

An example of feature fields are scalar fields, which transform according to the trivial representation $\psi_0(g) = 1$. GCNNs are special cases of steerable CNNs using intermediate features that transform according to the *regular representation* of G; see Sec. 3. Steerable CNNs generalize GCNNs by allowing feature fields to transform according to more complex geometric types. This is achieved by the steerable kernel which satisfy the *steerability constraint*. Given the feature fields with the input type $\rho_{\text{in}} : G \to \mathbb{R}^{d_{\text{in}} \times d_{\text{in}}}$ and the output type $\rho_{\text{out}} : G \to \mathbb{R}^{d_{\text{out}} \times d_{\text{out}}}$, the G-steerable convolutional kernel $K : \mathbb{R}^n \to \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ satisfies the steerability constraint $\forall g \in G, x \in \mathbb{R}^n$:

$$K(x) = \rho_{\text{out}}(g)K(g^{-1}x)\rho_{\text{in}}(g)^{-1}. \tag{2}$$

Any equivariant kernel can be described using a $G-$steerable kernel basis, as detailed in Cesa et al. (2022). For parametrization, these basis elements are combined with learnable weights. Since each pair of ρ_{in} and ρ_{out} requires a unique solution, the kernel constraint can be simplified by breaking the input and output representation into simpler and non-reducible components:

Irrep Decomposition Any orthogonal representation $\rho : G \to \mathbb{R}^{d_\rho \times d_\rho}$ can be decomposed as a direct sum of mutually orthogonal *irreducible representations (irreps)* $\rho(g) = Q^T(\bigoplus_{i \in I} \psi_i(g))Q$, where $\psi_i \in \hat{G}$ is an irrep, $\hat{G}$ is the set of irreps of the group G, I is an index set ranging over $\hat{G}$, and Q is the change of basis. The kernel constraint solution for arbitrary ρ_{in} and ρ_{out} can be derived from the irreps decompositions of the two representations, as detailed by Weiler & Cesa (2019). A more comprehensive discussion on irrep decomposition is presented in Section A.1.

In this work, we focus on convolutional networks operating on point clouds and 3D voxel data, and the compact group $SO(3)$. We employ *dense convolutions* for 3D volumetric voxels, where filters are applied across all input data. For point clouds, we leverage message passing between neighboring points to effectively extract features from the unstructured data.

Nonlinear Layers In equivariant networks, nonlinear layers must also satisfy the equivariance constraint. While the transformation law of most feature field types do not commute with point-wise nonlinearities, pointwise nonlinearities acting on the norm of each field preserves their rotational equivariance. This type of nonlinearity is called *Norm nonlinearity*. Moreover, *Gated nonlinearities*, which scales the norm of the feature fields using gated scalars, can be considered as a form of norm nonlinearity. Lastly, *Tensor product* nonlinearity combines feature vectors through tensor operations, which inherently introduces polynomial nonlinearities. Formal definitions of those nonlinear layers are provided in Sec. B.

3. Reqular and Quotient Nonlinearities

Although norm and gated nonlinearities satisfy the equivariance constraint, it has been observed that they typically under-perform compared to pointwise nonlinearities (Weiler & Cesa, 2019). Unfortunately, while they are straightforward to implement for discrete groups, they require some form of discretization when considering continuous groups. de Haan et al. (2021) first employed *pointwise non-linearities* for the 2D rotation groups by leveraging a discretized Fourier transformation. Previously, Cohen et al. (2018) used a similar idea to implement group convolution networks over the 3D rotation group. In this section, we delve in the theory behind this idea.

Peter-Weyl : Orthonormal Basis of Matrix Coefficients According to the Peter-Weyl theorem (Thm. 2), the matrix coefficients of (complex) irreps of a compact group G, i.e.

$$\left\{ \sqrt{d_\psi}\psi_{ij}(g) : G \to \mathbb{C} \mid \psi \in \widehat{G}, 1 \leq i, j \leq d_\psi \right\} \tag{3}$$

form a complete orthonormal basis for the vector space $L^2(G)$ of square-integrable functions over G, where d_ψ is the dimensionality of the irrep ψ and $\sqrt{d_\psi}$ is a scalar factor to normalize the basis. This generalizes the classical *Fourier transform* of periodic functions (corresponding to $G = SO(2) \cong U(1)$), where the notion of frequencies is replaced by the irreducible representations $\widehat{G}$ of the group G. See Sec. A.1 for more details.

Assumption In this work we are mostly interested in the group $G = SO(3)$ and real valued functions. For this reason, from now, we will assume only real valued representations (and irreps) and functions. The Peter-Weyl theorem above requires some minor adaptations in this case for certain groups, but it still holds exactly for $G = SO(3)$. See Cesa et al. (2022) for a precise statement of the theorem in this case and its implications for other groups. Moreover, as common in the literature, we assume all representations to be orthogonal, satisfying $\rho(g^{-1}) = \rho(g)^{-1} = \rho(g)^T$.

Fourier transform For convenience, the Fourier coefficients of a function $f \in L^2(G)$ are typically aggregated by the irrep they belong to. Hence, the Fourier transform $\hat{f}$ is matrix-valued:

$$\hat{f}(\psi) := \int_G f(g)\sqrt{d_\psi}\psi(g)dg \in \mathbb{R}^{d_\psi \times d_\psi} \tag{4}$$

for all $\psi \in \widehat{G}$. Similarly, the inverse Fourier transform is defined as

$$f(g) = \sum_{\psi \in \hat{G}} \sqrt{d_\psi} Tr(\hat{f}(\psi)\psi(g)^T) \tag{5}$$

where $Tr(AB^T)$ is the standard matrix (Frobenius) inner product.

Regular representation The group G carries an orthogonal action on $L^2(G)$ by translating functions via $g : f \mapsto g.f$, with $[g.f](h) := f(g^{-1}h)$. This action is the *regular representation* ρ of the group; steerable CNNs employing regular representations as intermediate feature type are equivalent to group convolution networks, hence the popularity of this design choice. Unfortunately, this construction is not practical when G is a continuous group since the space $L^2(G)$ is infinite dimensional. The Fourier transform just introduced provides an effective solution as *bandlimited functions* in $L^2(G)$ can be represented by a finite number of Fourier coefficients, i.e. a finite subset $\tilde{G} \subset \widehat{G}$ of the group's irreps.

Quotient representation A similar construction exists when considering functions over a *quotient space* $Q = G/H := \{gH | g \in G\}$ (with $H < G$ a subgroup) rather than G itself. Recall that an element of Q is *coset*, i.e. an equivalence class of elements of the form $gH = \{gh | h \in H\}$. A function $f \in L^2(Q)$ is then equivalent to a function in $L^2(G)$ invariant under the right action of the subgroup H, i.e. $f(gh) = f(g)$ for all $h \in H$. In particular, this subspace $L^2(Q) \subset L^2(G)$ is typically spanned by a smaller set of matrix coefficients than $L^2(G)$, providing a more compact representation than the regular representation. A standard example is given by spherical signals for $Q = S^2 \cong SO(3)/SO(2)$, which we will also use later in this work. Because the regular representation is a special case for $Q = G/\{e\} \cong G$, we will often use Q to refer to the underlying space on which functions are defined and generally discuss quotient non-linearities. Moreover, since the quotient space Q is not necessarily finite, the previous considerations about bandlimiting the regular representation apply for quotient representations too.

Finally, we note that this Fourier transform is precisely the change of basis matrix which performs the *irreps decomposition* described in Sec. 2 of the regular or quotient representation. Indeed, the Fourier coefficients jointly transform under a direct sum of irreps, in the same way each frequency component in the Fourier transform of a periodic function transforms independently when the function is translated. This is an important aspect for our method, so we derive this decomposition explicitly in Sec. 3.1.

Pointwise Non-Linearity via Fourier transform Representing an intermediate (bandlimited) regular or quotient type feature by its Fourier coefficients requires an additional step to incorporate pointwise non-linearities in the network. This is done by composing the activation function σ with a *sampling step* (via a discretized inverse Fourier transform, or IFT) and a Fourier transform (FT):

$$f'(x) = [\text{FT} \circ \sigma \circ \text{IFT}(f)](x) \tag{6}$$

where FT recovers the bandlimited coefficients of the output function from a finite set of samples, while IFT specifically computes the value of the bandlimited function $f(x)$ at selected points in space Q. It is important to note that this operation is only **approximately equivariant** and the degree of equivariance mostly relies on the number N of samples used and their distribution over the group. Uniform and well-spaced sampling in Q is necessary to ensure accurate representation and generalization over the group transformations. To create a *sample set* $\Gamma \subset Q$, it is possible to use elements from a discrete subgroup of G, or alternatively distribute N points within Q by simulating a particles repulsion system as proposed by Bekkers (2021) and implemented in Cesa et al. (2022).

3.1. Irreps Decomposition of Bandlimited Quotient Representations

In this section, we give an explicit construction of the irreps decomposition of a band-limited regular or quotient representation. This is necessary for implementing them in the framework of steerable CNNs, since solving the kernel constraint relies on the irreps decomposition of the intermediate features.

As previously mentioned, for a quotient space $Q = G/H$, a function $f \in L^2(Q) \subset L^2(G)$ is equivalent to a function over G invariant with respect to right action by H. By considering the Fourier transform Eq. 5 of f, we find that

$$f(gh) = \sum_{\psi \in \hat{G}} \sqrt{d_\psi} Tr(\hat{f}(\psi)\psi(h)^T \psi(g)^T) \tag{7}$$

Hence, the function has the desired invariance only if $\hat{f}(\psi)\psi(h)^T = \hat{f}(\psi)$ for any $h \in H$. Under a proper choice of basis[1] for $\psi \in \widehat{G}$, only certain columns of $\hat{f}(\psi)$ satisfy this invariance; then, let $P_\psi \in \mathbb{R}^{d_\psi \times q_\psi}$ be a mask matrix selecting only those q_ψ columns of ψ that are invariant. For notational convenience, we also assume that $\hat{f}(\psi) \in \mathbb{R}^{d_\psi \times q_\psi}$ contains only these q_ψ invariant columns (the other columns would contain only zero coefficients). Then, we can express the function f as:

$$f(gh) = \sum_{\psi \in \hat{G}} \sqrt{d_\psi} Tr(\hat{f}(\psi) P_\psi^T \psi(gh)^T) \tag{8}$$

$$= \sum_{\psi \in \hat{G}} \sqrt{d_\psi} Tr(\hat{f}(\psi) P_\psi^T \psi(g)^T). \tag{9}$$

Next, using the following identities $Tr(AB^T) = \text{vec}(B)^T\text{vec}(A)$ and $\text{vec}(AB) = (\bigoplus_i^d A)\text{vec}(B)$, where d is the number of columns of B and $\bigoplus_i^d A$ is the direct sum of d copies of A (i.e. stacked along the diagonal), we write:

$$f(g) = \sum_{\psi \in \hat{G}} \sqrt{d_\psi} \left(\left(\bigoplus^{q_\psi} \psi(g) \right) vec(P_\psi) \right)^T vec(\hat{f}(\psi)) \tag{10}$$

$$f(g) = \hat{\delta}^T \bigoplus_{\psi \in \hat{G}} \bigoplus^{q_\psi} \psi(g)^T \hat{f} \tag{11}$$

[1]This invariance constraint identifies the H-invariant subrepresentations of ψ. Indeed, the irrep ψ can be thought as a representation of H via *restriction*. This representation is not necessarily irreducible anymore but can be further reduced in a direct sum of irreps of H. Check the discussion about induced representations in Cesa et al. (2022) for more details. For simplicity, here we assume ψ is already expressed in a basis such that its restriction to H already has an irreps direct sum structure. In this way, the invariant components of $\hat{f}(\psi)$ corresponds to its columns which are associated with a trivial representation of H.

with $\hat{\delta} = \bigoplus_\psi \bigoplus^{q_\psi} \sqrt{d_\psi} vec(P_\psi)$ and $\hat{f} = \bigoplus_\psi vec(\hat{f}(\psi))$, where $\bigoplus$ concatenates vectors. In this formulation, $\hat{\delta}$ can be interpreted as a vector with the Fourier coefficients of an indicator function on the coset $eH \in Q$ - i.e. $\delta_H(g)$ is non-zero iff $g \in H$ - or just a Dirac delta centered on the origin $e \in G$ in case of a regular representation. Eq. (11) can be simplified further to

$$f(g_i) = \hat{\delta}^T \rho(g_i)^T \hat{f} \tag{12}$$

where $\rho(g_i)^T = \bigoplus_{\psi \in \hat{G}} \bigoplus^{q_\psi} \psi(g_i)^T$.

