Steerable Transformers for Volumetric Data

Soumyabrata Kundu [1] **Risi Kondor** [1 2]

Abstract

We introduce Steerable Transformers, an extension of the Vision Transformer mechanism that maintains equivariance to the special Euclidean group $\mathrm{SE}(d)$. We propose an equivariant attention mechanism that operates on features extracted by steerable convolutions. Operating in Fourier space, our network utilizes Fourier space non-linearities. Our experiments in both two and three dimensions show that adding steerable transformer layers to steerable convolutional networks enhances performance.

1. Introduction

Transformers have emerged as the preferred architecture for natural language processing tasks, with recent powerful models like Chat-GPT employing this framework. Their relatively straightforward design, coupled with remarkable success, has led to their widespread adoption across various domains, including image classification (Dosovitskiy et al., 2021), object detection (Carion et al., 2020), and graph-based problems (Dwivedi & Bresson, 2020). The self-attention mechanism (Bahdanau et al., 2014) employed in transformer architectures has proven to be crucial for capturing relationships between different parts of input sequences. Dosovitskiy et al. (2021) introduced transformers as an alternative to traditional convolutional architectures for image classification tasks. Unlike convolutional neural networks (CNNs), which focus on local neighborhoods, transformers excel at capturing relations across different parts of the input.

Equivariant neural network architectures have gained significant popularity in recent years due to their inherent ability to comprehend the underlying symmetries of problems, making them highly effective tools for real-world scenarios characterized by such symmetries. For instance, achieving equivariance to the permutation group S_n is crucial in graph-based problems, where the structure of the graph remains invariant under permutations. Similarly, in image-related tasks, it is desirable to have equivariance to rigid body transformations such as rotations and translations. Recent studies by Worrall et al. (2017); Marcos et al. (2017), and many more, have demonstrated the remarkable efficacy of these equivariant architectures in addressing such challenges, without relying on brute-force techniques like data augmentation.

In vision tasks, steerable convolutions have proven to be a powerful tool for achieving equivariance to the group $\mathrm{SE}(d)$ (see supplementary material for definition), which encompasses rotations and translations in d dimensions (Worrall et al., 2017; Cohen et al., 2018; Weiler et al., 2018a). Steerable convolution networks, as described by Cohen & Welling (2016; 2017); Weiler et al. (2018a), primarily operate in Fourier space, leveraging representations of the underlying group to encode rotation and translation equivariant features from the data. Nevertheless, these architectures, rooted in convolutions, retain the ability to learn local neighborhoods. Conversely, transformer architectures excel in learning relationships between different regions of an image. Our contribution lies in integrating these two concepts—vision transformers and steerable convolutions—yielding steerable transformers capable of effectively capturing both local and global patterns in images.

1.1. Related Work

The original idea of attention was introduced by Bahdanau et al. (2014), for sequence modeling. Vaswani et al. (2017) introduced the transformer encoder-decoder architecture for translation tasks in natural language processing. Dosovitskiy et al. (2021) proposed the transformer architecture for vision tasks. They advocated for a hybrid architecture where the features are extracted from traditional CNN. Ramachandran et al. (2019) introduced a stand alone attention based architecture which completely replaces traditional CNNs. Shaw et al. (2018) proposed learnable relative positional encoding instead of absolute positional encoding in the original work of Vaswani et al. (2017), to encode the relative positions of input sequences.

[1]Department of Statistics, University of Chicago, Chicago, USA [2]Department of Computer Science, University of Chicago, Chicago, USA. Correspondence to: Soumyabrata Kundu <soumyabratakundu@uchicago.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

These transformer architectures are not linear in their inputs and hence break the equivariance structure of traditional CNNs. Since then there has been a flurry of papers trying to make these models equivariant. Romero & Cordonnier (2021) showed that vision transformers with relative positional encoding are translation equivariant. They extended the idea of Ramachandran et al. (2019) to incorporate rotation and translation equivariance in the 2D regime, by lifting the attention mechanism to be defined on the group $\mathrm{SE}(2)$. For this they proposed a positional encoding operator and use regular representations of the group instead of irreducible representations. Xu et al. (2023) introduced a novel positional encoding in the method proposed by Romero & Cordonnier (2021) to enhance it's performance. Romero et al. (2020) incorporated attention in convolution itself by defining attention score on the equivariance group.

In the 3D regime, there have been efforts to use attention in vision task, however to the best of our knowledge, only on point cloud data and not volumetric data. Fuchs et al. (2020) proposed an $\mathrm{SE}(3)$ equivariant architecture for point cloud data by adding self-attention mechanism to the rotation, translation and permutation equivariant Tensor Field network (Thomas et al., 2018). Liao & Smidt (2023) developed Equiformer, an equivariant version of Graph Attention Network by Velicković et al. (2018) for 3D point cloud data and later Liao et al. (2024) introduced Equiformerv2, a more efficient version, using insights from Passaro & Zitnick (2023) to reduce $\mathrm{SO}(3)$ convolutions to $\mathrm{SO}(2)$ convolutions. Chen & Villar (2022) achieved equivariance in attention mechanisms by mapping the data onto $\mathrm{SE}(3)$ equivariant and invariant features. These maps are based on the work of Villar et al. (2021). Hutchinson et al. (2021) introduced an attention mechanism that is equivariant to the action of Lie groups using regular representations of the group and built a transformer architecture based on that. Chatzipantazis et al. (2023) introduced $\mathrm{SE}(3)$ equivariant attention for shape reconstruction tasks.

1.2. Our Contribution

Romero & Cordonnier (2021) have explored steerable transformers for volumetric data in the context of two dimensional images. Their method integrates an attention mechanism within convolutional layers, which comes at the cost of significantly higher memory usage. In contrast, the architectures proposed in Xu et al. (2023); Romero & Cordonnier (2021) base their design on Ramachandran et al. (2019), where they adopt a standalone approach and completely replace the convolution architecture for an attention based mechanism. However, as highlighted by Xiao et al. (2021), incorporating a convolutional encoder before transformers typically enhances performance, offering a more effective design.

In the context of three dimensions, Fuchs et al. (2020) and Hutchinson et al. (2021) have explored $\mathrm{SE}(3)$ equivariant transformers for point cloud data. While it is theoretically possible to interpret volumetric data as point clouds positioned on a grid, applying point cloud-based algorithms designed for sparse point clouds to dense volumetric grids can introduce unsuitable inductive biases. Considering volumetric data as point clouds disregards the inherent grid structure that is crucial for our approach. Furthermore, treating volumetric data as point clouds can cause out of memory errors due to the increased computational load.

To address these challenges, we present a novel steerable transformer architecture specifically designed for *volumetric data* in d-dimensions. Our proposed steerable transformer integrates on top of a steerable convolutional encoder and, as demonstrated through our experiments, enhances the performance of steerable architectures. Drawing inspiration primarily from Dosovitskiy et al. (2021) and Fuchs et al. (2020), our method operates in the Fourier domain, similar to steerable CNNs (Cohen & Welling, 2017), by using group representations to learn equivariant features. The self-attention mechanism follows a similar structure to Fuchs et al. (2020). Similar to Vaswani et al. (2017), we integrate these transformers alongside steerable convolutions. Both Dosovitskiy et al. (2021) and Fuchs et al. (2020) employ learnable positional encodings, whereas Vaswani et al. (2017) relies on predefined positional functions. Our work combines these two strategies by scaling a fixed function with a learnable parameter. Finally, in contrast to Xu et al. (2023); Romero & Cordonnier (2021), which implement invariant positional encodings, our approach leverages equivariant positional encoding.

2. Background

In this section we introduce the relevant background required to design the steerable transformer architecture.

2.1. The Multihead Self-Attention Mechanism

The attention mechanism, initially introduced by Bahdanau et al. (2014) for sequence-to-sequence tasks, involves three main components: query vectors q_i, key vectors k_i, and value vectors v_i for each input in a sequence. These vectors are utilized to compute attention weights, denoted α_{ij}, which determine the relevance of each input vector in producing the output. The score function $s(q_i, k_j)$ calculates the compatibility between query and key vectors. The softmax of these scores yields the attention weights α_{ij}. Finally, we get the output by linearly combining α_{ij} with the value vectors v_i:

$$\textsc{Atn}(q_i, \{k_i\}, \{v_i\}) = \sum_{j=1}^{N} \alpha_{ij} v_j$$

where the attention weights α_{ij} are computed as,

$$\alpha_{ij} = \frac{\exp(s(q_i, k_j))}{\sum_{j=1}^{N} \exp(s(q_i, k_j))} \quad s(q_i, k_j) = \frac{q_i^T k_j}{\sqrt{d_K}}.$$

Vaswani et al. (2017) proposed a Multihead version of this attention mechanism which involves stacking multiple attention mechanisms, concatenating their outputs and applying a linear transformation to get the final result

$$\textsc{Multi-Atn}(q_i, \{k_i\}, \{v_i\}) = \text{Concat}(\textsc{Atn}^1, \ldots, \textsc{Atn}^h) W_O.$$

Here, $W_O \in \mathbb{R}^{hd_V \times d_{\text{model}}}$, and d_{model} is the dimension of the output which is same as the dimension of the input vectors. This dimension is fixed through the Transformer encoder structure. Typically, the query, key, and value vectors are obtained through learnable linear embedding followed by adding *positional encoding*.

