GAUGE EQUIVARIANT MESH CNNS
ANISOTROPIC CONVOLUTIONS ON GEOMETRIC GRAPHS

Pim de Haan[*]
Qualcomm AI Research[†]
University of Amsterdam

Maurice Weiler[*]
QUVA Lab
University of Amsterdam

Taco Cohen
Qualcomm AI Research

Max Welling
Qualcomm AI Research
University of Amsterdam

ABSTRACT

A common approach to define convolutions on meshes is to interpret them as a graph and apply graph convolutional networks (GCNs). Such GCNs utilize *isotropic* kernels and are therefore insensitive to the relative orientation of vertices and thus to the geometry of the mesh as a whole. We propose Gauge Equivariant Mesh CNNs which generalize GCNs to apply *anisotropic* gauge equivariant kernels. Since the resulting features carry orientation information, we introduce a geometric message passing scheme defined by parallel transporting features over mesh edges. Our experiments validate the significantly improved expressivity of the proposed model over conventional GCNs and other methods.

1 INTRODUCTION

Convolutional neural networks (CNNs) have been established as the default method for many machine learning tasks like speech recognition or planar and volumetric image classification and segmentation. Most CNNs are restricted to flat or spherical geometries, where convolutions are easily defined and optimized implementations are available. The empirical success of CNNs on such spaces has generated interest to generalize convolutions to more general spaces like graphs or Riemannian manifolds, creating a field now known as geometric deep learning (Bronstein et al., 2017).

A case of specific interest is convolution on *meshes*, the discrete analog of 2-dimensional embedded Riemannian manifolds. Mesh CNNs can be applied to tasks such as detecting shapes, registering different poses of the same shape and shape segmentation. If we forget the positions of vertices, and which vertices form faces, a mesh M can be represented by a graph $\mathcal{G}$. This allows for the application of *graph convolutional networks* (GCNs) to processing signals on meshes.

[*]Equal Contribution
[†]Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

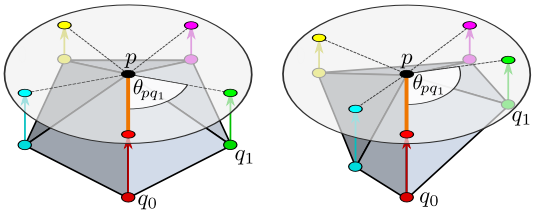


Figure 1: Two local neighbourhoods around vertices p and their representations in the tangent planes T_pM. The distinct geometry of the neighbourhoods is reflected in the different angles θ_{pq_i} of incident edges from neighbours q_i. Graph convolutional networks apply isotropic kernels and can therefore not distinguish both neighbourhoods. Gauge Equivariant Mesh CNNs apply anisotropic kernels and are therefore sensitive to orientations. The arbitrariness of reference orientations, determined by a choice of neighbour q_0, is accounted for by the gauge equivariance of the model.

However, when representing a mesh by a graph, we lose important geometrical information. In particular, in a graph there is no notion of angle between or ordering of two of a node's incident edges (see figure 1). Hence, a GCNs output at a node p is designed to be independent of relative angles and *invariant* to any permutation of its neighbours $q_i \in \mathcal{N}(p)$. A graph convolution on a mesh graph therefore corresponds to applying an *isotropic* convolution kernel. Isotropic filters are insensitive to the orientation of input patterns, so their features are strictly less expressive than those of orientation aware anisotropic filters.

To address this limitation of graph networks we propose *Gauge Equivariant Mesh CNNs* (GEM-CNN), which minimally modify GCNs such that they are able to use anisotropic filters while sharing weights across different positions and respecting the local geometry. One obstacle in sharing anisotropic kernels, which are functions of the angle θ_{pq} of neighbour q with respect to vertex p, over multiple vertices of a mesh is that there is no unique way of selecting a reference neighbour q_0, which has the direction $\theta_{pq_0} = 0$. The reference neighbour, and hence the orientation of the neighbours, needs to be chosen arbitrarily. In order to guarantee the equivalence of the features resulting from different choices of orientations, we adapt Gauge Equivariant CNNs (Cohen et al., 2019b) to general meshes. The kernels of our model are thus designed to be *equivariant under gauge transformations*, that is, to guarantee that the responses for different kernel orientations are related by a prespecified transformation law. Such features are identified as geometric objects like scalars, vectors, tensors, etc., depending on the specific choice of transformation law. In order to compare such geometric features at neighbouring vertices, they need to be *parallel transported* along the connecting edge.

In our implementation we first specify the transformation laws of the feature spaces and compute a space of gauge equivariant kernels. Then we pick arbitrary reference orientations at each node, relative to which we compute neighbour orientations and compute the corresponding edge transporters. Given these quantities, we define the forward pass as a message passing step via edge transporters followed by a contraction with the equivariant kernels evaluated at the neighbour orientations. Algorithmically, Gauge Equivariant Mesh CNNs are therefore just GCNs with anisotropic, gauge equivariant kernels and message passing via parallel transporters. Conventional GCNs are covered in this framework for the specific choice of isotropic kernels and trivial edge transporters, given by identity maps.

In Sec. 2, we will give an outline of our method, deferring details to Secs. 3 and 4. In Sec. 3.2, we describe how to compute general geometric quantities, not specific to our method, used for the computation of the convolution. In our experiments in Sec. 6.1, we find that the enhanced expressiveness of Gauge Equivariant Mesh CNNs enables them to outperform conventional GCNs and other prior work in a shape correspondence task.

2 Convolutions on Graphs with Geometry

We consider the problem of processing signals on discrete 2-dimensional manifolds, or meshes M. Such meshes are described by a set $\mathcal{V}$ of vertices in $\mathbb{R}^3$ together with a set $\mathcal{F}$ of tuples, each consisting of the vertices at the corners of a face. For a mesh to describe a proper manifold, each edge needs to be connected to two faces, and the neighbourhood of each vertex needs to be homeomorphic to a disk. Mesh M induces a graph $\mathcal{G}$ by forgetting the coordinates of the vertices while preserving the edges.