In Eq. (12), $\rho(g_i)$ represents the action of group element g_i on the vector containing all Fourier coefficients $\hat{f}$. When we use a finite subset of irreps $\tilde{G} \subset \widehat{G}$, this vector is finite dimensional and we refer to ρ as a *band-limited* quotient (or regular) representation and we denote its size as $F = \sum_{\psi \in \tilde{G}} d_\psi \cdot q_\psi$.

3.2. Activation Layer

In this section we construct the activation layer described in Eq. 6 by leveraging a finite set of N samples $\Gamma \subset Q$. We first group the terms $A_i := \rho(g_i)\hat{\delta}$ in Eq. (12):

$$f(g_i) = A_i^T \hat{f} \tag{13}$$

Note that $A_i := \rho(g_i)\hat{\delta}$ is the Fourier transform of an indicator function centered on the coset $g_i H \in Q$. Hence, we define the matrix A with a row A_i for each $g_i H$ in the sampling set $\Gamma \subset Q$. Conversely, the columns of A correspond to the finite subset of irreps $\tilde{G}$ used for band-limiting. See Fig. 2. Essentially, the matrix A carries out the discretized Inverse Fourier Transform (IFT) of $\hat{f}$; we refer to this matrix as the **sampling matrix**, as it integrates the group sampling.

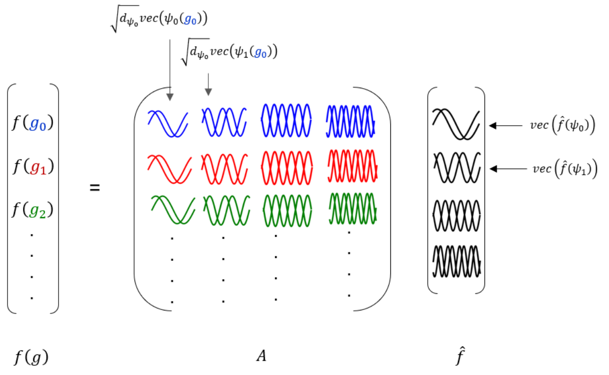

Figure 2. Discretized Inverse Fourier transform, where $\hat{f} = \bigoplus_\psi vec(\hat{f}(\psi)) \in \mathbb{R}^F$ and $A_i := \rho(g_i)\hat{\delta} \in \mathbb{R}^F$. N is the number of group samples, while F is the size of the representation. Each row in A corresponds to a single group sample, while each column corresponds to an irrep's matrix coefficients.

Fourier and inverse Fourier transform are inverse operations; as A implements the inverse Fourier transform, it suggests using its inverse to performing a discretized Fourier transform. Because the activation function σ introduces small high-frequency components in the function, we typically rely on oversampling to control *aliasing*, i.e. $N > F$ and, therefore, we leverage the *pseudo-inverse* matrix $A^\dagger$, rather than the matrix inverse. We provide more details on Fourier and inverse Fourier transforms in Apx. A.2.

The activation layer in Eq. 6 can then be formulated using the sampling matrix A as

$$\hat{f}' = A^\dagger \sigma(A\hat{f}(x)) \tag{14}$$

where $\hat{f}(x)$ is the input feature vector, and σ is a pointwise nonlinearity such as ELU.

Recall that this operation is only *approximately equivariant* due to the *aliasing* effect mentioned ealier. To improve the equivariance of the nonlinearity layer, it's crucial to increase and uniformly distribute the number of group samples over the space Q; however, as explored in Sec. 4, the computational cost of this operation scales with the number N of samples.

3.3. Adaptive Sampling Matrix

In a typical architecture, a predefined number of samples is stored for repeated use across the network. In this work, we dynamically learn the sampling matrix as an equivariant function $A(x)$ of the model's input x to ensure full equivariance regardless of the number N of samples:

$$A(g.x) = A(x)\rho(g)^T\ . \tag{15}$$

This strategy reduces the need for extensive sampling, effectively decreasing both the number of group samples N and the dimensions of A, thereby can lower the overall computational complexity. Additionally, we expect this approach to generate more expressive sampling matrices that can be specifically tailored to each input's salient directions.

In practice, we can generate the sampling matrix A using an equivariant layer processing the model input or intermediate features, depending on the implementation choice. We explore different strategies in Sec. 5.

Unfortunately, computing the pseudoinverse (as well as the inverse) of a matrix can be computationally expensive and difficult to backpropagate through, especially for large matrices. However, if sufficiently many samples N are used, the Peter-Weyl theorem ensures the columns of A become orthogonal to each other. Assuming the columns are sufficiently orthogonal, we can then approximate the pseudoinverse as $A(x)^\dagger \approx \frac{1}{N}A(x)^T$, further improving the computational efficiency. The final layer takes the form

$$\tilde{f} = \frac{1}{N}A(x)^T\sigma(A(x)\hat{f}(x)). \tag{16}$$

If the sampling matrix is generated by an equivariant layer, the nonlinear layer in Eq. 16 is fully equivariant regardless of the number N of samples considered. We prove this result in Apx. C.

4. Computational aspects

Assume an input feature field $\hat{f}$ comprising c ρ-fields, i.e. $\hat{f}(x) = \bigoplus_i^c f_i(x) \in \mathbb{R}^{cF}$, where ρ is a quotient (or regular) representation of size F. Accordingly, the computational complexity for the inverse Fourier transform $A\hat{f}$ and the Fourier transform $A^\dagger\sigma(A\hat{f})$ is $O(cNF)$, with N representing the number of group samples. The pointwise nonlinearity σ has a complexity of $O(cN)$. For a data sample containing m pixels / spatial samples, this yields a total complexity for the nonlinear layer of $O(mcNF) + O(mcN) + O(mcFN) = O(mcNF)$, indicating that the layer's computational complexity scales with the number of group samples N.

In the context of the convolutional layer, let k denote the computational cost for a single convolution operation involving one input and one output channel, while m represents the number of pixels / spatial samples in the input grid. For a convolutional layer with c input and output channels, the complexity becomes $O(kmc^2F^2)$. Then, the model has overall complexity $O(kmc^2F^2) + O(mcNF)$. Whenever $kcF >> N$, the complexity is dominated by the convolution term $O(kmc^2F^2)$, making the impact of the number of samples N less relevant, so we can not expect strong improvements in the final runtime. However, because of oversampling, $N > F$ and, therefore, the intermediate activations of the non-linear layers are larger than the features processed by the convolution layers and can increase the overall memory requirements of the model. We study the effect of oversampling and of adopting our method on memory usage in our experiments. Finally, in Sec. 8 we discuss potential future research directions to employ our method and benefit more from its reduced sampling rate.

5. Implementation

We can employ two main approaches to generate the sampling matrix A. Either a unique set of sampling matrices can be generated for each nonlinear layer in the network, or a set of matrices can be generated only once, and it is processed by all following nonlinear layers. In our implementation, we focused on the latter approach to obtain more expressive A and to increase computational efficiency. Empirical results also showed that the latter approach indeed outperforms the former in terms of efficiency.

Our approach generates a unique sampling matrix A for each point in the point cloud or for each pixel in 3D voxels. In both cases, the main architecture employs some form of

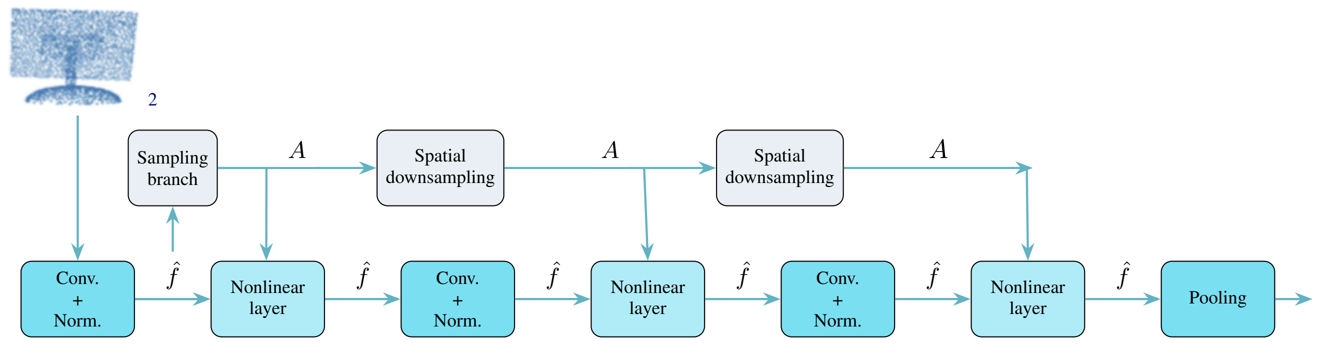

Figure 3. Architecture for point cloud processing. In this architecture, blue blocks at the bottom represents the main branch which process the point clouds and the gray blocks at the top correspond to sampling branch which generates the sampling matrix A and perform spatial downsampling accordingly. Although it is not illustrated in the figure, each convolutional block, which comprises the convolutional layer, batch normalization and the nonlinear layer, is followed by an equivariant MLP.

spatial downsampling. As the data is progressively downsampled through the layers, a compatible downsampling technique should also be applied to the sampling matrices to ensure their spatial resolution remains compatible with the feature space. Regarding the point clouds, this downsampling can be executed by simply indexing - i.e. retaining only the sampling matrices of the points that remain after downsampling. Alternatively, a convolutional layer can be used for a smoother downsampling for both data types.

Furthermore, the sampling matrix can be generated using either an equivariant MLP or an equivariant convolutional layer. The main advantage of employing a convolutional layer is its ability to utilize neighbouring information. While downsampling through convolutional layers is often needed for matrices generated by an equivariant MLP, it is not required for matrices produced by equivariant convolutional layers. Fig. 3 illustrates the architecture for the point cloud processing. The blue blocks at the bottom represent the main branch in the network, while the gray blocks at the top illustrates our approach, where the matrices are generated by processing the intermediate feature and downsampled through the network. In Sec. 7, we present the results from the selected architectures that demonstrate the best performance among various architectures. We also discuss the practical design choices in Apx. E.

6. Related Work

Equivariant Neural Networks The concept of group equivariance in convolutional networks was first introduced by Cohen & Welling (2016a), showing how CNNs can adapt to transformations like rotations and reflections. However, scaling to larger groups introduces computational challenges due to the need for extensive sampling. Steerable CNNs, also developed by Cohen & Welling (2016b), address the computational challenges by using the steerable kernels, which improves the flexibility of the network and reduces the computational overhead. Further exploration of steerable CNNs has been conducted in the 2D domain (Cohen & Welling, 2016b; Weiler & Cesa, 2019; Worrall et al., 2016; Weiler et al., 2018b), and their expansion into the 3D domain (Weiler et al., 2018a; Thomas et al., 2018; Esteves et al., 2020; Cesa et al., 2022) has demonstrated significant benefits in processing point clouds and graphs (Thomas et al., 2018; Anderson et al., 2019; Poulenard & Guibas, 2021).