Positional Encoding: Positional encoding in transformers is a technique employed to incorporate positional information about words or tokens into the input embeddings of a sequence. In vision tasks, these encodings inform the network about the original locations of different portions of the image. In the seminal work of Vaswani et al. (2017), they utilized sine and cosine functions of varying frequencies for positional encoding. Relative positional encoding, introduced by Shaw et al. (2018), represents an extension or variation of traditional positional encoding. It's goal is to use learnable position encodings that capture the relative distances between input sequences. As we explain below, relative positional encoding is crucial for achieving equivariance.

2.2. Vision Transformers

Dosovitskiy et al. (2021) extended the use of the transformer architecture to vision tasks. In their approach, they divided the image into equal-sized portions and fed them into the encoder part of the transformer architecture. They utilize the features obtained from the encoder for classification. Assuming a single input channel, the vision transformer layer can be summarized as follows:

$$\begin{aligned} z_0 &= [\mathbf{x}_1 E, \mathbf{x}_2 E, \ldots, \mathbf{x}_N E] + E_{\text{pos}} \\ z'_\ell &= \textsc{Multi-Attn}(\text{LN}(z_{\ell-1})) + z_{\ell-1} \quad \ell = 1, \ldots L \\ z_\ell &= \text{MLP}(\text{LN}(z_{\ell-1})) + z'_\ell \quad \ell = 1, \ldots L. \end{aligned}$$

Here, $\mathbf{x}_i \in \mathbb{R}^p$ represents a portion of the image which serves as the input, $E \in \mathbb{R}^{p \times d_{\text{model}}}$ is the learnable linear embedding matrix, $E_{\text{pos}} \in \mathbb{R}^{N \times d_{\text{model}}}$ is the positional encoding, and ℓ represents the various blocks of the transformer. LN denotes a normalization layer, and MLP stands for a Multilayer Perceptron. d_{model} is the dimension of the embedding fed into the transformer, which remains fixed throughout the entire encoder layer. Typically, the MLP layer comprises two hidden layers separated by a nonlinearity (Vaswani et al., 2017; Dosovitskiy et al., 2021).

2.3. Equivariance

Equivariance refers to a property where a mapping between vector spaces preserves the action of a group G. For each $g \in G$, a linear map $T_g : \mathcal{V} \to \mathcal{V}$ is defined on a vector space $\mathcal{V}$. Given actions T_g and T'_g on vector spaces $\mathcal{V}$ and $\mathcal{W}$ respectively, a map $\phi : \mathcal{V} \to \mathcal{W}$ is considered equivariant if:

$$\phi(T_g(\mathbf{v})) = T'_g(\phi(\mathbf{v})) \qquad \mathbf{v} \in \mathcal{V}.$$

Equivariance naturally emerges as a constraint in various real-world scenarios. For instance, in graph neural networks, equivariance to the permutation group S_n is fundamental (Thiede et al., 2020; Wang et al., 2020). Similarly, in vision tasks, equivariance to rotation and reflection symmetries, i.e., the action of the Euclidean group, is desirable. There is now an extensive literature on equivariant neural networks, in particular, focusing on cases where ϕ is linear, for instance the works of Worrall et al. (2017); Cohen et al. (2018); Weiler et al. (2018a); Cohen & Welling (2017; 2016); Weiler & Cesa (2019); Kondor et al. (2018); Anderson et al. (2019).

2.4. Steerable Convolutions

Steerable Networks are engineered to exhibit equivariance to the special Euclidean group $\mathrm{SE}(d)$ in d dimensions. Consider a d-dimensional image with a single input channel, represented by the function $f^{\text{in}} : \mathbb{R}^d \to \mathbb{R}$, which has compact support. These networks employ steerable convolution layers, which can be expressed as convolutions on the group $\mathrm{SE}(d)$ (Weiler et al., 2018a; Weiler & Cesa, 2019). The output of a steerable convolution layer is a function on $f^{\text{out}} : \mathrm{SE}(d) \to \mathbb{R}$ and can be expressed as

$$f^{\text{out}}(\mathbf{x}, R) = \int_{\mathbb{R}^d} f^{\text{in}}(\mathbf{x} + R\mathbf{y})\, w(\mathbf{y})\, d\mathbf{y}, \tag{1}$$

$$f^{\text{out}}(\mathbf{x}, R) = \int_{\mathrm{SO}(d)} \int_{\mathbb{R}^d} f^{\text{in}}(\mathbf{x} + R\mathbf{y}, RR') \, w(\mathbf{y}, R')\, d\mathbf{y}\, d\mu(R'). \tag{2}$$

Here, $\mathbf{x} \in \mathbb{R}^d$, $R \in \mathrm{SO}(d)$ and μ denotes the Haar measure on $\mathrm{SO}(d)$. Equation (1) represents the formula for the first layer, while (2) represents subsequent layers. The presence

of two distinct formulas is due to the fact that the input to the first layer is a vector field on $\mathbb{R}^d$, whereas the input to subsequent layers is a vector field on $\mathrm{SE}(d)$. These linear mappings possess a key property: whenever the input map f^{in} undergoes a transformation by an element in $\mathrm{SE}(d)$, the resulting output f^{out} also undergoes the same transformation. This property means that, if under the action of the group f^{in} changes to

$$(t, R) \cdot f^{\text{in}}(\mathbf{x}) = f^{\text{in}}(R\mathbf{x} + t) \tag{3}$$

then f^{out} changes to

$$(t, R) \cdot f^{\text{out}}(\mathbf{x}, R') = f^{\text{out}}(R\mathbf{x} + t, RR'). \tag{4}$$

For compact groups like $\mathrm{SO}(d)$, where discretization is not straightforward, it is advantageous to compute the rotation component of the output feature map of (2) in Fourier space (Worrall et al., 2017). Leveraging the convolution theorem, equation (2) in Fourier space becomes a product of the Fourier transforms of the input and the weights. The Fourier transform of a function f on a compact group can be computed by integrating it against it's irreducible representations or *irreps* (see supplementary material for definition):

$$\widehat{f}(\rho) = \int_{\mathrm{SO}(d)} f(R)\, \rho(R)\, d\mu(R).$$

By the translation property of Fourier Transform, these Fourier space vector fields possess the property that under input transformations like (3), the output transforms to $\rho(R)\widehat{f^{\text{out}}}(R\mathbf{x}+t, \rho)$ (Cohen & Welling, 2017; Weiler et al., 2018a).

In the 2D case, the irreps of $\mathrm{SO}(2)$ are just complex exponential indexed by integers, $\rho_k(\theta) = e^{\iota k\theta}$. Therefore, the k'th frequency component of the output feature map is a function $\widehat{f^{\text{out}}_k} : \mathbb{R}^2 \to \mathbb{C}$. For the 3D case, the irreps of $\mathrm{SO}(3)$ are indexed by integers $\ell \geq 0$, represented by $\rho_\ell(R) = D^\ell(R)$, known as the Wigner matrices (Wigner, 2012), which are unitary matrices of dimension $2\ell+1$. Correspondingly, the ℓ^{th} Fourier component of the output feature map is a function $\widehat{f^{\text{out}}_\ell} : \mathbb{R}^3 \to \mathbb{C}^{2\ell+1}$.

Non-linearities: In steerable neural networks, utilizing non-linearities like ReLU in Fourier space can disrupt equivariance. One approach to mitigate this issue is to transition back to real space, apply the desired non-linearity there, and then return to Fourier space. This method, for instance, is employed by Cohen et al. (2018). However, this back-and-forth transformation between real and Fourier space can be computationally expensive and error-prone, especially on groups like $\mathrm{SO}(3)$, where the grid is non-uniform.

As an alternative, Fourier space non-linearities are preferred in steerable neural networks. These non-linearities act directly on Fourier space while preserving equivariance. Several such non-linearities can be found in the literature. For instance, Worrall et al. (2017) apply non-linearity to the norm of the Fourier vector, as the norm remains invariant under rotations. Other works such as Kondor et al. (2018); Anderson et al. (2019) utilize the Clebsch–Gordan non-linearity, which involves a tensor product of Fourier vectors followed by a Clebsch-Gordan decomposition. Weiler et al. (2018a) introduce another steerable convolution filter to serve as a non-linearity. These methods enable effective non-linear transformations in Fourier space, while maintaining equivariance.

3. Method

The methodology behind steerable transformers builds on the foundation established by vision transformers, which rely on linear embeddings of fixed-size image patches along with positional encoding for these patches. A hybrid architecture discussed in the same work integrates features maps produced by CNNs as input to the transformer encoder. In the case of steerable transformers, we adopt this hybrid approach, with feature maps generated by steerable convolution layers serving as input to the transformer encoder (Dosovitskiy et al., 2021).

In Section 2.4, we discussed that the output of steerable networks is a compactly supported function on a continuous domain. However, practical neural networks operate on finite resolution rasterized images. Consequently, in real-world implementations, the output features for a particular irrep are represented as a collection of complex vectors on a grid. For simplicity, let us assume that there are N such grid points. For d_{model} channels, the input sequence to the transformer is described by:

$$f^{\text{in}}(\mathbf{x}_i, \rho) \in \mathbb{C}^{d_\rho \times d_{\text{model}}}, \quad i = 1, 2, \ldots, N. \tag{5}$$

In two dimensions, all components are one-dimensional, while in three dimensions, the ℓ^{th} component has dimension $2\ell + 1$.

3.1. Positional Encoding

Before we delve into the steerable version of self-attention, let us first explore a steerable approach to positional encoding. Vaswani et al. (2017) used fixed periodic functions of varying frequencies to encode positions. In contrast, Dosovitskiy et al. (2021) employed learnable positional embeddings for vision transformers. In our approach, we adopt a hybrid method by having both learnable and fixed components in positional encoding.