A conventional graph convolution between kernel K and signal f, evaluated at a vertex p, can be defined by

$$(K \star f)_p \;=\; K_{\text{self}} f_p \;+ \sum\nolimits_{q \in \mathcal{N}_p} K_{\text{neigh}} f_q, \tag{1}$$

where $\mathcal{N}_p$ is the set of neighbours of p in $\mathcal{G}$, and $K_{\text{self}} \in \mathbb{R}^{C_{\text{in}} \times C_{\text{out}}}$ and $K_{\text{neigh}} \in \mathbb{R}^{C_{\text{in}} \times C_{\text{out}}}$ are two linear maps which model a self interaction and the neighbour contribution, respectively. Importantly, graph convolution does not distinguish different neighbours, because each feature vector f_q is multiplied by the same matrix K_{neigh} and then summed. For this reason we say the kernel is *isotropic*.

Consider the example in figure 1, where on the left and right, the neighbourhood of one vertex p, containing neighbours $q \in \mathcal{N}_p$, is visualized. An isotropic kernel would propagate the signal from the neighbours to p in exactly the same way in both neighbourhoods, even though the neighbourhoods are geometrically distinct. For this reason, our method uses direction sensitive (*anisotropic*) kernels instead of isotropic kernels. Anisotropic kernels are inherently more expressive than isotropic ones which is why they are used universally in conventional planar CNNs.

Algorithm 1 Gauge Equivariant Mesh CNN layer

Input: mesh M, input/output feature types $\rho_{\text{in}}, \rho_{\text{out}}$, reference neighbours $(q_0^p \in \mathcal{N}_p)_{p\in M}$.

Compute basis kernels $K^i_{\text{self}}, K^i_{\text{neigh}}(\theta)$ ▷ Sec. 3

Initialise weights w^i_{self} and w^i_{neigh}.

For each neighbour pair, $p \in M, q \in \mathcal{N}_p$: ▷ App. A.
    compute neighbor angles θ_{pq} relative to reference neighbor
    compute parallel transporters $g_{q\to p}$

Forward$\big($input features $(f_p)_{p\in M}$, weights $w^i_{\text{self}}, w^i_{\text{neigh}}\big)$:

$f'_p \leftarrow \sum_i w^i_{\text{self}} K^i_{\text{self}} f_p + \sum_{i,q\in\mathcal{N}_p} w^i_{\text{neigh}} K^i_{\text{neigh}}(\theta_{pq})\rho_{\text{in}}(g_{q\to p}) f_q$

We propose the Gauge Equivariant Mesh Convolution, a minimal modification of graph convolution that allows for anisotropic kernels $K(\theta)$ whose value depends on an orientation $\theta \in [0, 2\pi)$.[1] To define the orientations θ_{pq} of neighbouring vertices $q \in \mathcal{N}_p$ of p, we first map them to the tangent plane T_pM at p, as visualized in figure 1. We then pick an *arbitrary* reference neighbour q_0^p to determine a reference orientation[2] $\theta_{pq_0^p} := 0$, marked orange in figure 1. This induces a basis on the tangent plane, which, when expressed in polar coordinates, defines the angles θ_{pq} of the other neighbours.

As we will motivate in the next section, features in a Gauge Equivariant CNN are coefficients of geometric quantities. For example, a tangent vector at vertex p can be described either geometrically by a 3 dimensional vector orthogonal to the normal at p or by two coefficients in the basis on the tangent plane. In order to perform convolution, geometric features at different vertices need to be linearly combined, for which it is required to first "parallel transport" the features to the same vertex. This is done by applying a matrix $\rho(g_{q\to p}) \in \mathbb{R}^{C_{\text{in}}\times C_{\text{in}}}$ to the coefficients of the feature at q, in order to obtain the coefficients of the feature vector transported to p, which can be used for the convolution at p. The transporter depends on the geometric *type* (group representation) of the feature, denoted by ρ and described in more detail below. Details of how the tangent space is defined, how to compute the map to the tangent space, angles θ_{pq}, and the parallel transporter are given in Appendix A.

In combination, this leads to the GEM-CNN convolution

$$(K \star f)_p \;=\; K_{\text{self}} f_p + \sum\nolimits_{q\in\mathcal{N}_p} K_{\text{neigh}}(\theta_{pq})\rho(g_{q\to p}) f_q \tag{2}$$

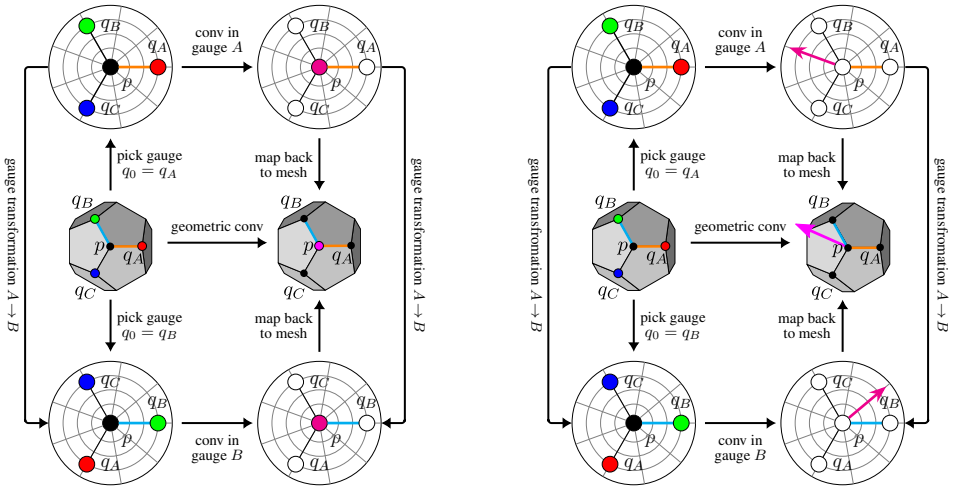
which differs from the conventional graph convolution, defined in Eq. 1 only by the use of an anisotropic kernel and the parallel transport message passing.

We require the outcome of the convolution to be *equivalent* for any choice of reference orientation. This is not the case for any anisotropic kernel but only for those which are *equivariant under changes of reference orientations* (gauge transformations). Equivariance imposes a linear constraint on the kernels. We therefore solve for complete sets of "basis-kernels" K^i_{self} and K^i_{neigh} satisfying this constraint and linearly combine them with parameters w^i_{self} and w^i_{neigh} such that $K_{\text{self}} = \sum_i w^i_{\text{self}} K^i_{\text{self}}$ and $K_{\text{neigh}} = \sum_i w^i_{\text{neigh}} K^i_{\text{neigh}}$. Details on the computation of basis kernels are given in section 3. The full algorithm for initialisation and forward pass, which is of time and space complexity linear in the number of vertices, for a GEM-CNN layer are listed in algorithm 1. Gradients can be computed by automatic differentiation.