Equivariant Nonlinearities One of the challenges in designing rotation equivariant networks is selecting appropriate activation functions. To maintain equivariance, nonlinearities must commute with the rotation of equivariant features, which cannot be achieved by conventional pointwise nonlinearities. To address this, norm and gated nonlinearities have been introduced by Thomas et al. (2018) and Weiler et al. (2018a), respectively, which apply to the norm of equivariant features and preserve rotation commutativity, but cannot capture directional information. An alternative approach, as proposed in (Anderson et al., 2019; Kondor et al., 2018), involves using tensor product operations on equivariant features as a form of nonlinearity, which is further discussed in (Kondor, 2018).

de Haan et al. (2021) introduce a nonlinearity built on Fourier transformation, where they interpret the features as Fourier coefficients of functions over the circle, which we also discussed in Sec. 3. In a concurrent work, Franzen & Wand (2021) propose a Fast Fourier Transform based approach to apply conventional pointwise nonlinearities on equivariant representations and obtain exact $SO(2)$ equivariance for polynomial functions.

Adaptive Sampling for Steerable Networks In conventional CNNs, convolutional operations are accompanied by downsampling methods such as pooling and strided convolutions. The fixed grid structure in downsampling methods can compromise the network's ability to handle shifted and rotated inputs, thereby disrupts the shift invariance and equivariance (Azulay & Weiss, 2019; Zhang, 2019). To address these limitations, Xu et al. (2021) introduced a new subsampling method that uses input-dependent grids instead of fixed ones, maintaining translation equivariance. Additionally, Chaman & Dokmanic (2021) proposed adaptive polyphase sampling (APS) to dynamically select downsampling grids. Building upon this, Rojas-Gomez et al. (2022) improves APS by making this selection process learnable, using neural networks. Kaba et al. (2023) took a different approach by using equivariant networks to predict input poses, allowing the network to revert inputs to a rotation invariant state for processing in non-equivariant models. Kim et al. (2023) expanded on this by predicting a distribution over possible poses, rather than a single pose.

In our study, we advance the concept of adaptive grids, previously applied in downsampling and canonicalization, by integrating it into Fourier-based nonlinearities, more specifically to the sampling process. Unlike the previous works that largely focused on utilizing global grids for downsampling or canonicalization, our approach takes a more localized perspective, learning a sampling grid at each spatial location.

7. Experiments

Methods that are explained in Sec 3.3 and in Sec. 5 are evaluated on two datasets, namely ModelNet10 (Wu et al., 2015) and NoduleMNIST3D (Yang et al., 2023). ModelNet10 (Wu et al., 2015) is a subset of the larger ModelNet40 which contains synthetic object point clouds, while NoduleMNIST3D (Yang et al., 2023) is a subset of the MedMNIST collection. All our implementations are based on the *escnn* library (Cesa et al., 2022).

ModelNet10 In our analysis, we benchmarked our method against the approach described in Poulenard & Guibas (2021), which combines *Tensor Field Networks (TFN)* (Thomas et al., 2018) with Fourier-based nonlinearities. In this study, we developed a similar model consisting of three convolutional blocks, each followed by a block of linear layers. We refer to this model as the *Base model*. Fig. 4 presents the test accuracies of the Base model with various nonlinearities. The figure demonstrates that the Base model with Fourier-based nonlinearities outperforms the models with other nonlinearities, such as norm and gated, provided a sufficient number of group elements are sampled. Additionally, the model with gated nonlinearity significantly outperforms the model with norm nonlinearity. This outcome is expected, since the gated nonlinearities are more proficient at processing directional information than norm nonlinearities. Furthermore, we have confirmed that the traditional PointNet++ framework lacks the capacity to handle equivariance adequately.

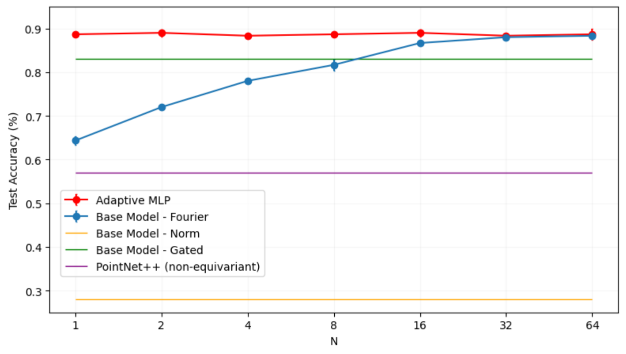

Figure 4. Test accuracy on ModelNet10, with respect to the number of group samples by model.

In our adaptive approach, we experimented with various downsampling and matrix generation methods, while keeping the main architecture consistent. We only included the best-performing model, which is referred to as the *Adaptive MLP*. Its architecture is depicted in Fig. 3. This model generates the sampling matrix through an equivariant MLP and performs spatial downsampling using convolutional layers. Fig 4 displays the performance of the models with different nonlinear layers, based on the number of group samples. Each model is trained three times, with three different predetermined seeds for initialization. Although the standard deviations among the runs are quite small, they are visualized in the figure. As shown, even with a single group sample, our Adaptive MLP outperforms the Base model. This improvement is attributed to its complete equivariance and enhanced expressiveness, achieved by introducing a tailored sampling matrix to each nonlinear layer.

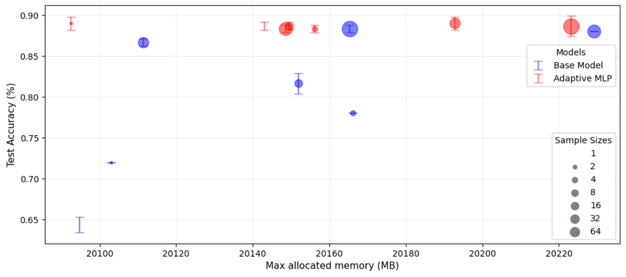

Figure 5. Test accuracies vs. memory cost by model. Results are computed on ModelNet10.

We can also observe the memory gain obtained by our approach in Fig. 5. Our Adaptive MLP with two group samples, located in the top left corner of the figure, outperforms other models in both test accuracy and computational efficiency. However, it is important to note that the computational gains are modest and do not increase proportionally with the number of group samples, as initially expected. Further analysis reveals that this can be attributed to the computational overhead in the convolutional layer, which diminishes the improvements gained from the nonlinear layer, making them less apparent in the overall model output. We discuss the computational limitations in more detail in Sec. 4.

NoduleMNIST3D We employ an equivariant architecture with dense convolutions and Fourier-based nonlinearities as a benchmark to evaluate the performance of our approach. This model will be referred to as the *Base model*. For the adaptive sampling approach, we have tried various implementations. Specifically, we examined three different methods for spatial downsampling of the sampling matrix.

Note that each model, with a specific number of group samples, is trained three times using different predetermined initialization seeds. In the first approach, convolutional layers are employed to perform all downsampling, and this model is referred to as *Adaptive Conv*. To further reduce the computational cost, in our second implementation, we replaced the convolutional layers performing downsampling with average pooling layers. This model is referred to as *Adaptive Pooling*. However, both *Adaptive Conv* and *Adaptive Pooling* were outperformed by the *Base model* in terms of test accuracy and computational efficiency. Our further analysis revealed that the memory overhead was primarily caused by the first convolutional layer in the sampling branch.

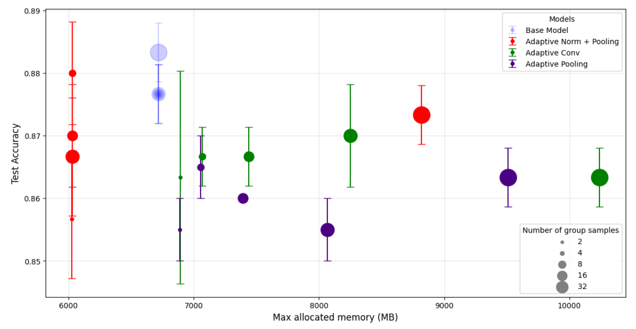

Figure 6. Test accuracy vs. memory cost by model. The results are computed on NoduleMNIST3D datastet. Each model is trained three times with three different predetermined seeds for initialization. Standard deviation among those runs are displayed in the figure. The Base model is illustrated with transparency to display the overlap in the plot.

This layer processes the initial intermediate features from the main branch and returns the first set of generated sampling matrices to the first activation layer (see Fig. 7). To address this issue, we replaced the Fourier-based nonlinearity in the first convolutional block with the norm nonlinearity, and started generating the sampling matrix from the second convolutional layer onwards. This implementation is referred to as *Adaptive Norm + Pooling* in the figures. The *Adaptive Norm + Pooling* model successfully resolved the convolutional overhead issue, leading to improved computational efficiency.

In Fig. 6, we present the test accuracy and the memory consumption for the different implementations. *Adaptive Conv* and *Adaptive Pooling*, represented by green and indigo points respectively, show lower test accuracies and higher computational demands compared to the base model as seen mainly on the right side of the figure. Conversely, Adaptive Norm + Pooling (red points) successfully reduces computational demands while achieving comparable test accuracies to the base model. It is important to note that the standard deviation values for the Adaptive Norm + Pooling with a smaller number of group samples are notably higher compared to the other models, which suggests more variability in their performance.

8. Discussion and Conclusion

In the previous sections, we presented the findings of our empirical analysis. When applied to point cloud data, our approach demonstrated improved classification accuracy compared to models that did not incorporate our method, demonstrating the benefits of maintaining exact equivariance and leveraging a local adaptive sampling grid. We also observed a marginal improvement in computational cost. However, when applied to voxel data, which lack exact symmetries due to discretization, our approach achieved moderate computational efficiency and comparable classification accuracy.

To better understand the limitations in computational efficiency, we analyzed the computational costs of each network layer using profiling tools. Our findings reveal that convolutional layers have a significantly higher computational load compared to nonlinear layers, particularly in point cloud processing. The cost ratio between convolutional and nonlinear layers varies throughout the network, with convolutional layers being up to seven times more computationally demanding than nonlinear layers. This substantial difference dominates the overall cost of the model, making the improvements in the nonlinear layers less noticeable in the overall computational cost of the model.

Further Improvements and Future Work To address the computational overhead in the convolutional layers, future works could adopt different model designs which aim to reduce the memory and computational overhead of convolutional layers, making the benefits of our method more observable. For instance, MobileNetV2 (Sandler et al., 2019) uses separable depthwise convolutions to significantly reduce computational load and could highlight the impact of nonlinear layers. Similarly, implementing the full model from Poulenard & Guibas (2021), which uses entire MLPs as pointwise activations, can introduce higher complexity but benefit from a small and adaptive grid to manage computational cost and reduce aliasing effects. Finally, another promising alternative is the architecture from Knigge et al. (2022), which leverages a separable group convolution.

Acknowledgement

We express our gratitude to the anonymous reviewers and Dr. Erik Bekkers for their invaluable feedback and insightful discussions.

References

Anderson, B., Hy, T.-S., and Kondor, R. Cormorant: Covariant molecular neural networks, 2019.

Azulay, A. and Weiss, Y. Why do deep convolutional networks generalize so poorly to small image transformations?, 2019.

Bekkers, E. J. B-spline cnns on lie groups, 2021.

Bekkers, E. J., Vadgama, S., Hesselink, R. D., van der Linden, P. A., and Romero, D. W. Fast, expressive se(n) equivariant networks through weight-sharing in position-orientation space, 2023.

Cesa, G., Lang, L., and Weiler, M. A program to build e(n)-equivariant steerable CNNs. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=WE4qe9xlnQw.

Chaman, A. and Dokmanic, I. Truly shift-invariant convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3773–3783, June 2021.

Cohen, T. S. and Welling, M. Group equivariant convolutional networks, 2016a.

Cohen, T. S. and Welling, M. Steerable cnns, 2016b.

Cohen, T. S., Geiger, M., Köhler, J., and Welling, M. Spherical cnns. *arXiv preprint arXiv:1801.10130*, 2018.

de Haan, P., Weiler, M., Cohen, T., and Welling, M. Gauge equivariant mesh cnns: Anisotropic convolutions on geometric graphs, 2021.

Esteves, C., Allen-Blanchette, C., Makadia, A., and Daniilidis, K. Learning so(3) equivariant representations with spherical cnns. *International Journal of Computer Vision*, 128(3):588–600, March 2020. ISSN 0920-5691. doi: 10.1007/s11263-019-01220-1. Publisher Copyright: © 2019, Springer Science+Business Media, LLC, part of Springer Nature.