Relative positional encoding, introduced by Shaw et al. (2018), considers positional encoding as learned functions of the relative distances between input sequences.

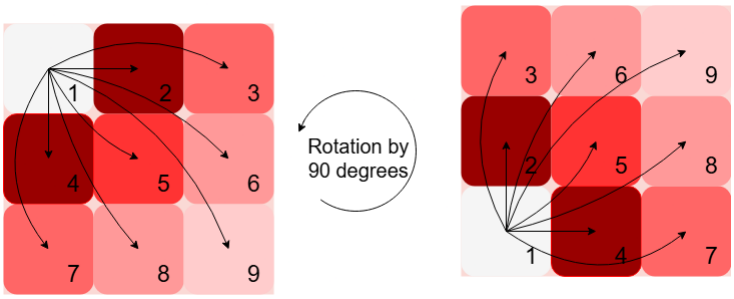

Figure 1: Visual Representation of steerable positional encoding. The arrows indicate the directional component of the relative positional encoding, which transforms consistently with the input image. The color gradient reflects the encoding's magnitude, which decays proportional to r^{-2}.

Romero & Cordonnier (2021) demonstrated that self-attention mechanisms with relative positional encoding exhibit translation equivariance. Therefore, the positional embeddings we seek are functions $P_{ij}^{\rho} = P(\mathbf{x}_i - \mathbf{x}_j, \rho)$, inherently invariant to translations.

To introduce the notion of steerability, we aim for P_{ij}^{ρ} to transform as $\rho(R)P_{ij}^{\rho}$ whenever the grid locations $\mathbf{x}_i$ undergo a rotation R. Essentially we will be using the Fourier basis to enforce the equivariance property (3,4). In two dimensions, this can be summarized as

$$P(\mathbf{x}, k) = \phi(r, k)e^{-ik\theta}, \quad \mathbf{x} = \begin{bmatrix} r\cos\theta \\ r\sin\theta \end{bmatrix}, \quad k \in \mathbb{Z}. \quad (6)$$

Here, $\phi(r, k)$ modulates the encoding strength for nearby and distant points. In the 3D case, we have

$$P(\mathbf{x}, \ell) = \phi(r, l)Y^{(\ell)}(\theta, \phi), \quad \mathbf{x} = \begin{bmatrix} r\sin\theta\cos\phi \\ r\sin\theta\sin\phi \\ r\cos\theta \end{bmatrix}, \quad (7)$$

for $\ell \in \mathbb{Z}_{\geq 0}$. Here $Y^{(\ell)}$ denotes the spherical harmonics (Byerly, 1893). The steerability property of positional encoding implies $P_{ii}^{\rho} = 0$ for any ρ except for the constant representation, i.e., $k = 0$ and $\ell = 0$ in two and three dimensions, respectively (as $P_{ii}^{\rho} = \rho(R)P_{ii}^{\rho}$ for any R). This condition can be enforced by setting ϕ to zero when $r = 0$. For our experiments, we employed $\phi(r, \rho) = w_{\rho}r^{-2}\mathbb{1}_{r>0}$, where w_{ρ} is a learnable scalar. This kind of inverse square modulation of the radial component results in positional encoding assigning higher weight to neighboring points, and the weights decrease as points move further apart. In our actual implementation, we have different learnable scalars, not only for different Fourier components, but also for different heads and query dimensions.

3.2. Steerable Self-Attention Mechanism

The query, key, and value embeddings are obtained by multiplying the input f^{in} in (5) with learnable weight matrices. Additionally, positional encoding is incorporated into the key embeddings (Shaw et al., 2018).

$$\begin{aligned}
\mathbf{q}_i^{\rho} &= f^{\text{in}}(\mathbf{x}_i, \rho)\mathbf{W}_Q^{\rho} && \mathbf{W}_Q^{\rho} \in \mathbb{C}^{d_{\text{model}} \times d_K} \\
\mathbf{k}_{ij}^{\rho} &= f^{\text{in}}(\mathbf{x}_i, \rho)\mathbf{W}_K^{\rho} + P(\mathbf{x}_i - \mathbf{x}_j, \rho) && \mathbf{W}_K^{\rho} \in \mathbb{C}^{d_{\text{model}} \times d_K} \\
\mathbf{v}_{ij}^{\rho} &= f^{\text{in}}(\mathbf{x}_i, \rho)\mathbf{W}_V^{\rho} + P(\mathbf{x}_i - \mathbf{x}_j, \rho) && \mathbf{W}_V^{\rho} \in \mathbb{C}^{d_{\text{model}} \times d_V}
\end{aligned}$$

Next, the score function for each Fourier component is calculated by taking the dot product between the query and key embeddings. The attention scores are obtained by applying the softmax function to the raw scores. Since we are dealing with complex numbers, we use the absolute value of these scores:

$$s_{ij} = \sum_{\rho} \frac{\text{vec}(\mathbf{q}_j^{\rho})^{\dagger}\,\text{vec}(\mathbf{k}_{ij}^{\rho})}{\sqrt{d_K}} \qquad \alpha_{ij} = \frac{\exp(|s_{ij}|)}{\sum_{j'=1}^{N}\exp(|s_{ij'}|)}$$

Here, the vec operation flattens the matrix into a vector. Since the vectors are complex-valued, the dot product involves the conjugate transpose, denoted by $\cdot^{\dagger}$, instead of a regular transpose. Finally, the output for each position and Fourier component is computed as a weighted sum of the value vectors using the attention scores, and then combined again using scalar weights:

$$f^{\text{out}}(\mathbf{x}_i, \rho) = \sum_{j=1}^{N} \alpha_{ij}\mathbf{v}_{ij}^{\rho}.$$

This self-attention mechanism maintains equivariance, as demonstrated by the following proposition.

Proposition 1. *If the input to a steerable transformer self-attention layer transforms as $f^{in}(\mathbf{x}_i, \rho) \mapsto \rho(R)\, f^{in}(R\mathbf{x}_i + \mathbf{t}, \rho)$, then the output also transforms as $f^{out}(\mathbf{x}_i, \rho) \mapsto \rho(R)\, f^{out}(R\mathbf{x}_i + \mathbf{t}, \rho)$.*

Despite the softmax operation's lack of equivariance, in this scenario, the dot product structure of the score function maintains rotational invariance, thereby preserving equivariance. For multihead attention, this mechanism is repeated independently for h heads. Finally, the outputs are

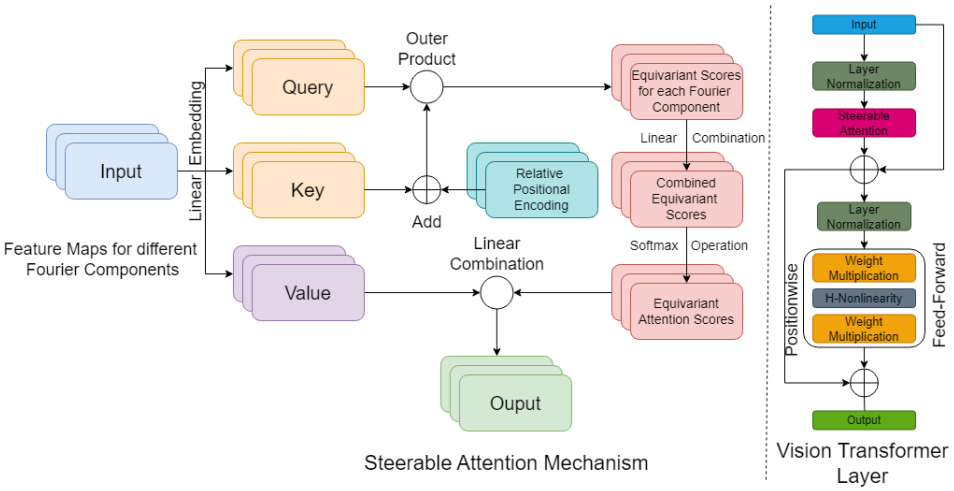

Figure 2: The schematic illustrates the steerable attention mechanism (left) and a steerable transformer encoder layer (right); c.f. Figure 1 by Dosovitskiy et al. (2021). The steerable attention mechanism is shown for a single head ($h = 1$) and one query dimension ($d_K = 1$). The stacks of feature maps represent different Fourier components, and the *H-Nonlinearity* denotes the harmonic non-linearity described by Worrall et al. (2017).

concatenated, and each Fourier component is scaled by another matrix $\mathbf{W}_O^\rho \in \mathbb{C}^{hd_V \times d_{\text{model}}}$. Consistent with Vaswani et al. (2017), we choose $d_V = d_K = d_{\text{model}}/h$.

3.3. Position-wise Feed Forward Layer

In a transformer architecture, the self-attention layer is typically followed by a Multilayer Perceptron (MLP) layer. This MLP layer commonly consists of two linear layers separated by a non-linearity (Vaswani et al., 2017). The linear layers themselves do not affect equivariance as they are applied to the input from the right:

$$f^{\text{out}}(\mathbf{x}_i, \rho) = \sigma\left(f^{\text{in}}(\mathbf{x}_i, \rho)W_1\right)W_2,$$

where $W_1, W_2^T \in \mathbb{C}^{d_{\text{model}} \times d_{\text{hidden}}}$. Here, d_{hidden} represents the hidden dimension between the two linear layers. However, when it comes to the non-linearity, caution is needed. As discussed in Section 2.4, typical non-linearities like ReLU or softmax may break equivariance. Instead, we require a Fourier space non-linearity. For our experiments, we utilize a non-linearity proposed by Worrall et al. (2017), where they apply ReLU to the magnitude of the equivariant vectors:

$$\sigma(f(\mathbf{x}, \rho)) = \frac{\text{ReLU}(||f(\mathbf{x}, \rho)|| + b)}{||f(\mathbf{x}, \rho)||} f(\mathbf{x}, \rho).$$

Here, b is a learnable weight. Since the norm of the features in Fourier space is invariant to rotations, this non-linearity preserves equivariance. Following the convention in Vaswani et al. (2017), we set $d_{\text{hidden}} = 2d_{\text{model}}$.