The GEM-CNN is gauge equivariant, but furthermore satisfies two important properties. Firstly, it depends only on the intrinsic shape of the 2D mesh, not on the embedding of the mesh in $\mathbb{R}^3$. Secondly, whenever a map from the mesh to itself exists that preserves distances and orientation, the convolution is equivariant to moving the signal along such transformations. These properties are proven in Appendix D and empirically shown in Appendix F.2.

[1] In principle, the kernel could be made dependent on the radial distance of neighboring nodes, by $K_{\text{neigh}}(r, \theta) = F(r)K_{\text{neigh}}(\theta)$, where $F(r)$ is unconstrained and $K_{\text{neigh}}(\theta)$ as presented in this paper. As this dependency did not improve the performance in our empirical evaluation, we omit it.

[2] Mathematically, this corresponds to a choice of *local reference frame* or *gauge*.



(a) Convolution from scalar to scalar features.

(b) Convolution from scalar to vector features.

Figure 2: Visualization of the Gauge Equivariant Mesh Convolution in two configurations, scalar to scalar and scalar to vector. The convolution operates in a gauge, so that vectors are expressed in coefficients in a basis and neighbours have polar coordinates, but can also be seen as a *geometric convolution*, a gauge-independent map from an input signal on the mesh to a output signal on the mesh. The convolution is equivariant if this geometric convolution does not depend on the intermediate chosen gauge, so if the diagram commutes.

3 Gauge Equivariance & Geometric Features

On a general mesh, the choice of the reference neighbour, or gauge, which defines the orientation of the kernel, can only be made arbitrarily. However, this choice should not arbitrarily affect the outcome of the convolution, as this would impede the generalization between different locations and different meshes. Instead, Gauge Equivariant Mesh CNNs have the property that their output transforms according to a known rule as the gauge changes.

Consider the left hand side of figure 2(a). Given a neighbourhood of vertex p, we want to express each neighbour q in terms of its polar coordinates (r_q, θ_q) on the tangent plane, so that the kernel value at that neighbour $K_{\text{neigh}}(\theta_q)$ is well defined. This requires choosing a basis on the tangent plane, determined by picking a neighbour as reference neighbour (denoted q_0), which has the zero angle $\theta_{q_0} = 0$. In the top path, we pick q_A as reference neighbour. Let us call this gauge A, in which neighbours have angles θ_q^A. In the bottom path, we instead pick neighbour q_B as reference point and are in gauge B. We get a different basis for the tangent plane and different angles θ_q^B for each neighbour. Comparing the two gauges, we see that they are related by a rotation, so that $\theta_q^B = \theta_q^A - \theta_{q_B}^A$. This change of gauge is called a gauge transformation of angle $g := \theta_{q_B}^A$.

In figure 2(a), we illustrate a gauge equivariant convolution that takes input and output features such as gray scale image values on the mesh, which are called scalar features. The top path represents the convolution in gauge A, the bottom path in gauge B. In either case, the convolution can be interpreted as consisting of three steps. First, for each vertex p, the value of the scalar features on the mesh at each neighbouring vertex q, represented by colors, is mapped to the tangent plane at p at angle θ_q defined by the gauge. Subsequently, the convolutional kernel sums for each neighbour q, the product of the feature at q and kernel $K(\theta_q)$. Finally the output is mapped back to the mesh. These three steps can be composed into a single step, which we could call a *geometric convolution*, mapping from input features on the mesh to output features on the mesh. The convolution is *gauge equivariant* if this geometric convolution does not depend on the gauge we pick in the interim, so in figure 2(a), if the convolution in the top path in gauge A has same result the convolution in the bottom path in gauge B, making the diagram commute. In this case, however, we see that the convolution output needs to be the same in both gauges, for the convolution to be equivariant. Hence, we must have that $K(\theta_q) = K(\theta_q - g)$, as the orientations of the neighbours differ by some angle g, and the kernel must be isotropic.

As we aim to design an anisotropic convolution, the output feature of the convolution at p can, instead of a scalar, be two numbers $v \in \mathbb{R}^2$, which can be interpreted as coefficients of a tangent feature

vector in the tangent space at p, visualized in figure 2(b). As shown on the right hand side, different gauges induce a different basis of the tangent plane, so that the *same tangent vector* (shown on the middle right on the mesh), is represented by *different coefficients* in the gauge (shown on the top and bottom on the right). This gauge equivariant convolution must be anisotropic: going from the top row to the bottom row, if we change orientations of the neighbours by $-g$, the coefficients of the output vector $v \in \mathbb{R}^2$ of the kernel must be also rotated by $-g$. This is written as $R(-g)v$, where $R(-g) \in \mathbb{R}^{2\times 2}$ is the matrix that rotates by angle $-g$.

Vectors and scalars are not the only type of geometric features that can be inputs and outputs of a GEM-CNN layer. In general, the coefficients of a geometric feature of C dimensions changes by an invertible linear transformation $\rho(-g) \in \mathbb{R}^{C\times C}$ if the gauge is rotated by angle g. The map $\rho : [0, 2\pi) \to \mathbb{R}^{C\times C}$ is called the *type* of the geometric quantity and is formally known as a group representation of the planar rotation group $\mathrm{SO}(2)$. Group representations have the property that $\rho(g + h) = \rho(g)\rho(h)$ (they are group homomorphisms), which implies in particular that $\rho(0) = \mathbb{1}$ and $\rho(-g) = \rho(g)^{-1}$. For more background on group representation theory, we refer the reader to (Serre, 1977) and, specifically in the context of equivariant deep learning, to (Lang & Weiler, 2020). From the theory of group representations, we know that any feature type can be composed from "irreducible representations" (irreps). For $\mathrm{SO}(2)$, these are the one dimensional invariant scalar representation ρ_0 and for all $n \in \mathbb{N}_{>0}$, a two dimensional representation ρ_n,

$$\rho_0(g) = 1, \quad \rho_n(g) = \begin{pmatrix} \cos ng & -\sin ng \\ \sin ng & \cos ng \end{pmatrix}.$$

where we write, for example, $\rho = \rho_0 \oplus \rho_1 \oplus \rho_1$ to denote that representation $\rho(g)$ is the direct sum (i.e. block-diagonal stacking) of the matrices $\rho_0(g), \rho_1(g), \rho_1(g)$. Scalars and tangent vector features correspond to ρ_0 and ρ_1 respectively and we have $R(g) = \rho_1(g)$.