Franzen, D. and Wand, M. General nonlinearities in so(2)-equivariant cnns. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 9086–9098. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/4bfbd52f4e8466dc12aaf30b7e057b66-Paper.pdf.

Kaba, S.-O., Mondal, A. K., Zhang, Y., Bengio, Y., and Ravanbakhsh, S. Equivariance with learned canonicalization functions, 2023.

Kim, J., Nguyen, T. D., Suleymanzade, A., An, H., and Hong, S. Learning probabilistic symmetrization for architecture agnostic equivariance, 2023.

Knigge, D. M., Romero, D. W., and Bekkers, E. J. Exploiting redundancy: Separable group convolutional networks on lie groups, 2022.

Kondor, R. N-body networks: a covariant hierarchical neural network architecture for learning atomic potentials, 2018.

Kondor, R., Lin, Z., and Trivedi, S. Clebsch–gordan nets: a fully fourier space spherical convolutional neural network. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/a3fc981af450752046be179185ebc8b5-Paper.pdf.

Poulenard, A. and Guibas, L. J. A functional approach to rotation equivariant non-linearities for tensor field networks. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13169–13178, 2021. doi: 10.1109/CVPR46437.2021.01297.

Rojas-Gomez, R. A., Lim, T.-Y., Schwing, A. G., Do, M. N., and Yeh, R. A. Learnable polyphase sampling for shift invariant and equivariant convolutional networks, 2022.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., and Riley, P. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds, 2018.

Weiler, M. and Cesa, G. General $e(2)$-equivariant steerable cnns, 2019.

Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. *CoRR*, abs/1807.02547, 2018a. URL http://arxiv.org/abs/1807.02547.

Weiler, M., Hamprecht, F. A., and Storath, M. Learning steerable filters for rotation equivariant cnns. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 849–858, 2018b. doi: 10.1109/CVPR.2018.00095.

Worrall, D. E., Garbin, S. J., Turmukhambetov, D., and Brostow, G. J. Harmonic Networks: Deep Translation and Rotation Equivariance. *ArXiv e-prints*, December 2016.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. 3d shapenets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1912–1920, 2015. doi: 10.1109/CVPR.2015.7298801.

Xu, J., Kim, H., Rainforth, T., and Teh, Y. W. Group equivariant subsampling, 2021.

Yang, J., Shi, R., Wei, D., Liu, Z., Zhao, L., Ke, B., Pfister, H., and Ni, B. Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Scientific Data*, 10(1):41, 2023.

Zhang, R. Making convolutional networks shift-invariant again. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7324–7334. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/zhang19a.html.

Contents

1 Introduction 1

2 Steerable Convolutional Neural Networks 2

3 Reqular and Quotient Nonlinearities 2

3.1 Irreps Decomposition of Bandlimited Quotient Representations 4

3.2 Activation Layer 4

3.3 Adaptive Sampling Matrix 5

4 Computational aspects 5

5 Implementation 5

6 Related Work 6

7 Experiments 7

8 Discussion and Conclusion 8

References 9

Contents 11

A Mathematical Background 12

A.1 Peter-Weyl Theorem 12

A.2 Fourier and Inverse Fourier Transform 13

A.3 From Fourier Transform to the Regular Representation 14

A.4 Quotient Representation 15

B Nonlinear Layers 16

C Equivariance of the Nonlinear Layer 17

D Orthogonality Metrics 17

E Practical Design Choices 20

F Experimental settings 21

F.1 ModelNet10 21

F.2 NoduleMNIST3D 23

G Hyperparameters 26

A. Mathematical Background

A.1. Peter-Weyl Theorem

In the context of harmonic analysis, Fourier transform decomposes a function into its frequency components. The Peter-Weyl theorem can be considered as a generalization of the classical Fourier transform to the setting of compact topological groups. Similar to the Fourier transform, the Peter-Weyl theorem decomposes certain functions on a compact group G into a series of irreducible components.

Theorem 1: Peter-Weyl Theorem : Decomposition into orthogonal subspaces

Given a compact group G, the space of square-integrable functions on G, which is denoted by $L^2(G)$, can be expressed as a *direct sum* of mutually orthogonal subspaces V_π;

$$L^2(G) = \bigoplus_{\pi \in \hat{G}} \bigoplus_{i=1}^{m_j} V_\pi \tag{17}$$

where $\hat{G}$ denotes the set of all equivalence classes of finite dimensional irreducible unitary representations of G, and m_j is the multiplicities of the corresponding subspace. Each subspace V_π is isomorphic to the representation π, and invariant under left and right translation by $g \in G$.

Note that this theorem only applies to the compact groups.

Intuitively, groups can be interpreted as rotations acting on geometric objects. In the context of **compact groups** [3], the matrix coefficients of their representations can be interpreted as square-integrable functions defined on the group. This perspective allows us to specifically redefine the decomposition of a unitary representation in terms of the group's representations.

Definition 1: Decomposition into irreducible representations

Given $\rho : G \to GL(V)$ is a unitary representation of a compact group G over a field with characteristic zero, ρ can be expressed as a direct sum of mutually orthogonal **irreducible representations** :

$$\rho(g) = Q^T \left(\bigoplus_{i \in I} \psi_i(g) \right) Q \tag{18}$$

where $\psi_i \in \hat{G}$ is an irrep, $\widehat{G}$ is the set of irreps of the group G, I is an index set ranging over $\hat{G}$, and $Q \in GL(V)$ is the change of basis.

Note that Peter-Weyl theorem ensures that each irrep acts on an invariant subspace of V.

[3] A compact group is a group that is also a compact topological space, meaning it is both closed and bounded. Compact groups can be infinite in terms of the number of elements, but have a finite size from a topological perspective. $SO(3)$ is an example of compact groups.

Theorem 2: Peter-Weyl : Orthonormal Basis

Let G be a compact group, $\rho : G \to GL(V)$ a unitary representation, and $L^2(G)$ vector space of square-integrable functions on G. Unitary representation ρ can be decomposed into irreps, ψ, as shown in Def. 1. Peter-Weyl Theorem guarantees that there are distinct linear subspaces of $L^2(G)$, each corresponding to an irreducible representation that transforms accordingly.

The matrix coefficients of the irrep ψ, denoted by $\psi_{ij}(g)$, form an orthonormal basis for each subspace V_ψ, and they satisfy the following orthonormality relations.

For infinite groups:

$$\int_G \overline{\psi'_{ij}(g)}\psi_{kl}(g)\, dg = \frac{1}{d_\psi}\delta_{\psi\psi'}\delta_{ik}\delta_{jl},$$

and for finite groups:

$$\frac{1}{|G|}\sum_{g\in G}\overline{\psi'_{ij}(g)}\psi_{kl}(g) = \frac{1}{d_\psi}\delta_{\psi\psi'}\delta_{ik}\delta_{jl}$$

where d_ψ is the dimension of the irrep ψ.

If the vector space V is a complex space, the orthonormality condition implies that the matrix coefficients form a complete orthonormal basis $\left\{\sqrt{d_\psi}\psi_{ij}(g) : G \to \mathbb{C} \mid \psi \in \widehat{G}, 1 \le i, j \le d_\psi\right\}$ for complex square integrable functions in $L^2(G)$.

In the case where the vector space is real $V = \mathbb{R}^n$, conjugated representations $\overline{\psi^l_{ij}(g)}$ is replaced by ψ^l_{ij}. From now on, we will assume vector spaces are of real type unless specified otherwise. It is important to note that for real irreps, some coefficients may be redundant. However, since we are exclusively working with $SO(3)$ in this study, this does not apply. For more details on this setting, please refer to Cesa et al. (2022).

Overall, the Peter-Weyl theorem provides a method for parameterizing functions over a group, which is especially useful for infinite groups.

A.2. Fourier and Inverse Fourier Transform

In the context of the Peter-Weyl theorem (Thm. 2), any square-integrable function on a compact group G, denoted $f : G \to \mathbb{R}$, can be expanded as a series using the matrix coefficients of the group's irreducible representations.

$$f(g) = \sum_{l=0}^{L}\sum_{\psi^l\in\hat{G}}\sum_{m,n<d_\psi} w_{lmn}\cdot\sqrt{d_{\psi^l}}\ \psi^l_{mn}(g) \tag{19}$$

where l is the degree of the spherical harmonics, $\hat{G}$ is the set of all irreducible representations of G, d_ψ is the dimension of irrep $\psi : G \to \mathbb{R}$ and the $\psi^l_{mn}(g)$ are the matrix coefficients of ψ. Coefficients w_{lmn} parameterize the function f on this basis, while $\sqrt{d_\psi}$ ensures that the basis is normalized.

This expansion is analogous to a Fourier series, where sines and cosines are replaced by the matrix coefficients of irreps. In this setting, coefficients w_{lmn} serve as the Fourier coefficients, which can be formulated as

$$w_{lmn} = \int_G f(g)\psi^l_{mn}(g)dg.$$

where dg is the normalized Haar measure on the group G. Haar measure can briefly be defined as the prior probabilities for compact groups of transformations, and it allows for the integration of functions over the group.
The projection onto the full set of entries of irrep ψ can be reformulated and referred to as the **Fourier transform**, as in Def 2.

Definition 2: Fourier Transform

For the compact group G, let $f : G \to \mathbb{R}$ be the square-integrable functions on G, and $\psi : G \to \mathbb{R}$ be the irreps of G. **Fourier transform** over the function f is formulated as follows for

$$\text{infinite groups:} \quad \hat{f}(\psi) = \int_G f(g)\sqrt{d_\psi}\psi(g)dg \quad \in \mathbb{R}^{d_\psi \times d_\psi} \tag{20}$$

$$\text{finite groups:} \quad \hat{f}(\psi) = \frac{1}{|G|}\sum_{g \in G} f(g)\sqrt{d_\psi}\psi(g) \quad \in \mathbb{R}^{d_\psi \times d_\psi} \tag{21}$$

where $\hat{f}(\psi)$ is the Fourier coefficients of the corresponding irrep ψ and $\sqrt{d_\psi}$ is the dimension of the irrep ψ.

It is important to note that the Fourier transform exhibits the equivariance property with respect to the action of a group element $g \in G$ on a function $f : G \to \mathbb{R}$:

$$\widehat{g \cdot f}(\psi) = \psi(g)\hat{f}(\psi)$$

for any irreducible representation (irrep) ψ.

Regarding the expansion of the square-integrable functions on G, the function in Eq. (19) can be redefined in a more compact form with the help of trace operation, which is defined as:

$$Tr(A^T B) = \sum_{m,n<d} A_{mn}B_{mn} \in \mathbb{R} \tag{22}$$

where $A, B \in \mathbb{R}^{d\times d}$. Since coefficients w^l_{mn} in Eq. (19) serves as Fourier coefficients, $\hat{f}(\psi^l) \in \mathbb{R}^{d_\psi \times d_\psi}$ can be defined as the matrix containing the coefficients $w^l_{mn} \in \mathbb{R}$. Given that, the **Inverse Fourier Transform** is defined.

Definition 3: Inverse Fourier Transform

Given group G and a function on G, $f : G \to \mathbb{R}$, **Inverse Fourier Transform** can be defined as

$$f(g) = \sum_{\psi \in \hat{G}} \sqrt{d_\psi}\; Tr(\psi(g)^T \hat{f}(\psi)) \tag{23}$$

where $\hat{G}$ is the set of irreps of the group G, $\hat{f}(\psi)$ is the Fourier coefficients of irrep ψ, and d_ψ is the dimension of the corresponding irrep.

For the infinite groups, there exists infinite irreducible representations. We can still parameterize a function using the same expression for the Inverse Fourier Transform (Def. 3), but only by taking the finite subset of the irreps, which is $\hat{G} \subset \tilde{G}$, into account in the computation. This finite subset is also referred to as *bandlimited* representations.

A.3. From Fourier Transform to the Regular Representation

The Peter-Weyl theorem and Fourier transform has a strong connection in terms of breaking down functions into their irreducible components. In this section, this connection will be shown in a more formal sense, focusing on the decomposition of the regular representation.