3.4. Layer Normalization

Another crucial component of the transformer architecture is layer normalization. Notably, the norm of equivariant vectors remains invariant under the Fourier transform, as it preserves the $\mathcal{L}_2$ norm. Leveraging this property, we can derive a steerable normalization as follows:

$$\text{LN}(f)(\mathbf{x}, \rho) = \frac{f(\mathbf{x}, \rho)}{\sqrt{\sum_\rho ||f(\mathbf{x}, \rho)||^2}}.$$

Indeed, the sum over ρ theoretically extends over all irreps of the group, but in practical implementations, it is truncated at particular values, which serves as a hyperparameter that can be tuned. This type of normalization was employed in the 2D case by Worrall et al. (2017).

3.5. Complexity

The computational complexity of a transformer block mainly stems from its Self-Attention (SA) mechanism and Feed-Forward Networks (FFN). For sequence length N and model dimension C, SA has a complexity of $O(NC^2 + N^2C)$, and the FFN adds $O(NC^2)$, totaling $O(NC^2 + N^2C)$. When $N \asymp C$, this becomes $O(NC^2)$. In steerable attention mechanisms, each component effectively has $d_\rho C$ channels, leading to a complexity of $O(Nd_\rho^2 C^2)$. Analogously, vanilla convolutions in d dimensions, processing N pixels with C input and output channels with a kernel size k^d has a complexity of $O(NC^2k^d)$. In steerable convolutions, replacing C with $d_\rho C$ yields $O(Nd_\rho^2 C^2 k^d)$.

In summary, with a fixed kernel size, the computational complexity of steerable attention matches that of convolu-

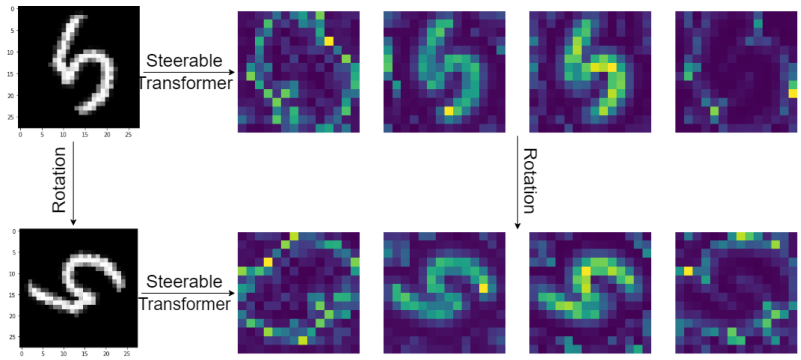

Figure 3: The figure demonstrates the equivariance of attention scores in a trained steerable transformer. For a fixed pixel i, we have plotted the maximum attention score for that pixel $(\max_j \alpha_{ij})$. The different subfigures represent individual heads. The first and last heads appear to capture the object's boundary, while the other two heads focus on the object's body.

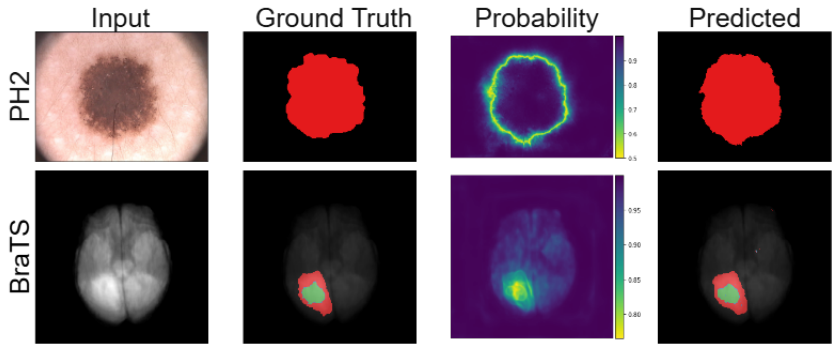

Figure 4: The figure shows the predicted and a ground truth segmentation, along with the predicted probability for the true class for one example in each dataset. The red in the PH2 example represent the binary mask. The red, blue and green in the BraTS dataset represent enhancing Tumor, tumor core and whole tumor respectively.

tion. Additionally, the increase in complexity from standard to steerable methods is similar for both attention mechanisms and convolutions when using many channels and small kernels, as is common in practice. This means the added complexity from steerable methods scales proportionally in both cases. This parallel implies that the additional complexity introduced by steerable methods scales proportionally across both attention mechanisms and convolution operations.

4. Experiments

We evaluate the performance of our steerable transformer architecture in both two and three dimensions. Specifically, we focus on two vision tasks: image classification and semantic segmentation. In all experiments, we employed a hybrid architecture that integrates steerable convolutional blocks with steerable transformer encoder layers, and the training was conducted using the Adam Optimizer (Kingma & Ba, 2014). Each steerable convolution block consists of two steerable convolutions, separated by Clebsch–Gordan nonlinearity (Anderson et al., 2019; Kondor et al., 2018), and is followed by layer normalization (Section 3.4).

In the classification networks, each convolution block is followed by an average pooling layer, and finally a steerable transformer encoder with two blocks. Afterward, a final convolution block with a kernel size equal to the input size is applied, functioning as the steerable equivalent of a flattening layer. This is followed by taking the norm, ensuring invariance to both rotation and translation. The final convolution layer is succeeded by two fully connected layers, with a ReLU activation between them and a dropout layer with a probability of 0.7 (Marcos et al., 2017). We report results on two datasets: Rotated MNIST (2D) and ModelNet10 (3D).

For the segmentation tasks, we used a U-net architecture (Ronneberger et al., 2015), with a steerable convolutional encoder for downsampling, and a decoder that combines

Datasets	Model	Frequency Cutoff	Accuracy / Dice Score	Parameter ($\sim \times 10^6$)	Avg Runtime Per Batch	
					Train (s)	Inference (s)
Rotated MNIST	Steerable Convolution	$k = 4$	$98.72_{\pm 0.02}$	1.18	0.14	0.08
		$k = 8$	$98.97_{\pm 0.01}$	2.54	0.31	0.15
	Steerable Transformer	$k = 4$	$\mathbf{98.82}_{\pm 0.04}$	1.13	0.13	0.08
		$k = 8$	$\mathbf{99.03}_{\pm 0.04}$	2.24	0.30	0.14
ModelNet10 (z Rotation) ($\mathrm{SO}(3)$ Rotation)	Steerable Convolution	$\ell = 4$	$90.13_{\pm 0.52}$ $86.62_{\pm 0.25}$	1.08	1.22	0.46
	Steerable Transformer	$\ell = 4$	$\mathbf{90.40}_{\pm 0.25}$ $\mathbf{86.80}_{\pm 0.58}$	0.92	1.20	0.45
PH2	Steerable Convolution	$k = 4$	$89.31_{\pm 0.17}$	0.35	0.27	0.09
		$k = 8$	$89.63_{\pm 0.30}$	0.96	0.67	0.19
	Steerable Transformer	$k = 4$	$\mathbf{90.61}_{\pm 0.90}$	0.44	0.28	0.10
		$k = 8$	$\mathbf{90.72}_{\pm 0.70}$	1.22	0.72	0.20
BraTS (Enhancing Tumor) (Tumor Core) (Whole Tumor)	Steerable Convolution	$\ell = 3$	$73.29_{\pm 0.76}$ $51.07_{\pm 1.23}$ $74.43_{\pm 0.77}$	0.12	5.04	1.00
	Steerable Transformer	$\ell = 3$	$\mathbf{75.01}_{\pm 0.32}$ $\mathbf{54.89}_{\pm 0.76}$ $\mathbf{76.37}_{\pm 0.46}$	0.16	5.37	1.01

Table 1: Comparison of steerable transformers and steerable convolutions. The radial resolution was fixed for these experiments. The mean and sd are reported for 5 runs. For ModelNet10 we have reported both the z rotation and $\mathrm{SO}(3)$ rotation variations. For PH2 dataset we have reported the dice score for segmentation of the binary mask. For the BraTS dataset we have reported the dice score individually for each tumor category.

interpolation with steerable convolutions to upsample the image back to the input resolution. The model includes skip connections, where features from the downsampling path are added to the corresponding resolution features in the upsampling path. Both the encoder and decoder consist of convolution blocks separated by average pooling layers. In the baseline model, a position-wise feed-forward layer is placed between the encoder and decoder, while in the steerable transformer model, this is replaced by a transformer encoder with two transformer blocks. We report results on two datasets: PH2 (2D) and BraTS (3D).

Table 1 presents a comparison between the baseline steerable convolutional architecture and the steerable transformer architecture. For the classification tasks, we report accuracy, while for the segmentation tasks, we provide the dice score for each class (excluding the background). For comparison of steerable transformers with other methods, please refer to Table 2 in the supplementary material. The code for the experiments can be found here.