The type of the feature at each layer in the network can thus be fully specified (up to a change of basis) by the number of copies of each irrep. Similar to the dimensionality in a conventional CNN, the choice of type is a hyperparameter that can be freely chosen to optimize performance.

3.1 Kernel Constraint

Given an input type ρ_{in} and output type ρ_{out} of dimensions C_{in} and C_{out}, the kernels are $K_{\text{self}} \in \mathbb{R}^{C_{\text{out}}\times C_{\text{in}}}$ and $K_{\text{neigh}} : [0, 2\pi) \to \mathbb{R}^{C_{\text{out}}\times C_{\text{in}}}$. However, not all such kernels are equivariant. Consider again examples figure 2(a) and figure 2(b). If we map from a scalar to a scalar, we get that $K_{\text{neigh}}(\theta - g) = K_{\text{neigh}}(\theta)$ for all angles θ, g and the convolution is isotropic. If we map from a scalar to a vector, we get that rotating the angles θ_q results in the same tangent vector as rotating the output vector coefficients, so that $K_{\text{neigh}}(\theta - g) = R(-g)K_{\text{neigh}}(\theta)$.

In general, as derived by Cohen et al. (2019b) and in appendix B, the kernels must satisfy for any gauge transformation $g \in [0, 2\pi)$ and angle $\theta \in [0, 2\pi)$, that

$$K_{\text{neigh}}(\theta - g) = \rho_{\text{out}}(-g)K_{\text{neigh}}(\theta)\rho_{\text{in}}(g), \quad (3)$$

$$K_{\text{self}} = \rho_{\text{out}}(-g)\ K_{\text{self}}\ \rho_{\text{in}}(g). \quad (4)$$

The kernel can be seen as consisting of multiple blocks, where each block takes as input one irrep and outputs one irrep. For example if ρ_{in} would be of type $\rho_0 \oplus \rho_1 \oplus \rho_1$ and ρ_{out} of type $\rho_1 \oplus \rho_3$, we have 4×5 matrix

$$K_{\text{neigh}}(\theta) = \begin{pmatrix} K_{10}(\theta) & K_{11}(\theta) & K_{11}(\theta) \\ K_{30}(\theta) & K_{31}(\theta) & K_{31}(\theta) \end{pmatrix}$$

$\rho_{\text{in}} \to \rho_{\text{out}}$	linearly independent solutions for $K_{\text{neigh}}(\theta)$
$\rho_0 \to \rho_0$	(1)
$\rho_n \to \rho_0$	$(\cos n\theta\ \ \sin n\theta), (\sin n\theta\ \ -\cos n\theta)$
$\rho_0 \to \rho_m$	$\begin{pmatrix}\cos m\theta \\ \sin m\theta\end{pmatrix}, \begin{pmatrix}\sin m\theta \\ -\cos m\theta\end{pmatrix}$
$\rho_n \to \rho_m$	$\begin{pmatrix}c_- & -s_- \\ s_- & c_-\end{pmatrix}, \begin{pmatrix}s_- & c_- \\ -c_- & s_-\end{pmatrix}, \begin{pmatrix}c_+ & s_+ \\ s_+ & -c_+\end{pmatrix}, \begin{pmatrix}-s_+ & c_+ \\ c_+ & s_+\end{pmatrix}$
$\rho_{\text{in}} \to \rho_{\text{out}}$	linearly independent solutions for K_{self}
$\rho_0 \to \rho_0$	(1)
$\rho_n \to \rho_n$	$\begin{pmatrix}1 & 0 \\ 0 & 1\end{pmatrix}, \begin{pmatrix}0 & 1 \\ -1 & 0\end{pmatrix}$

Table 1: Solutions to the angular kernel constraint for kernels that map from ρ_n to ρ_m. We denote $c_\pm = \cos((m \pm n)\theta)$ and $s_\pm = \sin((m \pm n)\theta)$.

where e.g. $K_{31}(\theta) \in \mathbb{R}^{2\times 2}$ is a kernel that takes as input irrep ρ_1 and as output irrep ρ_3 and needs to satisfy Eq. 3. As derived by Weiler & Cesa (2019) and in Appendix C, the kernels $K_{\text{neigh}}(\theta)$ and K_{self} mapping from irrep ρ_n to irrep ρ_m can be written as a linear combination of the basis kernels listed in Table 1. The table shows that equivariance requires the self-interaction to only map from one irrep to the same irrep. Hence, we have $K_{\text{self}} = \begin{pmatrix} 0 & K_{11} & K_{11} \\ 0 & 0 & 0 \end{pmatrix} \in \mathbb{R}^{4\times 3}$.

All basis-kernels of all pairs of input irreps and output irreps can be linearly combined to form an arbitrary equivariant kernel from feature of type ρ_{in} to ρ_{out}. In the above example, we have $2 \times 2 + 4 \times 4 = 20$ basis kernels for K_{neigh} and 4 basis kernels for K_{self}. The layer thus has 24 parameters. As proven in (Weiler & Cesa, 2019) and (Lang & Weiler, 2020), this parameterization of the equivariant kernel space is *complete*, that is, more general equivariant kernels do not exist.

3.2 Geometry and Parallel Transport

In order to implement gauge equivariant mesh CNNs, we need to make the abstract notion of tangent spaces, gauges and transporters concrete.

As the mesh is embedded in $\mathbb{R}^3$, a natural definition of the tangent spaces T_pM is as two dimensional subspaces that are orthogonal to the normal vector at p. We follow the common definition of normal vectors at mesh vertices as the area weighted average of the adjacent faces' normals. The Riemannian logarithm map $\log_p : \mathcal{N}_p \to T_pM$ represents the one-ring neighborhood of each point p on their tangent spaces as visualized in figure 1. Specifically, neighbors $q \in \mathcal{N}_p$ are mapped to $\log_p(q) \in T_pM$ by first projecting them to T_pM and then rescaling the projection such that the norm is preserved, i.e. $|\log_p(q)| = |q - p|$; see Eq. 6. A choice of reference neighbor $q_p \in \mathcal{N}$ uniquely determines a right handed, orthonormal reference frame $(e_{p,1},\, e_{p,2})$ of T_pM by setting $e_{p,1} := \log_p(q_0)/|\log_p(q_0)|$ and $e_{p,2} := n \times e_{p,1}$. The polar angle θ_{pq} of any neighbor $q \in \mathcal{N}$ relative to the first frame axis is then given by $\theta_{pq} := \operatorname{atan2}\left(e_{p,2}^\top \log_p(q),\; e_{p,1}^\top \log_p(q)\right)$.