The inverse Fourier transform (Eq. (23)) is defined in the previous section. Transpose of an inner product of two matrices can be defined in terms of vectorization of the matrices, as in $Tr(A^T B) = \text{vec}(A)^T\text{vec}(B)$. Given this, the trace in the inverse Fourier transform (Def. 3) can be rewritten as

$$Tr(\psi(g_i)^T \hat{f}(\psi)) = \text{vec}(\psi(g_i))^T\text{vec}(\hat{f}(\psi)). \tag{24}$$

This makes the inverse Fourier transform (Eq. (23))

$$f(g) = \sum_{\psi \in \hat{G}} \sqrt{d_\psi} \text{vec}(\psi(g))^T \text{vec}(\hat{f}(\psi) \tag{25}$$

The summation can be converted into the matrix notation with the help of direct sum. Fourier coefficients $\hat{f}(\psi) \in \mathbb{R}^{d_\psi \times d_\psi}$, by stacking the columns of each $\hat{f}(\psi)$:

$$\mathbf{f} = \bigoplus_{\psi \in \hat{G}} \text{vec}(\hat{f}(\psi)) = \begin{bmatrix} \text{vec}(\hat{f}(\psi_1)) \\ \text{vec}(\hat{f}(\psi_2)) \\ \vdots \end{bmatrix}. \tag{26}$$

As stated before, Fourier coefficients has the equivariance property which can be expressed as $\widehat{g \cdot f}(\psi) = \psi(g)\widehat{f}(\psi)$. Given Eq. (26) and the equivariance property of the Fourier coefficients, vectorized function $[g \cdot f]$ can be rewritten as:

$$\bigoplus_{\psi \in \hat{G}} \text{vec}(\psi(g)\hat{f}(\psi)) = \bigoplus_{\psi \in \hat{G}} \left(\bigoplus^{d_\psi} \psi(g) \right) \text{vec}(\hat{f}(\psi)) \tag{27}$$

$$= \bigoplus_{\psi \in \hat{G}} \left(\bigoplus^{d_\psi} \psi(g) \right) \mathbf{f}. \tag{28}$$

In simpler terms, the group G acts on the vector **f** with the following representation:

$$\rho(g) = \bigoplus_{\psi \in \hat{G}} \bigoplus^{d_\psi} \psi(g). \tag{29}$$

This means that $\rho(g)\mathbf{f}$ is the vector containing Fourier coefficients of the function $[g \cdot f]$. Representation ρ acts on a space of functions over the group G. If G is finite, this representation is **isomorphic** (equivalent) to the *regular representation* defined in Sec. 3. Given Q is the matrix that serves as a change of basis that performs Fourier transform, Q^{-1} is the matrix that performs the inverse Fourier transform. This can be formulated as:

$$\rho_{reg}(g) = Q^{-1} \left(\bigoplus_{\psi \in \hat{G}} \bigoplus^{d_\psi} \psi(g) \right) Q. \tag{30}$$

where d_ψ is the multiplicities of the corresponding irrep ψ, and also its dimension.

A.4. Quotient Representation

In Section 3.1, we investigated the inverse Fourier transform on groups, and defined the final equation as:

$$f(g_i) = \hat{\delta}^T \rho(g_i)^T \hat{f} \tag{31}$$

where $\rho(g_i) = \bigoplus_{\psi \in \hat{G}} \bigoplus^{d_\psi} \psi(g)$ for regular representation and $\rho(g_i) = \bigoplus_{\psi \in \hat{G}} \psi(g)$ for quotient representation of $Q = SO(3)/SO(2)$. Recall that $\hat{\delta} = \bigoplus_\psi \bigoplus^{q_\psi} \sqrt{d_\psi} vec(P_\psi)$ is interpreted as a vector with Fourier coefficients of a Dirac delta function centred at identity, and $\hat{f} = \bigoplus_\psi vec(\hat{f}(\psi))$ is a vector of Fourier coefficients of the irreps ψ.

For the quotient representation ρ with the quotient space $Q = SO(3)/SO(2)$, multiplication in $\hat{\delta}^T \rho(g_i)^T$ in Eq. (31) results in only the mid-column of each irrep ψ in $\rho(g_i)$. This can be illustrated as

$$\rho(g_i)\hat{\delta} = \underbrace{\left(\begin{array}{c|c|c|c} \psi_0(g_i) & & & \\ \hline & \psi_1(g_i) & & \\ \hline & & \psi_2(g_i) & \\ \hline & & & \ddots \end{array}\right)}_{\rho(g_i)=\bigoplus_{\psi\in\hat{G}}\psi(g_i)} \cdot \underbrace{\left(\begin{array}{c} \sqrt{d_{\psi_0}} \\ - \\ \sqrt{d_{\psi_1}} \\ - \\ \hline - \\ - \\ \sqrt{d_{\psi_2}} \\ - \\ - \\ \hline \vdots \end{array}\right)}_{\hat{\delta}} \tag{32}$$

where l in ψ_l corresponds to the degree of spherical harmonics. Note that the blocks in $\hat{\delta}$ corresponds to the non-scaled matrices $\sqrt{d_\psi} vec(P_\psi)$ in Eq. (10).

Spherical harmonics are functions defined on the surface of a sphere that form an orthogonal basis for square-integrable functions on the sphere. When used in equivariant kernel design, they extend the feature space onto this spherical surface, allowing the kernel to capture more complex patterns. Spherical harmonics can also be described as functions $\tilde{Y}^l_m : SO(3) \to \mathbb{R}$ on $SO(3)$. and are considered as a special case of the Wigner D-matrices[4]. Specifically, the $n = 0$ order within these matrices corresponds to spherical harmonics, suggesting that spherical harmonics can be represented as the $n = 0$ column of a Wigner-D matrix, formulated as

$$Y^l_m(\theta, \phi) = \frac{1}{\sqrt{2l+1}} D^l_{m0}(\theta, \phi, \gamma). \tag{33}$$

This corresponds to P_ψ selecting the columns of ψ which are H invariant in Eq. (10). The column, D^l_{m0}, consists of functions that are invariant to rotations around a particular axis, indicating that spherical harmonics remain invariant under such rotations. Consequently, focusing on the middle column of the Wigner-D matrices allows us to represent rotation-equivariant features in a more computationally efficient manner, as it requires fewer channels to be stored, while also maintaining efficiency (Bekkers et al., 2023). Eq. (32) illustrates how the middle column of the Wigner-D matrices are extracted in inverse Fourier transform.

B. Nonlinear Layers

In equivariant networks, nonlinear layers must also satisfy the equivariance constraint. The nonlinearities discussed in this section operate on each feature vector $f(x) \in \mathbb{R}^{c_{\text{in}}}$ for all $x \in \mathbb{R}^n$.

Norm Nonlinearity Point-wise nonlinearities acting on the norm of each field preserves the rotational equivariance, as described in Worrall et al. (2016). Norm nonlinearity can be formulated as:

$$f(x) \to \sigma\left(\|f(\mathbf{x})\|_2\right) \frac{f(\mathbf{x})}{\|f(\mathbf{x})\|_2} \tag{34}$$

where σ represents the point-wise nonlinearity. In the case of σ being ReLU, a learnable bias $b \in \mathbb{R}^+$ can be introduced, which makes $\sigma\left(\|f(\mathbf{x})\|_2\right) = ReLU(\|f(\mathbf{x})\|_2 - b)$.

Gated Nonlinearity Weiler et al. (2018a) introduce gated nonlinearity, which can be considered as a special case of the norm nonlinearity. Gated nonlinearities act on a feature vector $f(\mathbf{x})$ by scaling its norm by a gating scalar that is computed via a sigmoid nonlinearity $\frac{1}{1+e^{-s(\mathbf{x})}}$, where s is a scalar feature field. In this case, $\sigma\left(\|f(\mathbf{x})\|_2\right)$ in Eq. 34 can be reformulated as $\|f(\mathbf{x})\|_2 \frac{1}{1+e^{-s(\mathbf{x})}}$.

Tensor Product Nonlinearity Feature vectors $f_1(\mathbf{x}) \in \mathbb{R}^{c_1}$ and $f_2(\mathbf{x}) \in \mathbb{R}^{c_2}$ of arbitrary types ρ_1 and ρ_2 can be combined to a tensor product feature $(f_1 \otimes f_2)(\mathbf{x}) \in \mathbb{R}^{c_1 c_2}$. Product output transforms under the representation $\rho_1 \otimes \rho_2 : G \to GL(\mathbb{R}^{c_1 c_2})$. Tensor product operation is nonlinear and equivariant, therefore it can be integrated into the equivariant network without the need for any further nonlinearities. More details can be found in Kondor et al. (2018); Weiler et al. (2018a).

[4]Wigner D-matrices are unitary matrices representing irreducible representations of the groups $SU(2)$ and $SO(3)$

C. Equivariance of the Nonlinear Layer

In equivariant network architectures, the assumption is that the previous MLP and convolution layers are all equivariant, satisfying $\hat{f}(g.x) = \rho(g)\hat{f}(x)$. In our adaptive approach, the sampling matrix is generated through equivariant layers to ensure full equivariance, meaning that it satisfies the constraint

$$A(g.x) = A(x)\rho(g)^T. \tag{35}$$

Given the equivariance of the previous layers and the sampling matrix A, equivariance of the activation layer can be shown as:

$$\hat{f}(g.x) = \frac{1}{N}A(g.x)^T\sigma(A(g.x)\hat{f}(g.x)) \tag{36}$$

$$\hat{f}(g.x) = \frac{1}{N}(A(x)\rho(g)^T)^T\sigma(A(x)\rho(g)^T\rho(g)\hat{f}(x)) \tag{37}$$

$$\hat{f}(g.x) = \rho(g)\frac{1}{N}A(x)^T\sigma(A(x)\hat{f}(x)) \tag{38}$$

$$\hat{f}(g.x) = \rho(g)\hat{f}(x) \tag{39}$$

D. Orthogonality Metrics

In Section 3.3, we discussed the orthogonality of the sampling matrix, meaning that it satisfies the condition $A^TA = AA^T = NI$, where N is the number of group samples. In this section, we will compute the deviation of the sampling matrices from orthogonality using two metrics; ϵ_1 in Eq. (40) and Eq. (48). Note that the representations in this section are assumed to be **quotient representations**. Towards the end of this section, we briefly discuss how the metrics differ for regular representations. The first metric ϵ_1 is defined as

$$\epsilon_1 = \frac{1}{N}\sum_{i,j<N}|AA^T - I_N| \tag{40}$$

where N is the number of group samples, and I_N is the identity matrix of size N. To simplify, AA^T can be visualized as

$$AA^T = \underbrace{\left(\begin{array}{c|c|c|c} \underset{\blacksquare}{\psi_0(g_0)\hat{\delta}_0} & \underset{\blacksquare\blacksquare\blacksquare}{\psi_1(g_0)\hat{\delta}_1} & \cdots & \underset{\blacksquare\blacksquare\cdots\blacksquare}{\psi_L(g_0)\hat{\delta}_L} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline \underset{\blacksquare}{\psi_0(g_N)\hat{\delta}_0} & \underset{\blacksquare\blacksquare\blacksquare}{\psi_1(g_N)\hat{\delta}_1} & \cdots & \underset{\blacksquare\blacksquare\cdots\blacksquare}{\psi_L(g_N)\hat{\delta}_L} \end{array}\right)}_{A\ \in\ \mathbb{R}^{N\times F}} \cdot \underbrace{\left(\begin{array}{c|c|c} \blacksquare\,[\psi_0(g_0)\hat{\delta}_0]^T & \cdots & \blacksquare\,[\psi_0(g_N)\hat{\delta}_0]^T \\ \hline \begin{smallmatrix}\blacksquare\\\blacksquare\\\blacksquare\end{smallmatrix}\,[\psi_1(g_0)\hat{\delta}_1]^T & \cdots & \begin{smallmatrix}\blacksquare\\\blacksquare\\\blacksquare\end{smallmatrix}\,[\psi_1(g_N)\hat{\delta}_1]^T \\ \hline \vdots & \ddots & \vdots \\ \hline \begin{smallmatrix}\blacksquare\\\blacksquare\\\vdots\\\blacksquare\end{smallmatrix}\,[\psi_L(g_0)\hat{\delta}_L]^T & \cdots & \begin{smallmatrix}\blacksquare\\\blacksquare\\\vdots\\\blacksquare\end{smallmatrix}\,[\psi_L(g_N)\hat{\delta}_L]^T \end{array}\right)}_{A^T\ \in\ \mathbb{R}^{F\times N}} \tag{41}$$

$$= \underbrace{\sum_{\psi\in\hat{G}}\left(\begin{array}{c|c|c} \underset{\blacksquare}{\psi(g_0)\hat{\delta}\cdot[\psi(g_0)\hat{\delta}]^T} & \cdots & \underset{\blacksquare}{\psi(g_0)\hat{\delta}\cdot[\psi(g_N)\hat{\delta}]^T} \\ \hline \vdots & \ddots & \vdots \\ \hline \underset{\blacksquare}{\psi(g_N)\hat{\delta}\cdot[\psi(g_0)\hat{\delta}]^T} & \cdots & \underset{\blacksquare}{\psi(g_N)\hat{\delta}\cdot[\psi(g_N)\hat{\delta}]^T} \end{array}\right)}_{AA^T\ \in\ \mathbb{R}^{N\times N}} \tag{42}$$

where F is the size of the representation $\rho(g_i)\hat{\delta}$, which can be computed as $F = \sum_{l=0}^{L} 2l+1$ for $SO(3)$, with degree up to L. The matrices are formed in blocks to emphasize the multiplication between the corresponding representations. Each square $\blacksquare$ in the matrix corresponds to a single matrix coefficient.