Rotated MNIST: Rotated MNIST, a variant of the original MNIST dataset (LeCun et al., 2010), includes images that have been randomly rotated. The dataset contains 12,000 training images and 50,000 testing images. Table 1 compares the performance of steerable transformers and steerable convolutions, clearly showing that the addition of a transformer encoder to the steerable convolutions enhances performance, even with a low Fourier cutoff.

ModelNet10: The ModelNet10 dataset (Wu et al., 2015) consists of 3D CAD models from 10 common object categories, with a train/test split of 3991:908. Each category includes triangular meshes representing objects at various orientations and scales. For our experiments, $2,048$ points were uniformly sampled from the shape surfaces using the farthest point sampling algorithm. The point cloud data was then embedded into a grid to generate voxel data. Additionally, we introduced two types of perturbations: random rotations along the "gravity up" direction (z-axis) and random rotations from $\mathrm{SO}(3)$. The voxel values were also rescaled to $[-1, 5]$ instead of $[0, 1]$, following Brock et al. (2016). Table 1 presents the comparison between the steerable transformer architecture and steerable convolutions. The results show improvement in performance with the introduction of the steerable transformer encoder layer across both variations of the dataset.

PH2: The PH2 dataset (Mendonça et al., 2013) is a collection of 200 dermoscopic images curated for research and benchmarking in the classification and segmentation of skin lesions. We randomly split the dataset into 100:50:50 for training, testing, and validation, respectively. Each image has a resolution of 578×770 with RGB channels, and

is accompanied by comprehensive annotations, including binary masks of the same resolution that delineate lesion regions. Table 1 presents the dice score for predicting the binary mask in the segmentation task. Steerable transformers demonstrate superior performance compared to steerable convolutions.

BraTS: The Brain Tumor Segmentation (BraTS) dataset (Menze et al., 2015) is an essential resource for developing and evaluating algorithms for brain tumor segmentation in MRI scans. It includes pre-operative MRI scans across four modalities, with each image having a resolution of $240\times240\times155$ and annotated to delineate three tumor regions. We used a train/validation/test split of $243{:}96{:}145$. As shown in Table 1, steerable transformers outperform steerable convolutions in segmenting all three tumor regions.

5. Limitations

The use of transformer architecture comes with its challenges. For instance, the dot product structure used in attention scores can create a memory bottleneck for large images, restricting the network size and batch size during training. Scaling up these architectures is essential to fully understand the benefits of the attention mechanism in improving performance. Our experiments were conducted with relatively small Fourier cutoffs. We believe that using higher values for these Fourier cutoffs would enhance performance, but implementing them in a steerable transformer architecture would necessitate scaling up to accommodate larger models.

6. Conclusion

The transformer framework is increasingly being adopted across various machine learning fields, including computer vision, due to its exceptional ability to capture relationships across different parts of the input. In this work, we introduce steerable transformer encoders that can be used in conjunction with steerable convolutions. While convolutions excel at capturing local relationships, incorporating the attention mechanism, which captures global relationships, enhances performance, as demonstrated in our experiments. This architecture has the potential to make a significant impact in fields such as medical imaging, where equivariance is critical.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning, specifically computer vision and equivariant neural networks. This potentially has a positive societal impact as this architecture can be used in the field of biomedical imaging to further the advance the fields of AI assisted diagnosis and medicine.

References

Anderson, B., Hy, T. S., and Kondor, R. Cormorant: Covariant molecular neural networks. *Advances in neural information processing systems*, 32, 2019.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

Brock, A., Lim, T., Ritchie, J. M., and Weston, N. Generative and discriminative voxel modeling with convolutional neural networks. *ArXiv*, abs/1608.04236, 2016.

Byerly, W. E. *An elemenatary treatise on Fourier's series, and spherical, cylindrical, and ellipsoidal harmonics, with applications to problems in mathematical physics*. Dover Publicatiions, 1893.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers. In *European conference on computer vision*, pp. 213–229. Springer, 2020.

Chatzipantazis, E., Pertigkiozoglou, S., Dobriban, E., and Daniilidis, K. $\mathrm{SE}(3)$-equivariant attention networks for shape reconstruction in function space. In *The Eleventh International Conference on Learning Representations*, 2023.

Chen, N. and Villar, S. Se (3)-equivariant self-attention via invariant features. In *Machine Learning for Physics NeurIPS Workshop*, 2022.

Cohen, T. and Welling, M. Group equivariant convolutional networks. In *International conference on machine learning*, pp. 2990–2999. PMLR, 2016.

Cohen, T. S. and Welling, M. Steerable CNNs. In *International Conference on Learning Representations*, 2017.

Cohen, T. S., Geiger, M., Koehler, J., and Welling, M. Spherical CNNs. In *International Conference on Learning Representations*, 2018.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Dwivedi, V. P. and Bresson, X. A generalization of transformer networks to graphs. In *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*, 2020.

Fuchs, F., Worrall, D., Fischer, V., and Welling, M. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in neural information processing systems*, 33:1970–1981, 2020.

Hegde, V. and Zadeh, R. Fusionnet: 3d object classification using multiple data representations. *arXiv preprint arXiv:1607.05695*, 2016.

Hutchinson, M. J., Le Lan, C., Zaidi, S., Dupont, E., Teh, Y. W., and Kim, H. Lietransformer: Equivariant self-attention for lie groups. In *International Conference on Machine Learning*, pp. 4533–4543. PMLR, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

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 31*, pp. 10117–10126. Curran Associates, Inc., 2018.

Laptev, D., Savinov, N., Buhmann, J. M., and Pollefeys, M. Ti-pooling: Transformation-invariant pooling for feature learning in convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Larochelle, H., Erhan, D., Courville, A. C., Bergstra, J., and Bengio, Y. An empirical evaluation of deep architectures on problems with many factors of variation. In *International Conference on Machine Learning*, 2007.

LeCun, Y., Cortes, C., and Burges, C. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Li, J., Chen, B. M., and Lee, G. H. So-net: Self-organizing network for point cloud analysis. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9397–9406, 2018. doi: 10.1109/CVPR.2018.00979.

Li, J., Bi, Y., and Lee, G. H. Discrete rotation equivariance for point cloud recognition. In *2019 International conference on robotics and automation (ICRA)*, pp. 7269–7275. IEEE, 2019.

Liao, Y.-L. and Smidt, T. Equiformer: Equivariant graph attention transformer for 3d atomistic graphs. In *The Eleventh International Conference on Learning Representations*, 2023.

Liao, Y.-L., Wood, B. M., Das, A., and Smidt, T. Equiformerv2: Improved equivariant transformer for scaling to higher-degree representations. In *The Twelfth International Conference on Learning Representations*, 2024.

Marcos, D., Volpi, M., Komodakis, N., and Tuia, D. Rotation equivariant vector field networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5048–5057, 2017.

Maturana, D. and Scherer, S. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 922–928, 2015. doi: 10.1109/IROS.2015.7353481.

Mendonça, T., Ferreira, P. M., Marques, J., Marcal, A. R. S., and Rozeira, J. Ph2 - a dermoscopic image database for research and benchmarking. In *35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 5437–5440. IEEE, 2013.

Menze, B. H., Jakab, A., Bauer, S., Kalpathy-Cramer, J., Farahani, K., and et al., J. K. The multimodal brain tumor image segmentation benchmark (brats). *IEEE Transactions on Medical Imaging*, 34(10):1993–2024, 2015. doi: 10.1109/TMI.2014.2377694.

Munkres, J. *Topology; a First Course*. Prentice-Hall, 1974. ISBN 9780139254956.

Passaro, S. and Zitnick, C. L. Reducing so (3) convolutions to so (2) for efficient equivariant gnns. In *International Conference on Machine Learning*, pp. 27420–27438. PMLR, 2023.

Peter, F. and Weyl, H. Die vollständigkeit der primitiven darstellungen einer geschlossenen kontinuierlichen gruppe. *Mathematische Annalen*, 97(1):737–755, 1927.

Ramachandran, P., Parmar, N., Vaswani, A., Bello, I., Levskaya, A., and Shlens, J. Stand-alone self-attention in vision models. In *Advances in neural information processing systems*, volume 32, 2019.

Robert, A. *Introduction to the representation theory of compact and locally compact groups*, volume 80. Cambridge University Press, 1983.

Romero, D., Bekkers, E., Tomczak, J., and Hoogendoorn, M. Attentive group equivariant convolutional networks. In *International Conference on Machine Learning*, pp. 8188–8199. PMLR, 2020.

Romero, D. W. and Cordonnier, J.-B. Group equivariant stand-alone self-attention for vision. In *International Conference on Learning Representations*, 2021.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241. Springer, 2015.

Sedaghat, N., Zolfaghari, M., Amiri, E., and Brox, T. Orientation-boosted voxel nets for 3d object recognition. In *British Machine Vision Conference (BMVC) 2017*, 2016.

Shaw, P., Uszkoreit, J., and Vaswani, A. Self-attention with relative position representations. In *North American Chapter of the Association for Computational Linguistics*, 2018.

Simonovsky, M. and Komodakis, N. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 29–38, 2017. doi: 10.1109/CVPR.2017.11.

Sohn, K. and Lee, H. Learning invariant representations with local transformations. In *International Conference on Machine Learning*, 2012.

Thiede, E. H., Hy, T. S., and Kondor, R. The general theory of permutation equivarant neural networks and higher order graph variational encoders. *arXiv preprint arXiv:2004.03990*, 2020.