Given the reference frame $(e_{p,1}, e_{p,2})$, a 2-tuple of coefficients $(v_1, v_2) \in \mathbb{R}^2$ specifies an (embedded) tangent vector $v_1 e_{p,1} + v_2 e_{p,2} \in T_pM \subset \mathbb{R}^3$. This assignment is formally given by the *gauge map* $E_p : \mathbb{R}^2 \to T_pM \subset \mathbb{R}^3$ which is a vector space isomorphism. In our case, it can be identified with the matrix

$$E_p = \begin{bmatrix} | & | \\ e_{p,1} & e_{p,2} \\ | & | \end{bmatrix} \in \mathbb{R}^{3\times 2}. \tag{5}$$

Feature vectors f_p and f_q at neighboring (or any other) vertices $p \in M$ and $q \in \mathcal{N}_p \subseteq M$ live in different vector spaces and are expressed relative to independent gauges, which makes it invalid to sum them directly. Instead, they have to be parallel transported along the mesh edge that connects the two vertices. As explained above, this transport is given by group elements $g_{q\to p} \in [0, 2\pi)$, which determine the transformation of tangent vector *coefficients* as $v_q \mapsto R(g_{q\to p})v_q \in \mathbb{R}^2$ and, analogously, for feature vector coefficients as $f_q \mapsto \rho(g_{q\to p})f_q$. Figure 4 in the appendix visualizes the definition of edge transporters for flat spaces and meshes. On a flat space, tangent vectors are transported by keeping them parallel in the usual sense on Euclidean spaces. However, if the source and target frame orientations disagree, the vector coefficients relative to the source frame need to be transformed to the target frame. This coordinate transformation from polar angles φ_q of v to φ_p of $R(g_{q\to p})v$ defines the transporter $g_{q\to p} = \varphi_p - \varphi_q$. On meshes, the source and target tangent spaces T_qM and T_pM are not longer parallel. It is therefore additionally necessary to rotate the source tangent space and its vectors parallel to the target space, before transforming between the frames. Since transporters effectively make up for differences in the source and target frames, the parallel transporters transform under gauge transformations g_p and g_q according to $g_{q\to p} \mapsto g_p + g_{q\to p} - g_q$. Note that this transformation law cancels with the transformation law of the coefficients at q and lets the transported coefficients transform according to gauge transformations at p. It is therefore valid to sum vectors and features that are parallel transported into the same gauge at p.

A more detailed discussion of the concepts presented in this section can be found in Appendix A.

4 Non-linearity

Besides convolutional layers, the GEM-CNN contains non-linear layers, which also need to be gauge equivariant, for the entire network to be gauge equivariant. The coefficients of features built out of irreducible representaions, as described in section 3, do not commute with point-wise non-linearities (Worrall et al., 2017; Thomas et al., 2018; Weiler et al., 2018a; Kondor et al., 2018). Norm non-linearities and gated non-linearities (Weiler & Cesa, 2019) can be used with such features, but generally perform worse in practice compared to point-wise non-linearities (Weiler & Cesa,

2019). Hence, we propose the *RegularNonlinearity*, which uses point-wise non-linearities and is approximately gauge equivariant.

This non-linearity is built on Fourier transformations. Consider a continuous periodic signal, on which we perform a band-limited Fourier transform with band limit b, obtaining $2b + 1$ Fourier coefficients. If this continuous signal is shifted by an arbitrary angle g, then the corresponding Fourier components transform with linear transformation $\rho_{0:b}(-g)$, for $2b + 1$ dimensional representation $\rho_{0:b} := \rho_0 \oplus \rho_1 \oplus ... \oplus \rho_b$.

It would be exactly equivariant to take a feature of type $\rho_{0:b}$, take a continuous inverse Fourier transform to a continuous periodic signal, then apply a point-wise non-linearity to that signal, and take the continuous Fourier transform, to recover a feature of type $\rho_{0:b}$. However, for implementation, we use N intermediate samples and the discrete Fourier transform. This is exactly gauge equivariant for gauge transformation of angles multiple of $2\pi/N$, but only approximately equivariant for other angles. In App. G we prove that as $N \to \infty$, the non-linearity is exactly gauge equivariant.

The run-time cost per vertex of the (inverse) Fourier transform implemented as a simple linear transformation is $\mathcal{O}(bN)$, which is what we use in our experiments. The pointwise non-linearity scales linearly with N, so the complexity of the RegularNonLineariy is also $\mathcal{O}(bN)$. However, one can also use a fast Fourier transform, achieving a complexity of $\mathcal{O}(N \log N)$. Concrete memory and run-time cost of varying N are shown in appendix F.1.

5 Related Work

The irregular structure of meshes leads to a variety of approaches to define convolutions. Closely related to our method are graph based methods which are often based on variations of graph convolutional networks (Kipf & Welling, 2017; Defferrard et al., 2016). GCNs have been applied on spherical meshes (Perraudin et al., 2019) and cortical surfaces (Cucurull et al., 2018; Zhao et al., 2019a). Verma et al. (2018) augment GCNs with anisotropic kernels which are dynamically computed via an attention mechanism over graph neighbours.

Instead of operating on the graph underlying a mesh, several approaches leverage its geometry by treating it as a discrete manifold. Convolution kernels can then be defined in geodesic polar coordinates which corresponds to a projection of kernels from the tangent space to the mesh via the exponential map. This allows for kernels that are larger than the immediate graph neighbourhood and message passing over faces but does not resolve the issue of ambiguous kernel orientation. Masci et al. (2015); Monti et al. (2016) and Sun et al. (2018) address this issue by restricting the network to orientation invariant features which are computed by applying anisotropic kernels in several orientations and pooling over the resulting responses. The models proposed in (Boscaini et al., 2016) and (Schonsheck et al., 2018) are explicitly gauge dependent with preferred orientations chosen via the principal curvature direction and the parallel transport of kernels, respectively. Poulenard & Ovsjanikov (2018) proposed a non-trivially gauge equivariant network based on geodesic convolutions, however, the model parallel transports only partial information of the feature vectors, corresponding to certain kernel orientations. In concurrent work, Wiersma et al. (2020) also define convolutions on surfaces equivariantly to the orientation of the kernel, but differ in that they use norm non-linearities instead of regular ones and that they apply the convolution along longer geodesics, which adds complexity to the geometric pre-computation - as partial differential equations need to be solved, but may result in less susceptibility to the particular discretisation of the manifold.