Given the orthogonality of the compact group representations, $\rho(g^{-1}) = \rho(g)^{-1} = \rho(g)^T$, and the Peter-Weyl theorem (Thm. 2), it is expected that the matrix product AA^T yield an identity matrix. The ϵ_1 quantifies the deviation of the representations from the perfect orthogonality.

Moreover, A^TA can be illustrated as

$$A^TA = \underbrace{\left(\begin{array}{c|c|c} \blacksquare\,[\psi_0(g_0)\hat{\delta}_0]^T & \cdots & \blacksquare\,[\psi_0(g_N)\hat{\delta}_0]^T \\ \hline \vdots\,[\psi_1(g_0)\hat{\delta}_1]^T & \cdots & \vdots\,[\psi_1(g_N)\hat{\delta}_1]^T \\ \hline \vdots & \ddots & \vdots \\ \hline \vdots\,[\psi_L(g_0)\hat{\delta}_L]^T & \cdots & \vdots\,[\psi_L(g_N)\hat{\delta}_L]^T \end{array}\right)}_{A^T \;\in\; \mathbb{R}^{F\times N}} \cdot \underbrace{\left(\begin{array}{c|c|c|c} \underset{\blacksquare}{\psi_0(g_0)\hat{\delta}_0} & \underset{\blacksquare\blacksquare\blacksquare}{\psi_1(g_0)\hat{\delta}_1} & \cdots & \underset{\blacksquare\blacksquare\cdots\blacksquare}{\psi_L(g_0)\hat{\delta}_L} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline \underset{\blacksquare}{\psi_0(g_N)\hat{\delta}_0} & \underset{\blacksquare\blacksquare\blacksquare}{\psi_1(g_N)\hat{\delta}_1} & \cdots & \underset{\blacksquare\blacksquare\cdots\blacksquare}{\psi_L(g_N)\hat{\delta}_L} \end{array}\right)}_{A \;\in\; \mathbb{R}^{N\times F}} \tag{43}$$

$$= \underbrace{\sum_{g\in G} \left(\begin{array}{c|c|c} \blacksquare\;\; [\psi_0(g)\hat{\delta}_0]^T\psi_0(g)\hat{\delta}_0 & \cdots & \cdots \\ \hline \vdots & \ddots\, [\psi_1(g)\hat{\delta}_1]^T\psi_1(g)\hat{\delta}_1 & \cdots \\ \hline \vdots\;\; [\psi_0(g)\hat{\delta}_0]^T\psi_L(g)\hat{\delta}_L & \ddots & \ddots \end{array}\right)}_{A^TA \;\in\; \mathbb{R}^{F\times F}} \tag{44}$$

Recall the orthogonality property of irreps in Peter-Weyl theorem (Thm. 2). The orthonormality condition for the $\psi^l_{mn}(g)$ can be expressed as an integral over the group as in

$$\int_{SO(3)} \psi^l_{mn}(g)\psi^{l'}_{m'n'}(g)\,d\mu = \delta_{ll'}\delta_{mm'}\delta_{nn'}\frac{1}{2l+1} \tag{45}$$

and in the discretized form

$$\sum_{g\in G} \psi^l_{mn}(g)\psi^{l'}_{m'n'}(g) = \frac{|G|}{2l+1}\delta_{ll'}\delta_{mm'}\delta_{nn'} \tag{46}$$

where ψ^l is an irrep of $SO(3)$ with degree l and with m, n matrix indices. $|G|$ is a finite number of group samples from $SO(3)$.

Given Eq. (46), each square (▪) in diagonal blocks $\sum_{g\in G}[\psi_l(g)\hat{\delta}_l]^T\psi_l(g)\hat{\delta}_l$ (Eq. (44)) corresponds to $|G|$. Note that $2l+1$ in Eq. (46) is cancelled with $\hat{\delta}^T\hat{\delta}$ in the blocks (see $\hat{\delta}$ in Eq. (32)). This makes the A^TA

$$A^TA = \left(\begin{array}{c|c|c} |G| & & \\ \hline & |G| & \\ \hline & & \ddots \end{array}\right) \quad \in \mathbb{R}^{F\times F}. \tag{47}$$

To obtain the identity matrix, we can divide A^TA by $|G|$.

$$\epsilon_2 = \frac{1}{F}\sum_{i,j<d_\rho}\left|\frac{1}{|G|}A^TA - I_F\right| \tag{48}$$

The value ϵ_2 helps us understand how closely the learned group representations align with the principles of the Peter-Weyl theorem.

Normalization The concepts discussed so far applies if $\hat{\delta}$ is not normalized in any way. Models in Sec. 7 employ normalization over the vector $\hat{\delta}$ as in

$$\hat{\delta} \mapsto \tilde{\delta} := \frac{\hat{\delta}}{\|\hat{\delta}\|}. \tag{49}$$

Normalization over the entire vector has no effect on ϵ_1, since in AA^T (Eq. (42)), each matrix coefficient has the entire $\hat{\delta}$. However, in $A^T A$ (Eq. (44)), each matrix coefficient is associated with the respective segment of the kernel corresponding to the degree l. Therefore, in the computation of the ϵ_2 (Eq. (48)), each matrix coefficient can be multiplied by the norm of the kernel to assess the orthogonality more accurately. As an alternative, different normalization methods can be applied.

Regular Representation Regular representation is defined as

$$\rho_{reg}(g) = Q^{-1}\left(\bigoplus_{\psi}\bigoplus^{d_\psi}\psi(g)\right)Q.$$

where Q is the change of basis, d_ψ is the size and the multiplicities of the corresponding irrep ψ. This means that, in the irreps decomposition, each irrep ψ is repeated as many times as its size. For regular representations, $A \in \mathbb{R}^{N\times F}$ matrix takes a similar form, with the only difference being the repetition of the irreps, and the size $F = \sum_{l=0}^{L}(2l+1)^2$.

E. Practical Design Choices

In Section 5, we cover the implementation details, and here, we discuss various possible designs.

The implementation for voxel data is relatively straightforward by introducing convolutional layers for the generation and downsampling of the sampling matrix. For the point cloud processing, we have tried various implementation for the sampling branch by prioritizing the expressiveness and the computational efficiency. In both architectures, the sampling matrix A can be generated by processing the model input directly or by processing the first intermediate feature tensor, which corresponds to the output of the first pair of convolution and normalization layer. It is important to note that directly using the original input in an equivariant MLP may introduce specific challenges. The linear map W, in an equivariant MLP, is parameterized based on the Schur's lemma, which states that the irrep intertwiners[5] are zero for non-isomorphic[6] irreps. In other words, if ρ_1 and ρ_2 are two non-isomorphic irreps, then any linear map W such that $W\rho_1(g) = \rho_2(g)W \quad \forall g \in G$ must be the zero map. This property ensures that the space of intertwiners between different irreps is trivial unless the irreps are isomorphic. Model input is initially interpreted as feature space with representations of frequency zero. This means that W will be parametrized in a way that the input won't be mapped to a feature space with representations of higher frequencies, which results in information loss and less expressive intermediate features. This issue can be mitigated by the sampling branch processing the first intermediate features instead of model input, since the first feature space is generated by a convolutional layer which doesn't have similar restrictions in the parametrization.

Both point clouds and voxel data are downsampled through the convolutional layers in the main branch of the architecture. As detailed in (Sec. 5), the model generates a unique sampling matrix for each point in the point clouds and each pixel in the voxel data. To ensure compatibility between the feature map and the sampling matrices, we apply subsampling to the sampling matrix. We can only employ convolutional layers in the processing of voxel data, however for point clouds, the simple approach would be to use indexing, where only the points that are still in the feature space are kept for further processing. This might result with the information loss since while the points are processed by message passing in the downsampling in the main architecture, the sampling matrix won't go through any further processing. Therefore, the expressiveness of the sampling matrices might decrease, particularly in the case where it is generated by an MLP layer which discards the neighbouring information.

We have implemented multiple architectures given the points discussed. In Sec. 7, we only provide the results from the models that perform the best.

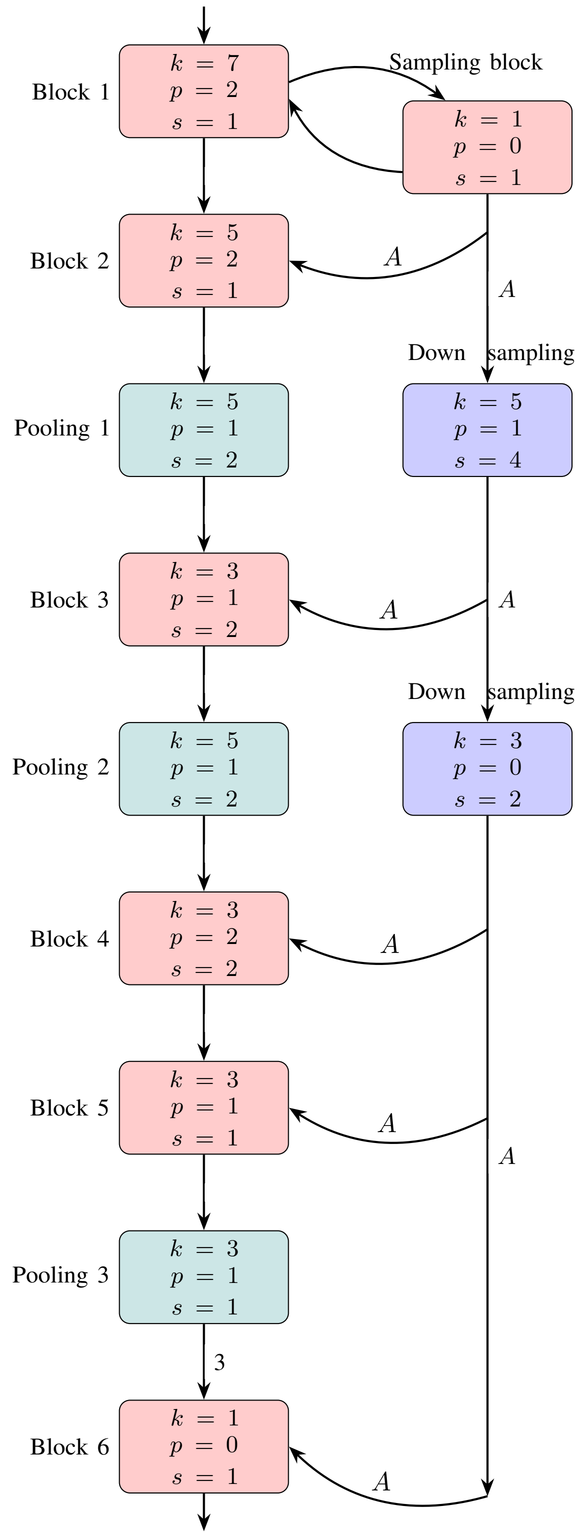

Figure 7. Architecture for voxel processing. The branch on the left represents the main branch that process the voxel data, while the branch on the right can be referred to as the sampling branch. The sampling block receives the first intermediate feature from the main branch and returns the sampling matrix A to the first nonlinear layer in the first convolutional block. Each convolutional block comprises convolutional layer, batch normalization and activation layer.