Thomas, N., Smidt, T. E., Kearnes, S. M., Yang, L., Li, L., Kohlhoff, K., and Riley, P. F. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds. *ArXiv*, abs/1802.08219, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Velicković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Villar, S., Hogg, D. W., Storey-Fisher, K., Yao, W., and Blum-Smith, B. Scalars are universal: Equivariant machine learning, structured like classical physics. *Advances in Neural Information Processing Systems*, 34: 28848–28863, 2021.

Wang, R., Albooyeh, M., and Ravanbakhsh, S. Equivariant networks for hierarchical structures. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 13806–13817. Curran Associates, Inc., 2020.

Weiler, M. and Cesa, G. General e (2)-equivariant steerable cnns. *Advances in neural information processing systems*, 32, 2019.

Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. S. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. *Advances in Neural Information Processing Systems*, 31, 2018a.

Weiler, M., Hamprecht, F. A., and Storath, M. Learning steerable filters for rotation equivariant cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018b.

Wigner, E. *Group theory: and its application to the quantum mechanics of atomic spectra*, volume 5. Elsevier, 2012.

Worrall, D. E. and Brostow, G. J. Cubenet: Equivariance to 3d rotation and translation. In *European Conference on Computer Vision*, 2018.

Worrall, D. E., Garbin, S. J., Turmukhambetov, D., and Brostow, G. J. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5028–5037, 2017.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.

Xiao, T., Singh, M., Mintun, E., Darrell, T., Dollár, P., and Girshick, R. Early convolutions help transformers see better. *Advances in neural information processing systems*, 34:30392–30400, 2021.

Xu, R., Yang, K., Liu, K., and He, F. $e(2)$-equivariant vision transformer. In *Uncertainty in Artificial Intelligence*, pp. 2356–2366. PMLR, 2023.

A. Prerequisites

A.1. Groups

Definition : A group is a non-empty set G together with a binary operation (commonly denoted by "$\cdot$", that combines any two elements a and b of G to form an element of G, denoted $a \cdot b$, such that the following three requirements, known as group axioms, are satisfied:

- *Associativity* : For all $a, b, c \in G$, one has $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
- *Identity Element* : There exists as element $e \in G$ such that for every $a \in G$, one has $e \cdot a = a \cdot e = a$. Such an element is unique. It is called the *identity element* of the group.
- *Inverse* : For each $a \in G$, there exists an element $b \in G$ such that $a \cdot b = b \cdot a = e$, where e is the identity element. For each a, the element b is unique and it is called the inverse of a and is commonly denoted by a^{-1}.

Examples include set of integers $\mathbb{Z}$ with the addition operation and the set of non-zero reals $\mathbb{R} \setminus \{0\}$ with the multiplication operation. From here on, we will drop '$\cdot$', for simplicity. The group operation will be clear from the elements of the group concerned.

Group Homomorphism : Given two groups G and H, a function $\phi : G \to H$ is called a group homomorphism if $\phi(ab) = \phi(a)\phi(b)$ for any $a, b \in G$. If the map ϕ is a bijection, it is called an *isomorphism* and furthermore, if $G = H$, then an isomorphism is called *automorphism*. The set of all automorphisms of a group G with the operation of composition form a group in itself and is denoted by $\mathrm{Aut}(G)$.

Compact Groups : A topological group is a topological space that is also a group such that the group operation and the inverse map are continuous. A compact group is a topological group whose topology realizes it as a compact topological space (see Munkres (1974) for definition of topological and compact topological spaces). Some classic examples of compact groups are the groups $\mathrm{SO}(d)$ (the group of all real orthogonal matrices in d dimensions with determinant 1), $\mathrm{U}(d)$ (the group of all complex unitary matrices) and $\mathrm{SU}(d)$ (the group of all complex unitary matrices with determinant 1).

Special Orthonormal Group $\mathrm{SO}(d)$*:* This group comprises all real orthogonal matrices in d dimensions with a determinant of 1. These groups are associated with rotation matrices in d dimensions. Specifically, we are focused on the groups $\mathrm{SO}(2)$ and $\mathrm{SO}(3)$ in 2 and 3 dimensions, respectively. $\mathrm{SO}(2)$ can be parametrized by a single angle θ, corresponding to the rotation matrix:

$$R(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

Here, $R(\theta)$ signifies rotation in the x-y plane by an angle $\theta \in [0, 2\pi)$. The group $\mathrm{SO}(3)$ can be parameterized by the so-called Euler angles. In our discussions, we adopt the z-y-z convention:

$$R_z(\alpha) = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \quad R_y(\beta) = \begin{bmatrix} \cos(\beta) & -\sin(\beta) & 0 \\ \sin(\beta) & \cos(\beta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Here, $R_z(\alpha)$ represents rotation about the z-axis by α radians, and $R_y(\beta)$ represents rotation about the y-axis by β radians. An element $R \in \mathrm{SO}(3)$ can then be expressed as $R = R_z(\alpha)R_z(\beta)R_y(\gamma)$, where $\alpha, \gamma \in [0, 2\pi)$ and $\beta \in [0, \pi)$ are the Euler angles.

Semi-direct Product Groups : Given two groups N and H and a group homomorphism $\phi : H \to \mathrm{Aut}(N)$, we can construct a new group $N \rtimes_\phi H$ defined as follows,

- The underlying set is the Cartesian product $N \times H$
- The group operation is given by $(n_1, h_1)(n_2, h_2) = (n_1\phi_{h_1}(n_2), h_1h_2)$

Special Euclidean Group SE(d) : The special Euclidean group $\mathrm{SE}(d)$ encompasses all rotations and translations in d dimensions. The translation group for an d dimensional real vector is isomorphic to $\mathbb{R}^n$ and the rotation group is given by $\mathrm{SO}(n)$. Then, we can define a group homomorphism $\phi : \mathrm{SO}(d) \to \mathrm{Aut}(\mathbb{R}^d)$, as $[\phi(R)](t) = Rt$, for any rotation $R \in \mathrm{SO}(d)$ and a translation vector $t \in \mathbb{R}^d$. Then, the special Euclidean group $\mathrm{SE}(d)$ is defined as $\mathbb{R}^d \rtimes_\phi \mathrm{SO}(d)$. Hence the group action is given by

$$(t_1, R_1)(t_2, R_2) = (t_1 + R_1 t_2, R_1 R_2)$$

for any $(t_1, R_1), (t_2, R_2) \in \mathrm{SE}(d)$, the identity is given by $(0, I)$, and the inverse is given by, $(t, R)^{-1} = (-R^{-1}t, R^{-1})$ for any $(t, R) \in \mathrm{SE}(d)$.

A.2. Group Actions

Definition : If G is a group with identity element e, and X is a set, then a (left) group action α of G on X is a function $\alpha : G \times X \to X$, that satisfies the following two axioms for all $g, h \in G$ and $x \in X$:

- *Identity*: $\alpha(e, x) = x$,
- *Compatibility*: $\alpha(g, \alpha(h, x)) = \alpha(gh, x)$.

Often $\alpha(g, x)$ is shortened to $g \cdot x$. Any group G acts on itself by the group operation $g \cdot h = gh$ for any $g, h \in G$. If G acts on X, then it can also act on any function f defined on X, as $(g \cdot f)(x) = f(g^{-1} \cdot x)$.

Action of $\mathrm{SE}(d)$*:* The special Euclidean group acts on a vector in $\mathbb{R}^d$ by first applying the rotation component followed by translation. For $x \in \mathbb{R}^d$ and $(t, R) \in \mathrm{SE}(d)$

$$(t, R) \cdot x = Rx + t,$$

gives us the action of $\mathrm{SE}(d)$ on $\mathbb{R}^d$.

A.3. Group Representations

Definition : A representation of a group G is a group homomorphism ρ from G to $\mathrm{GL}(\mathbb{C}^n)$ (group of invertible linear maps on $\mathbb{C}^n$), i.e,

$$\rho(g_1 g_2) = \rho(g_1)\rho(g_2) \quad \forall g_1, g_2 \in G$$

Here n is called the dimension of the representation, which can possibly be infinite. A representation is *unitary* if ρ maps to unitary linear transformation of $\mathbb{C}^n$.

Irreducible Representations : If we have two representations, ρ_1 and ρ_2 of dimensions n_1 and n_2 respectively, then the two can be combined by a direct sum to give another representation of dimension $n_1 + n_2$,

$$\rho_1(g) \oplus \rho_2(g) = \begin{bmatrix} \rho_1(g) & 0 \\ 0 & \rho_2(g) \end{bmatrix}.$$

A representation is said to be *completely reducible* if it can be expressed as a direct sum of other representations after maybe a change of basis, i.e,

$$U\rho(g)U^{-1} = \bigoplus_i \rho_i(g)$$

where U is a unitary change of basis matrix and the direct sum extends over some number of representations. However, for *every* group there are a some representations which cannot be broken further into a direct sum of other representations. These are called the *irreducible representations* or *irreps* of the group. These irreps are the building blocks of the all other representations of the group, in the sense that any representation can be written as a direct sum of the irreps:

$$\rho(g) = U \left[\bigoplus_i \rho^{(i)}(g) \right] U^{-1},$$

where again U is a change of basis matrix and $\rho^{(i)}$ are the irreps. The Peter-Weyl Theorem by Peter & Weyl (1927) tells us that for a compact group G, any unitary representation ρ is completely reducible and splits into direct sum of irreducible *finite dimensional unitary representations* of G.