Another class of approaches defines spectral convolutions on meshes. However, as argued in (Bronstein et al., 2017), the Fourier spectrum of a mesh depends heavily on its geometry, which makes such methods instable under deformations and impedes the generalization between different meshes. Spectral convolutions further correspond to isotropic kernels. Kostrikov et al. (2018) overcomes isotropy of the Laplacian by decomposing it into two applications of the first-order Dirac operator.

A construction based on toric covering maps of topologically spherical meshes was presented in (Maron et al., 2017). An entirely different approach to mesh convolutions is to apply a linear map to a spiral of neighbours (Bouritsas et al., 2019; Gong et al., 2019), which works well only for meshes with a similar graph structure.

The above-mentioned methods operate on the intrinsic, 2-dimensional geometry of the mesh. A popular alternative for embedded meshes is to define convolutions in the embedding space $\mathbb{R}^3$. This can for instance be done by voxelizing space and representing the mesh in terms of an occupancy grid (Wu et al., 2015; Tchapmi et al., 2017; Hanocka et al., 2018). A downside of this approach are the high memory and compute requirements of voxel representations. If the grid occupancy is low, this can partly be addressed by resorting to an inhomogeneous grid density (Riegler et al., 2017). Instead of voxelizing space, one may interpret the set of mesh vertices as a point cloud and run a convolution on those (Qi et al., 2017a;b). Point cloud based methods can be made equivariant w.r.t. the isometries of $\mathbb{R}^3$ (Zhao et al., 2019b; Thomas et al., 2018), which implies in particular the isometry equivariance on the embedded mesh. In general, geodesic distances within the manifold differ usually substantially from the distances in the embedding space. Which approach is more suitable depends on the particular application.

On flat Euclidean spaces our method corresponds to Steerable CNNs (Cohen & Welling, 2017; Weiler et al., 2018a; Weiler & Cesa, 2019; Cohen et al., 2019a; Lang & Weiler, 2020). As our model, these networks process geometric feature fields of types ρ and are equivariant under gauge transformations, however, due to the flat geometry, the parallel transporters become trivial. Regular nonlinearities are on flat spaces used in group convolutional networks (Cohen & Welling, 2016; Weiler et al., 2018b; Hoogeboom et al., 2018; Bekkers et al., 2018; Winkels & Cohen, 2018; Worrall & Brostow, 2018; Worrall & Welling, 2019; Sosnovik et al., 2020).

6 Experiments

6.1 Embedded MNIST

We first investigate how Gauge Equivariant Mesh CNNs perform on, and generalize between, different mesh geometries. For this purpose we conduct simple MNIST digit classification experiments on embedded rectangular meshes of 28×28 vertices. As a baseline geometry we consider a flat mesh as visualized in figure 5(a). A second type of geometry is defined as different *isometric* embeddings of the flat mesh, see figure 5(b). Note that this implies that the *intrinsic* geometry of these isometrically embedded meshes is indistinguishable from that of the flat mesh. To generate geometries which are intrinsically curved, we add random normal displacements to the flat mesh. We control the amount of curvature by smoothing the resulting displacement fields with Gaussian kernels of different widths σ and define the roughness of the resulting mesh as $3 - \sigma$. Figures 5(c)-5(h) show the results for roughnesses of 0.5, 1, 1.5, 2, 2.25 and 2.5. For each of the considered settings we generate 32 different train and 32 test geometries.

To test the performance on, and generalization between, different geometries, we train equivalent GEM-CNN models on a flat mesh and meshes with a roughness of 1, 1.5, 2, 2.25 and 2.5. Each model is tested individually on each of the considered test geometries, which are the flat mesh, isometric embeddings and curved embeddings with a roughness of 0.5, 1, 1.25, 1.5, 1.75, 2, 2.25 and 2.5. Figure 3 shows the test errors of the GEM-CNNs on the different train geometries (different curves) for all test geometries (shown on the x-axis). Since our model is purely defined in terms of the intrinsic geometry of a mesh, it is expected to be insensitive to isometric changes in the embeddings. This is empirically confirmed by the fact that the test performances on flat and isometric embeddings are exactly equal. As expected, the test error increases for most models with the surface roughness. Models trained on more rough surfaces are hereby more robust to deformations. The models generalize well from a rough training to smooth test geometry up to a training roughness of 1.5. Beyond that point, the test performances on smooth meshes degrades up to the point of random guessing at a training roughness of 2.5.

As a baseline, we build an *isotropic* graph CNN with the same network topology and number of parameters ($\approx 163k$). This model is insensitive to the mesh geometry and therefore performs exactly equal on all surfaces. While this enhances its robustness on very rough meshes, its test error of $19.80 \pm 3.43\%$ is an extremely bad result on MNIST. In contrast, the use of anisotropic filters of GEM-CNN allows it to reach a test error of only $0.60 \pm 0.05\%$ on the flat geometry. It is therefore competitive with conventional CNNs on pixel grids, which apply anisotropic kernels as well. More details on the datasets, models and further experimental setup are given in appendix E.1.

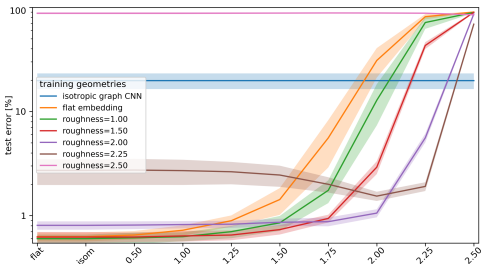


Figure 3: Test errors for MNIST digit classification on embedded meshes. Different lines denote train geometries, x-axis shows test geometries. Regions are standard errors of the means over 6 runs.

Model	Features	Accuracy (%)
ACNN (Boscaini et al., 2016)	SHOT	62.4
Geodesic CNN (Masci et al., 2015)	SHOT	65.4
MoNet (Monti et al., 2016)	SHOT	73.8
FeaStNet (Verma et al., 2018)	XYZ	98.7
ZerNet (Sun et al., 2018)	XYZ	96.9
SpiralNet++ (Gong et al., 2019)	XYZ	99.8
Graph CNN	XYZ	1.40±0.5
Graph CNN	SHOT	23.80±8
Non-equiv. CNN (SHOT frames)	XYZ	73.00±4.0
Non-equiv. CNN (SHOT frames)	SHOT	75.11±2.4
GEM-CNN	XYZ	99.73±0.04
GEM-CNN (broken symmetry)	XYZ	**99.89**±0.02

Table 2: Results of FAUST shape correspondence. Statistics are means and standard errors of the mean of over three runs. All cited results are from their respective papers.