[5]An intertwiner is a linear map between two group representations that commutes with the group action, preserving the structure of the representations.

[6]In group theory, an isomorphism is a bijective mapping between two groups that preserves the group structure. It respects the group operations and establishes a one-to-one correspondence between the elements of the groups.

F. Experimental settings

Here, we will outline the experimental settings and analyze the model results on two separate datasets. Additionally, we will present the findings in terms of model equivariance, runtime, and the orthogonality of the sampling matrices.

Equivariance Due to the final invariant layer in the network, the entire network is expected to be invariant to rotations. To investigate the impact of different nonlinear layers and the number of group samples on the degree of equivariance in each model, we examined the relative equivariance error, which is defined as:

$$\epsilon_g(f) = \frac{\|f(x) - f(T(x))\|}{\max\{\|f(x)\|, \|f(T(x))\|\}} \tag{50}$$

where f is the network, T is the transformation of interest, which is $SO(3)$ rotations in this work, and x is the model input. It is important to note that this formulation computes the invariance error, since the numerator in the formulation corresponds to invariance. However, we will refer to it as the *equivariance error*, to emphasize the degree of equivariance of the model before the final invariant layer empirically.

Orthogonality Recall from Sec. 3.3 that our approach assumes the sampling matrix is orthogonal, satisfying $AA^T = A^TA = NI$, where N is the number of group samples. In this section, we measure the deviation from orthogonality using two metrics introduced in Sec. D. To compute the final ϵ_1 and ϵ_2 values (Eq. (51)), we calculate these metrics for each data sample and average the results over the samples in the test set.

$$\begin{aligned} \epsilon_1 &= \frac{1}{N}\sum_{i,j<N}(AA^T - I_N) \\ \epsilon_2 &= \frac{1}{F}\sum_{i,j<d_\rho}\left(\frac{1}{|G|}(A^TA) - I_F\right) \end{aligned} \tag{51}$$

In the context of matrix orthogonality, for a $p \times q$ matrix, if $p < q$, all rows can potentially be orthogonal and linearly independent. Conversely, if $p > q$, not all rows can be orthogonal or independent due to dimensional constraints. In our specific case, the sampling matrix A has the shape $N \times F$, where N represents the number of group samples and F is the size of the representations. This means that not all rows of A can be orthogonal to each other when the number of group samples exceeds the size of the representations.

F.1. ModelNet10

To evaluate the model's robustness to SO(3) rotations, we generated a new set of training and test set. The new training set is generated by expanding the original ModelNet10 training set to three times its original size by incorporating random $SO(3)$ rotations into the point clouds. This means that each model is trained with potentially different datasets containing the same point clouds but in different orientations. The test set is also expanded to three times its original size by introducing $SO(3)$ rotations to the point clouds in the original dataset. However, the rotations for the test set are precomputed, ensuring consistency as all the models are evaluated on the same set. Training is conducted three times for each model, using three different seeds consistently to initialize the models. Each model is trained for 70 epochs and the highest test accuracy is reported in Fig 4. All models discussed in this section uses Fourier-based nonlinearity with quotient representation.

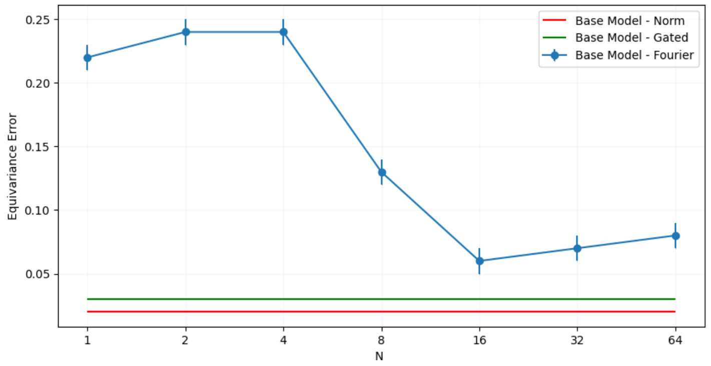

Figure 8. Equivariance error based on the number of group samples.

Equivariance Equivariance error is computed by rotating a single data sample by every octahedron rotations and taking the average of the errors with respect to each rotation. As seen in the Fig. 8, the equivariance improves as the number of samples increases. In our approach, as described in Sec. C, the network is designed to be fully equivariant. Consequently, the equivariance error is zero, which is consistent with our results.

Runtimes The inference runtimes were obtained by averaging the results of eleven epochs, excluding the highest and lowest values among those runs. We also excluded the first ten batches for the warm-up.

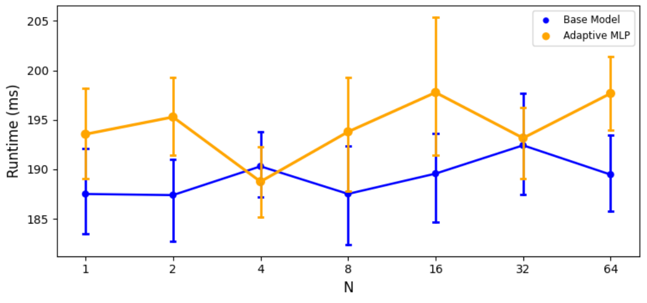

Figure 9. Inference runtime with respect to the number of group samples for Base model (blue) and Adaptive MLP (yellow).

The Adaptive MLP generally exhibits longer inference runtimes compared to the Base model, as reported in Fig. 9. The only exception is when the model is trained with four samples. The increase in runtime for the adaptive approach can be attributed to the higher number of parameters (see Table 1) and the additional computational steps involved in generating the sampling matrices and performing downsampling on them. Alternatively, using just-in-time (JIT) compilation could improve the results, but we did not experiment with that.

Orthogonality In the conventional approach, there is a single sampling matrix generated and used throughout the network. In our approach, Adaptive MLP generates a separate sampling matrix for each point in the point cloud, which enables to capture local symmetries. In this work, we use quotient representations for $Q = S^2 \cong SO(3)/SO(2)$ up to degree 3, where F becomes equal to $\sum_{l=0}^{3} 2l + 1 = 16$. This means that we can ideally achieve perfect orthogonality in terms of row independence, making ϵ_1 close to zero as long as $N < 16$. For ϵ_2, the criteria for orthogonality is almost the opposite, meaning that we can obtain a value close to zero for ϵ_2 as long as $N > 16$. This is indeed the case within the base models, as also shown in the first row of Fig. 10. Regarding the ϵ_1, the values start to diverge around 8 samples, which is slightly earlier than expected. A similar pattern is observed for ϵ_2. Notably, ϵ_2 values do not converge to zero, but to around 0.7. This is due to normalization over the $\hat{\delta}$ in Eq. (49).

In our Adaptive MLP approach, we observe that even though we don't enforce orthogonality to the matrices, the network itself learns to create orthogonal sampling matrices. This is evident in both ϵ_1 and ϵ_2, which exhibit trends similar to those in the Base model, as shown in Fig. 10. Here, Block 1, Block 2, and Block 3 correspond to the sampling branch, first downsampling, and second downsampling blocks, respectively, as depicted in Fig. 3. ϵ's are computed for the sampling matrices processed in each one of those blocks. Additionally, from the second row onwards, there is a noticeable increase in ϵ_1 and in ϵ_2 from Block 1 to Block 3, particularly when $N > 16$. This trend further suggests that the model is autonomously learning orthogonality, without the need for external guidance. In terms of normalization applied to the sampling matrix, we experimented different techniques. However, these techniques didn't improve the model performance in terms of classification accuracy. The most effective normalization was the one applied over the entire $\hat{\delta}$ in Eq. (49).

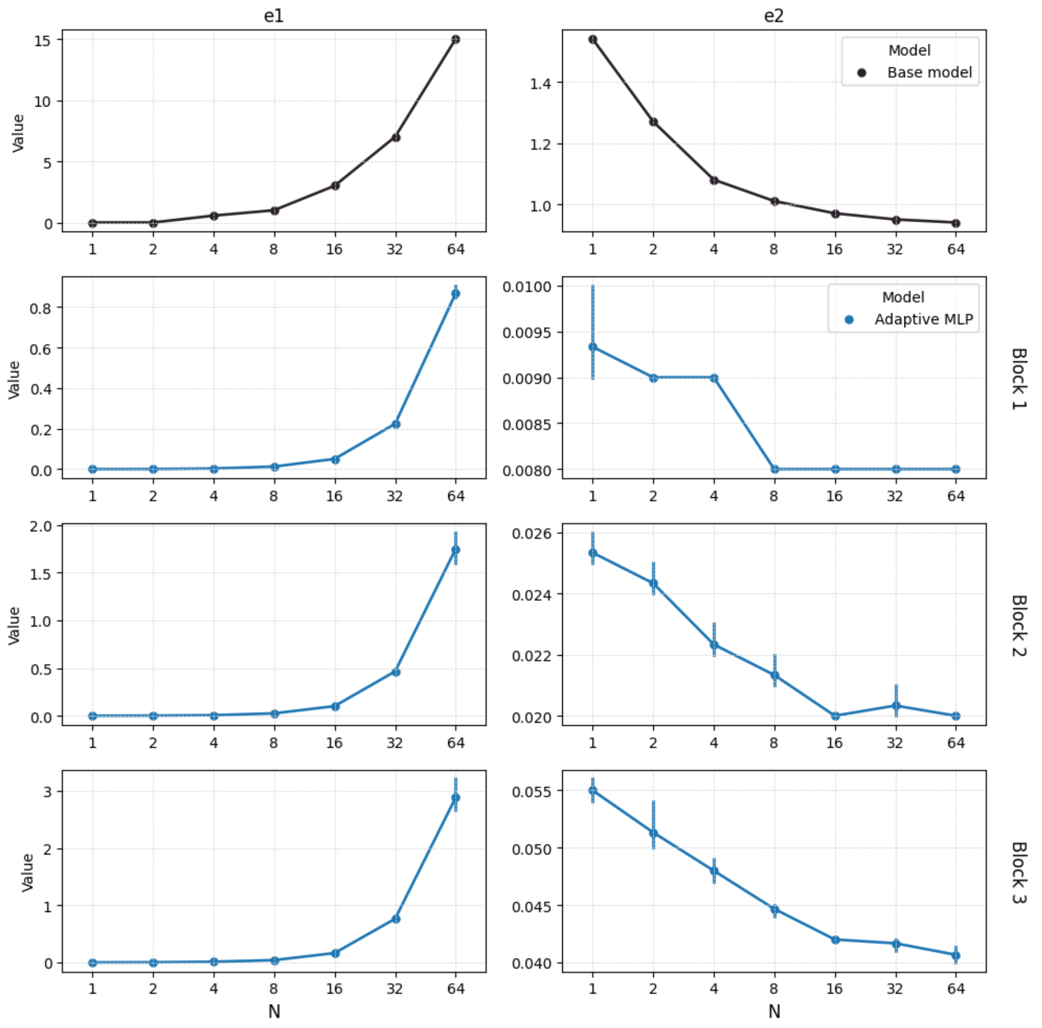

Figure 10. Orthogonality metrics for Base model and Adaptive MLP. The first row presents the values obtained from the Base model with respect to the number of group samples. Block 1, Block 2 and Block 3 refer to the sampling block, first downsampling block and the second sampling block, respectively, in the architecture (Fig. 3). Metrics are computed for the sampling matrices computed in each of those blocks, and presented on the second-to-last rows.