Irreducible Representations of SO(2) and SO(3) : $\mathrm{SO}(n)$ being a compact group, all its irreps are finite dimensional unitary representations. The irreps of SO(2) are all of dimension one, and are indexed by the set of integers. The group SO(2) can be realized by 2×2 rotation matrices, with rotation angle θ:

$$R_\theta = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}.$$

With this identification of SO(2), the irreducible representations of SO(2) are given by,

$$\rho_k(\theta) = e^{ik\theta} \qquad k \in \mathbb{Z}.$$

The irreps correspond to the Fourier basis over $\mathbb{R}$. The SO(3) case is more involved. The irreducible representations are indexed by $\ell \in \{0, 1, 2, \dots\}$, where the ℓ'th representation is of dimension $2\ell + 1$:

$$\rho_\ell(R) = D^\ell(R) \qquad \ell \in \{0, 1, 2, \dots\},$$

where D^ℓ are unitary matrices called Wigner D-matrices.

A.4. Fourier Transform

Haar Measure : There is, up to a positive multiplicative constant, a unique countably additive, nontrivial measure μ on the Borel subsets of G satisfying the following properties:

- The measure μ is left-translation-invariant: $\mu(gS) = \mu(S)$ for every $g \in G$ and all Borel sets $S \subseteq G$.
- The measure μ is finite on every compact set: $\mu(K) < \infty$ for all compact $K \subseteq G$.
- The measure μ on Borel sets $S \subseteq G$ is given by

$$\mu(S) = \inf\{\mu(U) : S \subseteq U, U \text{ open}\}.$$

- The measure μ is inner regular on open sets $U \subseteq G$:

$$\mu(U) = \sup\{\mu(K) : K \subseteq U, K \text{ compact}\}.$$

Such a measure on G is called a left Haar measure. It can be shown that as a consequence of the above properties that $\mu(U) > 0$ for every non-empty open subset $U \subseteq G$. In particular, if G is compact then $\mu(G)$ is finite and positive, so we can uniquely specify a left Haar measure on G by adding the normalization condition $\mu(G) = 1$.

Fourier Transform on Compact groups : A nice property of compact groups is that the set of (isomorphism classes of) irreducible representations of G is countable (for details see Robert (1983)). If we have a complex valued function supported on a compact group G, $f : G \to \mathbb{C}$, then the Fourier transform of f is given by

$$\hat{f}(\rho) = \frac{1}{\mu(G)} \int_G f(g)\rho(g) d\mu(G),$$

where μ is the Haar measure.

B. Proofs

B.1. Proof of Proposition 1

Proof. We will assume number of input channels $C = 1$. If the input is rotated and translated by $(t, R) \in \mathrm{SO}(d)$ then input to the transformer encoder transforms as $f^{\text{in}}(\mathbf{x}, \rho) \mapsto \rho(R) f^{\text{in}}(R\mathbf{x} + t, \rho)$ and positional encoding transforms as $P(\mathbf{x}, \rho) \mapsto \rho(R)P(R\mathbf{x}, \rho)$. We change the notation to $\mathbf{q}(\mathbf{x}_i, \rho) = \mathbf{q}_i^\rho$, and similarly for $\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j, \rho)$ and $\mathbf{v}(\mathbf{x}_i, \mathbf{x}_j, \rho)$. Then, under the action of (t, R) on the input the query, key and value vectors change as follows:

$$
\begin{aligned}
&\mathbf{q}(\mathbf{x}_i, \rho) \mapsto \rho(R) f^{\text{in}}(R\mathbf{x}_i + t, \rho)\mathbf{W}_Q^\rho = \rho(R)\mathbf{q}(R\mathbf{x}_i + t, \rho) \\
&\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j, \rho) \mapsto \rho(R) f^{\text{in}}(R\mathbf{x}_i + t, \rho)\mathbf{W}_K^\rho + \rho(R)P(R(\mathbf{x}_i - \mathbf{x}_j), \rho) = \rho(R)\mathbf{k}(R\mathbf{x}_i + t, R\mathbf{x}_j + t, \rho) \\
&\mathbf{v}(\mathbf{x}_i, \mathbf{x}_j, \rho) \mapsto \rho(R) f^{\text{in}}(R\mathbf{x}_i + t, \rho)\mathbf{W}_V^\rho + \rho(R)P(R(\mathbf{x}_i - \mathbf{x}_j), \rho) = \rho(R)\mathbf{v}(R\mathbf{x}_i + t, , R\mathbf{x}_j + t, \rho).
\end{aligned}
$$

Let $s(\mathbf{x}_i, \mathbf{x}_j) = s_{ij}$ and $\alpha(\mathbf{x}_i, \mathbf{x}_j) = \alpha_{ij}$. Then using the fact $\rho(R)$ is unitary we have,

$$
\begin{aligned}
s(\mathbf{x}_i, \mathbf{x}_j) &= \sum_\rho \frac{\mathbf{q}(\mathbf{x}_i, \rho)^\dagger \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j, \rho)}{\sqrt{d_k}} \\
&\mapsto \sum_\rho \frac{\mathbf{q}(R\mathbf{x}_i + t, \rho)\rho(R)^\dagger \rho(R)\mathbf{k}(R\mathbf{x}_i + t, R\mathbf{x}_j + t, \rho)}{\sqrt{d_k}} \\
&= \sum_\rho \frac{\mathbf{q}(R\mathbf{x}_i + t, \rho)\mathbf{k}(R\mathbf{x}_i + t, R\mathbf{x}_j + t, \rho)}{\sqrt{d_k}} \\
&= s(R\mathbf{x}_i + t, R\mathbf{x}_j + t)
\end{aligned}
$$

The first equality follows from the fact that since $C = 1$, the *vec* operation is no longer needed. Since, α is obtained by taking the softmax of s, it follows that.

$$
\alpha(\mathbf{x}_i, \mathbf{x}_j) \mapsto \alpha(R\mathbf{x}_i + t, R\mathbf{x}_j + t).
$$

Finally,

$$
\begin{aligned}
f^{\text{out}}(\mathbf{x}_i, \rho) &\mapsto \sum_j \alpha(R\mathbf{x}_i + t, R\mathbf{x}_j + t)\rho(R)\mathbf{v}(R\mathbf{x}_i + t, R\mathbf{x}_j + t, \rho) \\
&= \rho(R) \sum_j \alpha(R\mathbf{x}_i + t, R\mathbf{x}_j + t)\mathbf{v}(R\mathbf{x}_i + t, R\mathbf{x}_j + t, \rho) \\
&= \rho(R) f^{\text{out}}(R\mathbf{x}_i + t, \rho).
\end{aligned}
$$

This completes the proof. □

C. Experiment Details

Here we provide the details of all the experiments we have conducted. Table 2 shows detailed results on the two datasets.

Datasets	Model	Radial Resolution	Frequncy Cutoff	Accuracy	Parameter ($\sim \times 10^6$)
Rotated MNIST	Steerable Convolution	$r = 2$	$k = 4$	$98.69_{\pm 0.05}$	1.00
			$k = 8$	$98.93_{\pm 0.02}$	1.18
		$r = 4$	$k = 4$	$98.72_{\pm 0.02}$	1.18
			$k = 8$	$98.97_{\pm 0.01}$	2.54
	Steerable Transformer	$r = 2$	$k = 4$	$98.78_{\pm 0.08}$	0.91
			$k = 8$	$\mathbf{99.03}_{\pm 0.04}$	1.78
		$r = 4$	$k = 4$	$98.82_{\pm 0.04}$	1.13
			$k = 8$	$\mathbf{99.03}_{\pm 0.06}$	2.24
ModelNet10 (z Rotation) ($\mathrm{SO}(3)$ Rotation)	Steerable Convolution	$r = 1$	$\ell = 4$	$89.40_{\pm 0.53}$	0.90
				$\mathbf{86.78}_{\pm 0.62}$	
		$r = 2$	$\ell = 4$	$90.13_{\pm 0.52}$	1.08
				$86.62_{\pm 0.25}$	
	Steerable Transformer	$r = 1$	$\ell = 4$	$\mathbf{89.52}_{\pm 0.55}$	0.90
				$85.87_{\pm 0.74}$	
		$r = 2$	$\ell = 4$	$\mathbf{90.40}_{\pm 0.25}$	0.92
				$\mathbf{86.80}_{\pm 0.58}$	

Table 2: Comparison of Steerable convolution and steerable transformers. The mean and sd are provided for 5 separate runs

C.1. Rotated MNIST

Type	Method	error%
Miscellaneous	SVM (Larochelle et al., 2007)	11.11
	TIRBM (Sohn & Lee, 2012)	4.2
	TI-Pooling (Laptev et al., 2016)	1.2
Equivariant Convolution	P4CNN (Cohen & Welling, 2016)	2.28
	Harmonic Net (Worrall et al., 2017)	1.69
	RotEqNet (Marcos et al., 2017)	1.09
	Weiler et al. (Weiler et al., 2018b)	0.71
	E2CNN (Weiler & Cesa, 2019)	**0.69**
Equivariant Attention	α-R4 CNN (Romero et al., 2020)	1.69
	GSA-Nets (Romero & Cordonnier, 2021)	2.03
	GE-ViT (Xu et al., 2023)	1.99
	Our ($k = 4$)	1.18
	Our ($k = 8$)	0.97

Table 3: Comparison of the performance of the equivariant attentive architecture on the Rotated MNIST dataset.