6.2 Shape Correspondence

As a second experiment, we perform non-rigid shape correspondence on the FAUST dataset (Bogo et al., 2014), following Masci et al. (2015) [3] . The data consists of 100 meshes of human bodies in various positions, split into 80 train and 20 test meshes. The vertices are registered, such that vertices on the same position on the body, such as the tip of the left thumb, have the same identifier on all meshes. All meshes have 6890 vertices, making this a 6890-class segmentation problem.

The architecture transforms the vertices' XYZ coordinates (of type $3\rho_0$), via 6 convolutional layers to features $64\rho_0$, with intermediate features $16(\rho_0 \oplus \rho_1 \oplus \rho_2)$, with residual connections and the RegularNonlinearity with $N = 5$ samples. Afterwards, we use two 1×1 convolutions with ReLU to map first to 256 and then 6980 channels, after which a softmax predicts the registration probabilities. The 1×1 convolutions use a dropout of 50% and 1E-4 weight decay. The network is trained with a cross entropy loss with an initial learning rate of 0.01, which is halved when training loss reaches a plateau.

As all meshes in the FAUST dataset share the same topology, breaking the gauge equivariance in higher layers can actually be beneficial. As shown in (Weiler & Cesa, 2019), symmetry can be broken by treating non-invariant features as invariant features as input to the final 1×1 convolution.

As baselines, we compare to various models, some of which use more complicated pipelines, such as (1) the computation of geodesics over the mesh, which requires solving partial differential equations, (2) pooling, which requires finding a uniform sub-selection of vertices, (3) the pre-computation of SHOT features which locally describe the geometry (Tombari et al., 2010), or (4) post-processing refinement of the predictions. The GEM-CNN requires none of these additional steps. In addition, we compare to SpiralNet++ (Gong et al., 2019), which requires all inputs to be similarly meshed. Finally, we compare to an isotropic version of the GEM-CNN, which reduces to a conventional graph CNN, as well as a non-gauge-equivariant CNN based on SHOT frames. The results in table 2 show that the GEM-CNN outperforms prior works and a non-gauge-equivariant CNN, that isotropic graph CNNs are unable to solve the task and that for this data set breaking gauge symmetry in the final layers of the network is beneficial. More experimental details are given in appendix E.2.

7 Conclusions

Convolutions on meshes are commonly performed as a convolution on their underlying graph, by forgetting geometry, such as orientation of neighbouring vertices. In this paper we propose Gauge Equivariant Mesh CNNs, which endow Graph Convolutional Networks on meshes with anisotropic kernels and parallel transport. Hence, they are sensitive to the mesh geometry, and result in equivalent outputs regardless of the arbitrary choice of kernel orientation.

We demonstrate that the inference of GEM-CNNs is invariant under isometric deformations of meshes and generalizes well over a range of non-isometric deformations. On the FAUST shape correspondence task, we show that Gauge equivariance, combined with symmetry breaking in the final layer, leads to state of the art performance.

[3]These experiments were executed on QUVA machines.

REFERENCES

Bekkers, E. J., Lafarge, M. W., Veta, M., Eppenhof, K. A., Pluim, J. P., and Duits, R. Roto-translation covariant convolutional networks for medical image analysis. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2018.

Bogo, F., Romero, J., Loper, M., and Black, M. J. Faust: Dataset and evaluation for 3d mesh registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3794–3801, 2014.

Boscaini, D., Masci, J., Rodolà, E., and Bronstein, M. M. Learning shape correspondence with anisotropic convolutional neural networks. In *NIPS*, 2016.

Bouritsas, G., Bokhnyak, S., Ploumpis, S., Bronstein, M., and Zafeiriou, S. Neural 3d morphable models: Spiral convolutional networks for 3d shape representation learning and generation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 7213–7222, 2019.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, 2017.

Cohen, T. and Welling, M. Group equivariant convolutional networks. In *ICML*, 2016.

Cohen, T. S. and Welling, M. Steerable CNNs. In *ICLR*, 2017.

Cohen, T. S., Geiger, M., and Weiler, M. A general theory of equivariant CNNs on homogeneous spaces. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019a.

Cohen, T. S., Weiler, M., Kicanaoglu, B., and Welling, M. Gauge equivariant convolutional networks and the Icosahedral CNN. 2019b.

Crane, K., Desbrun, M., and Schröder, P. Trivial connections on discrete surfaces. *Computer Graphics Forum (SGP)*, 29(5):1525–1533, 2010.

Crane, K., de Goes, F., Desbrun, M., and Schröder, P. Digital geometry processing with discrete exterior calculus. In *ACM SIGGRAPH 2013 courses*, SIGGRAPH '13, New York, NY, USA, 2013. ACM.

Cucurull, G., Wagstyl, K., Casanova, A., Veličković, P., Jakobsen, E., Drozdzal, M., Romero, A., Evans, A., and Bengio, Y. Convolutional neural networks for mesh-based parcellation of the cerebral cortex. 2018.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pp. 3844–3852, 2016.

Gallier, J. and Quaintance, J. *Differential Geometry and Lie Groups: A Computational Perspective*, volume 12. Springer Nature, 2020.

Gong, S., Chen, L., Bronstein, M., and Zafeiriou, S. Spiralnet++: A fast and highly efficient mesh convolution operator. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 0–0, 2019.

Hanocka, R., Fish, N., Wang, Z., Giryes, R., Fleishman, S., and Cohen-Or, D. Alignet: Partial-shape agnostic alignment via unsupervised learning. *ACM Transactions on Graphics (TOG)*, 38(1):1–14, 2018.

Hoogeboom, E., Peters, J. W. T., Cohen, T. S., and Welling, M. HexaConv. In *International Conference on Learning Representations (ICLR)*, 2018.

Kipf, T. N. and Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*, 2017.

Kondor, R., Lin, Z., and Trivedi, S. Clebsch-gordan nets: a fully fourier space spherical convolutional neural network. In *NIPS*, 2018.

Kostrikov, I., Jiang, Z., Panozzo, D., Zorin, D., and Bruna, J. Surface networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2540–2548, 2018.

Lai, Y.-K., Jin, M., Xie, X., He, Y., Palacios, J., Zhang, E., Hu, S.-M., and Gu, X. Metric-driven rosy field design and remeshing. *IEEE Transactions on Visualization and Computer Graphics*, 16(1): 95–108, 2009.