F.2. NoduleMNIST3D

MedMNIST3D (Yang et al., 2023) offers a collection of six biomedical image datasets, each containing 3D images standardized to 28x28x28 pixels and annotated with relevant classification label. In this work, the experiments are conducted on the MedMNIST3D Nodule dataset, which is a subset of the MedMNIST collection. This dataset is mainly designed for binary classification, focusing on the presence of the lung nodules in 3D CT scans. To evaluate the models, we generated rotated training, validation, and test sets. Rotating 3D voxels with random $SO(3)$ rotations introduces several issues such as interpolation artifacts due to misalignment with the original grid, and aliasing effects caused by the voxel grid's discrete nature. Furthermore, boundary effects at the grid edges can create artificial artifacts. To minimize the impact of those issues, we only perform cubic rotations, which is a subgroup of $SO(3)$, and we upsample each sample from 28 pixels to 29 pixels in each dimension.

The training dataset is generated by applying random cubic rotations to each sample from the original training set, to ensure variation in the training process. The validation set is expanded to three times the size of the original test set by applying precomputed rotations. For the test set, we rotate each sample in the original test set with every cubic rotation, resulting in a set of size twenty-four times the original. We also trained the each model for 100 epochs and choose the one with the best validation set accuracy for testing. All results reported here are based on this test set performance. Furthermore, our implementation uses Fourier based nonlinearity with ELU as the pointwise nonlinearity and with regular representations up to degree 3. All the hyperparameters used in the experiments are presented in Sec. G.

Additionally, in our experiments, we follow the approach, where the first intermediate features are used by the sampling branch to generate the sampling matrix. This approach is particularly effective since intermediate features, as opposed to the original input, already contain extracted features, and provide more information for the sampling branch. The full architecture of the model is illustrated in Fig. 7.

Equivariance Equivariance error is computed by applying all cubic rotations to a single Nodule data sample and then averaging the equivariance error over these operations. As shown in Fig. 11, while the base model is not fully equivariant to rotations, the degree of equivariance tends to align with the number of group samples used, especially in models trained with sixteen or more samples. In contrast, our approach consistently ensures full equivariance, regardless of the implementation choice or the number of group samples.

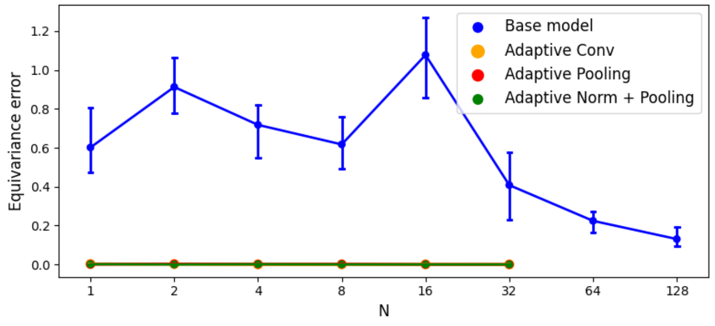

Figure 11. Equivariance with respect to the number of samples

Runtimes For consistency, each model was tested on the same test set over eleven epochs, and we calculated their average runtime, discarding the highest and lowest values from these runs. The results, depicted in Fig. 12, represent the average runtime across models initialized with different seeds.

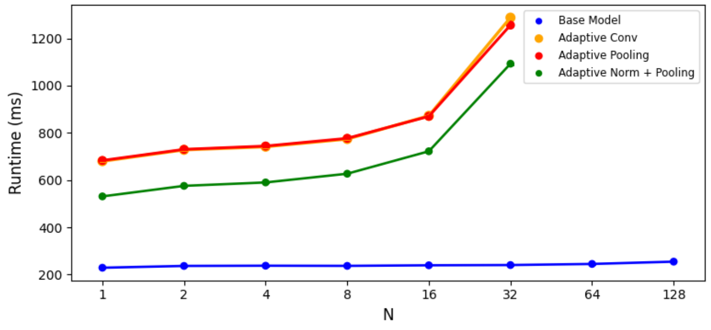

Figure 12. Inference runtime with respect to the number of group samples by model.

Fig. 12 reveals that our approach, incorporating extra layers into the architecture, typically results in an increased runtime compared to the base model. However, when the architecture's first nonlinearity is a norm nonlinearity, a shorter runtime is observed. This decrease in runtime can be attributed to the computationally less demanding nature of norm nonlinearities compared to the Fourier-based nonlinearities and the reduced number of parameters in the model, as detailed in Table 3.

Orthogonality Our approach generates a separate sampling matrix for each pixel in the voxel data. To minimize the computational complexity and also to obtain a more representative matrix, we first computed the average of the sampling matrices over the pixels for each sample and integrate the result into the nonlinear layer, to take a more global approach and also to reduce the computational cost to a certain degree. However, the final performance was quite low in terms of classification accuracy, therefore, we discard this implementation. In the computation of the final ϵ_1 (Eq. (40)) and ϵ_2 (Eq. (48)) for each model, we compute the metrics for each pixel in the data sample and average the results over the spatial dimensions, so that each data sample has a single ϵ_1 and ϵ_2 values. For the final results, we average each metric over the samples in the test set.

In these experiments, we use regular representations up to and including degree 3, where F becomes equal to $\sum_{l=0}^{3}(2l+1)^{2l+1} = 35$ (see Eq. (30)). This means that we can ideally achieve perfect orthogonality in terms of row independence, making ϵ_1 close to zero as long as $N < 35$, and making ϵ_2 close to zero as long as $N > 35$. This is indeed our observation, as also shown in the first row of Fig. 13. However, in the figure, see that the values are actually around 0.8, and not around zero. As previously noted in Sec. D, the normalization of the sampling matrix impacts the computation of ϵ_2. Given that the library by default normalizes across the entire vector of representations for each group sample, rather than on specific segments of the kernel representation, it is anticipated that ϵ_2 may deviate slightly from zero.

Similar to the results for ModelNet10, in our approach, the model demonstrates a tendency to approximate orthogonality in the sampling matrix, displaying a similar trend to the base model with respect to the number of group samples. The trend is more evident in ϵ_2, where adaptive models tends to have a narrower error margin than the base model. We also experimented with various normalization methods for the sampling matrix. However, these modifications did not yield improvements in model performance regarding test accuracy. In the models we present, normalization is applied across the entire representation, similar to the approach used in the base model.

Additionally, it is important to note that while the base model was trained up to 128 group samples, we restricted our adaptive models to a maximum of thirty-two group samples. This was due to GPU memory constraints in our server, as models with higher number of group samples exceeded the available memory capacity.

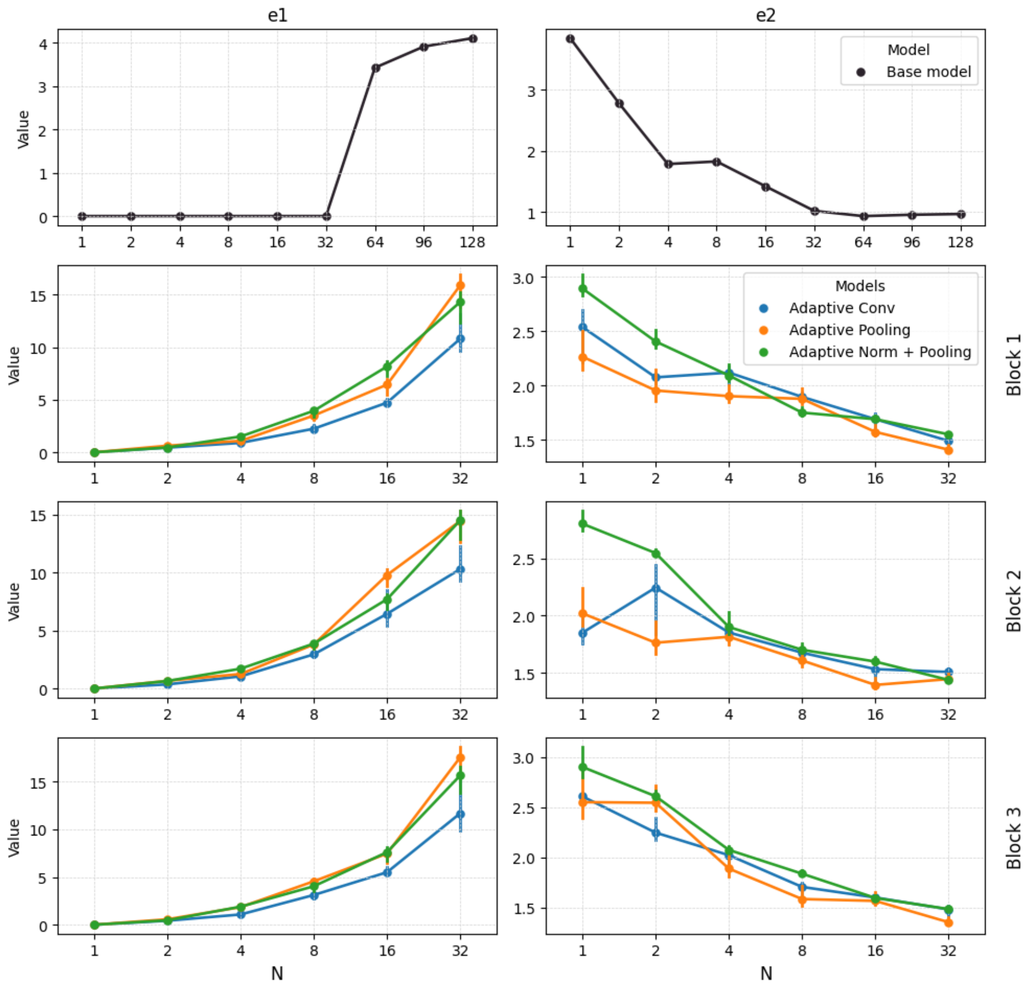

Figure 13. Orthogonality metrics for the Base model and Adaptive models. The first row presents the values obtained from the Base model. Block 1, Block 2, and Block 3 correspond to the sampling block, first downsampling, and the second downsampling block, respectively, in the architecture Fig. 7.

G. Hyperparameters

Architecture for Point Cloud Processing

Table 1. Number of Parameters $(\times 10^4)$

	Number of Parameters $(\times 10^4)$						
Model	N1	N2	N4	N8	N16	N32	N64
Base model	93.97						
Adaptive MLP	94.35	94.4	94.49	94.69	95.07	95.82	97.35

Table 2. Hyperparameters

Hyperparameter	**Value**
batch_size	12
set_abstraction_ratio_1	0.190
set_abstraction_radius_1	0.542
set_abstraction_ratio_2	0.380
set_abstraction_radius_2	0.188
set_abstraction_ratio_3	0.475
set_abstraction_radius_3	0.368
width1	0.494
width2	0.339
width3	0.290
width4	2.000
n_rings	5
learning_rate	0.0000087
weight_decay	0.088
frequency	3
dropout	0.100
channels_conv	[16, 32, 64]
channels_mlp	[32, 64, 128]
activ	Quotient
max_num_neighbors	64

Architecture for Voxel Data Processing

Table 3. Number of Parameters $(\times 10^5)$

	Number of Parameters $(\times 10^5)$						
Model	N1	N2	N4	N8	N16	N32	N64
Base model	15.398						
Adaptive Conv	15.4	15.43	15.5	15.8	17	21.7	-
Adaptive Pooling	15.4	15.4	15.4	15.4	15.46	15.51	-
Adaptive Norm + Pooling	14.72	14.73	14.85	14.78	14.83	14.9	15.17

Table 4. Hyperparameters

Parameter	**Value**
batch_size	32
learning_rate	0.0001
activ	FourierELU
frequency	2
channels	[10, 20, 40, 60, 60, 60, 80]
kernel_sizes	[7, 5, 3, 3, 3, 1]
paddings	[2, 2, 1, 2, 1, 0]
strides	[1, 1, 2, 2, 1, 1]