The network architecture in this experiment consists of three convolutional blocks with progressively increasing numbers of channels. These blocks are separated by average pooling layers, which down-sample the feature maps to improve computational efficiency. The output is flattened and passed through fully connected layers that apply linear transformations, batch normalization, ReLU activation, and dropout to prevent overfitting. The final layer maps the output to ten classes, making it suitable for classification tasks. In the case of the steerable transformer, a transformer block with a single layer follows the convolutional blocks.

The networks were trained using the Adam optimizer (Kingma & Ba, 2014), starting with a learning rate of 5×10^{-3}, which was reduced by a factor of 0.5 every 20 epochs, along with a weight decay of 5×10^{-4} for 150 epochs. Additional

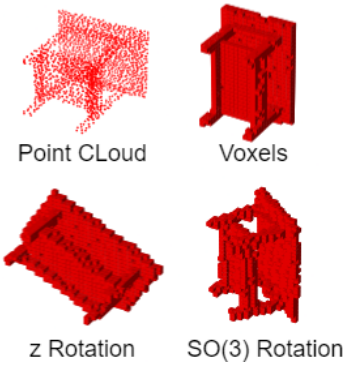

Figure 5: Examples from the ModelNet10 dataset are shown in various formats: point cloud, voxel representation, and rotated perturbations of voxels.

	Method	Accuracy	Parameters ($\sim \times 10^6$)
Point Cloud	ECC (Simonovsky & Komodakis, 2017)	90.8	-
	SO-Net (Li et al., 2018)	93.9	2.5
	Rot-SO-Net (Li et al., 2019)	94.5	2.5
Voxel	3D ShapeNets (Wu et al., 2015)	83.5	12
	VRN (Brock et al., 2016)	91.3	18
	VoxNet (Maturana & Scherer, 2015)	92.0	0.92
	FusionNet (Hegde & Zadeh, 2016)	93.1	120
	ORION (Sedaghat et al., 2016)	93.8	0.91
	Cubenet (Worrall & Brostow, 2018)	**94.6**	4.5
	Our	91.1	0.92

Table 4: Comparison of performance on ModelNet10 with z rotation perturbation. Other methods used both train and test time augmentation, while we applied augmentation only during test time, not during training.

experiments, where the radial resolution and frequency cutoff were varied, are shown in Table 2. A batch size of 25 was used, and training the largest model took 4 hours on a 16GB GPU.

Table 3 compares the performance of the steerable transformers with other attention-based methods reported for this dataset. Our approach significantly outperforms the other attention-based methods, which are standalone, while ours is built on top of convolutions. The only methods that surpass our results, by Weiler et al. (2018b) and Weiler & Cesa (2019), use a Fourier cutoff of 16, while we use a cutoff of 8. We believe this accounts for the performance difference, which is due to implementation and resource limitations, as higher Fourier cutoffs lead to out-of-memory issues.

C.2. ModelNet10

The point cloud data with 2048 points can be accessed here. Similar to the Rotated MNIST experiment, we utilized three convolutional blocks. The features for each irrep are in a 1:1:1:1 ratio. The number of features per irrep increases with each convolutional block, and the data is downsampled using average pooling. For the steerable transformer, this convolutional encoder is supplemented with a transformer block containing a single transformer layer. After an additional normalization step, the output is flattened and passed through fully connected layers that reduce the dimensionality to 128 features. To ensure training stability and prevent overfitting, batch normalization, ReLU activation, and dropout are applied, with the final output layer classifying the data into ten classes.

The networks were trained using the Adam optimizer (Kingma & Ba, 2014), with an initial learning rate of 1×10^{-3}, which decreased by a factor of 0.5 every 20 epochs, for a total of 50 epochs. Detailed experiments exploring variations in radial resolution are presented in Table 2. A batch size of 5 was used, and training the largest model took 12 hours on a 16GB GPU.

Table 4 compares our method to other non-attention-based methods on the z-rotated version of the dataset. Unlike our approach, which does not use augmentation during training, all other methods were trained with augmentation involving 12 uniformly stratified rotations along the z-axis. To ensure a fair comparison, we applied test-time augmentation by averaging the accuracy of all 12 predictions for each test data point. Despite not using train-time augmentation and having fewer parameters, our method achieves performance comparable to these other methods.

C.3. PH2

The data for the experiment is available here. The network begins with two convolutional stem blocks, each composed of steerable convolutional layers, non-linearities, and batch normalization, which progressively downsample the input and extract relevant features. A transformer encoder is used to capture global dependencies in the data. After encoding, the model utilizes two convolutional head blocks to upsample and refine the extracted features using steerable convolutions.

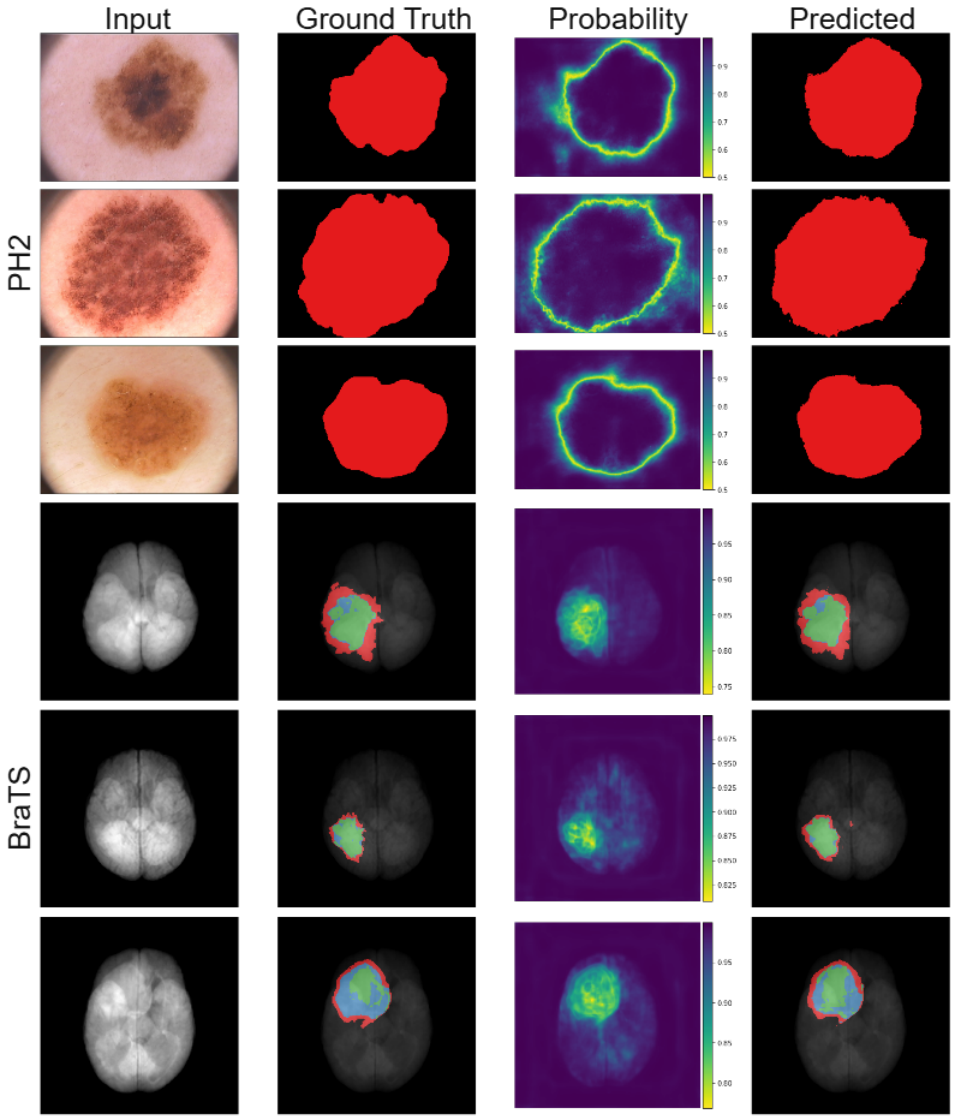

Figure 6: More segmented examples from the test datasets of PH2 and BraTS.

The output is then processed through complex-valued bilinear interpolation to restore the original input dimensions. In the final step, the vector norm is computed, converting the complex-valued feature maps into real values.

The networks were trained using the Adam optimizer (Kingma & Ba, 2014), starting with an initial learning rate of 1×10^{-2}, which decreased by a factor of 0.5 every 20 epochs, for a total of 100 epochs. We used a batch size of 1, and training the largest model took 2 hours on a 16GB GPU. A larger batch size could not be used due to out-of-memory errors.

C.4. BraTS

The data for the experiment is available here. Only the training data is available for this task, consisting of 484 images. We split this dataset into training, validation, and test sets for our experiments.

The network begins with two convolutional stem blocks that use steerable convolutions to extract multi-resolution features. These blocks include non-linearities and batch normalization for stable training, as well as average pooling layers for downsampling. The features for each irrep are in a ratio of 8:4:2, with the number of features increasing in subsequent layers according to this ratio. The encoded features are then passed through a transformer encoder, which captures

long-range dependencies and spatial relationships using multiple layers and positional encodings. The decoder consists of convolutional head blocks that upsample the features. Complex-valued feature maps are interpolated using trilinear interpolation, and the final output is generated by taking the absolute value of these complex features, corresponding to the constant representation.

The networks were trained using the Adam optimizer (Kingma & Ba, 2014), with an initial learning rate of 1×10^{-2}, which was reduced by a factor of 0.5 every 20 epochs, for a total of 100 epochs. A batch size of 1 was used, and training the largest model took 40 hours on a 16GB GPU. A larger batch size could not be used due to out-of-memory errors.