Lang, L. and Weiler, M. A Wigner-Eckart Theorem for Group Equivariant Convolution Kernels. *arXiv preprint arXiv:2010.10952*, 2020.

Maron, H., Galun, M., Aigerman, N., Trope, M., Dym, N., Yumer, E., Kim, V. G., and Lipman, Y. Convolutional neural networks on surfaces via seamless toric covers. *ACM Trans. Graph.*, 36(4): 71–1, 2017.

Masci, J., Boscaini, D., Bronstein, M. M., and Vandergheynst, P. Geodesic convolutional neural networks on riemannian manifolds. *ICCVW*, 2015.

Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., and Bronstein, M. M. Geometric deep learning on graphs and manifolds using mixture model cnns. *CoRR*, abs/1611.08402, 2016. URL `http://arxiv.org/abs/1611.08402`.

Perraudin, N., Defferrard, M., Kacprzak, T., and Sgier, R. Deepsphere: Efficient spherical convolutional neural network with healpix sampling for cosmological applications. *Astronomy and Computing*, 27:130–146, 2019.

Poulenard, A. and Ovsjanikov, M. Multi-directional geodesic neural networks via equivariant convolution. *ACM Transactions on Graphics*, 2018.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017a.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pp. 5099–5108, 2017b.

Riegler, G., Osman Ulusoy, A., and Geiger, A. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3577–3586, 2017.

Schonsheck, S. C., Dong, B., and Lai, R. Parallel Transport Convolution: A New Tool for Convolutional Neural Networks on Manifolds. *arXiv:1805.07857 [cs, math, stat]*, May 2018.

Serre, J.-P. Linear representations of finite groups. 1977.

Sosnovik, I., Szmaja, M., and Smeulders, A. Scale-equivariant steerable networks. In *International Conference on Learning Representations (ICLR)*, 2020.

Sun, Z., Rooke, E., Charton, J., He, Y., Lu, J., and Baek, S. Zernet: Convolutional neural networks on arbitrary surfaces via zernike local tangent space estimation. *arXiv preprint arXiv:1812.01082*, 2018.

Tchapmi, L., Choy, C., Armeni, I., Gwak, J., and Savarese, S. Segcloud: Semantic segmentation of 3d point clouds. In *2017 international conference on 3D vision (3DV)*, pp. 537–547. IEEE, 2017.

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.

Tombari, F., Salti, S., and Di Stefano, L. Unique signatures of histograms for local surface description. In *European conference on computer vision*, pp. 356–369. Springer, 2010.

Tu, L. W. *Differential geometry: connections, curvature, and characteristic classes*, volume 275. Springer, 2017.

Verma, N., Boyer, E., and Verbeek, J. Feastnet: Feature-steered graph convolutions for 3d shape analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2598–2606, 2018.

Weiler, M. and Cesa, G. General E(2)-equivariant steerable CNNs. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019. URL https://arxiv.org/abs/1911.08251.

Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. 3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data. In *NeurIPS*, 2018a.

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

Wiersma, R., Eisemann, E., and Hildebrandt, K. CNNs on Surfaces using Rotation-Equivariant Features. *Transactions on Graphics*, 39(4), July 2020. doi: 10.1145/3386569.3392437.

Winkels, M. and Cohen, T. S. 3D G-CNNs for pulmonary nodule detection. In *Conference on Medical Imaging with Deep Learning (MIDL)*, 2018.

Worrall, D. and Welling, M. Deep scale-spaces: Equivariance over scale. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

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

Worrall, D. E., Garbin, S. J., Turmukhambetov, D., and Brostow, G. J. Harmonic Networks: Deep Translation and Rotation Equivariance. In *CVPR*, 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.

Zhao, F., Xia, S., Wu, Z., Duan, D., Wang, L., Lin, W., Gilmore, J. H., Shen, D., and Li, G. Spherical u-net on cortical surfaces: Methods and applications. *CoRR*, abs/1904.00906, 2019a. URL http://arxiv.org/abs/1904.00906.

Zhao, Y., Birdal, T., Lenssen, J. E., Menegatti, E., Guibas, L., and Tombari, F. Quaternion equivariant capsule networks for 3d point clouds. *arXiv preprint arXiv:1912.12098*, 2019b.

A GEOMETRY & PARALLEL TRANSPORT

A gauge, or choice of reference neighbor at each vertex, fully determines the neighbor orientations θ_{pq} and the parallel transporters $g_{q\to p}$ along edges. The following two subsections give details on how to compute these quantities.

A.1 LOCAL NEIGHBORHOOD GEOMETRY

Neighbours q of vertex p can be mapped uniquely to the tangent plane at p using a map called the Riemannnian logarithmic map, visualized in figure 1. A choice of reference neighbor then determines a reference frame in the tangent space which assigns polar coordinates to all other neighbors. The neighbour orientations θ_{pq} are the angular components of each neighbor in this polar coordinate system.

We define the tangent space T_pM at vertex p as that two dimensional subspace of $\mathbb{R}^3$, which is determined by a normal vector n given by the area weighted average of the normal vectors of the adjacent mesh faces. While the tangent spaces are two dimensional, we implement them as being embedded in the ambient space $\mathbb{R}^3$ and therefore represent their elements as three dimensional vectors. The reference frame corresponding to the chosen gauge, defined below, allows to identify these 3-vectors by their coefficient 2-vectors.

Each neighbor q is represented in the tangent space by the vector $\log_p(q) \in T_pM$ which is computed via the discrete analog of the Riemannian logarithm map. We define this map $\log_p : \mathcal{N}_p \to T_pM$ for neighbouring nodes as the projection of the edge vector $q - p$ on the tangent plane, followed by a rescaling such that the norm $|\log_p(q)| = |q - p|$ is preserved. Writing the projection operator on the tangent plane as $(\mathbb{1} - nn^\top)$, the logarithmic map is thus given by:

$$\log_p(q) \;:=\; |q-p| \frac{(\mathbb{1} - nn^\top)(q-p)}{|(\mathbb{1} - nn^\top)(q-p)|} \tag{6}$$

Geometrically, this map can be seen as "folding" each edge up to the tangent plane, and therefore encodes the orientation of edges and preserves their lengths.

The normalized reference edge vector $\log_p(q_0)$ uniquely determines a right handed, orthonormal reference frame $(e_{p,1},\, e_{p,2})$ of T_pM by setting $e_{p,1} := \log_p(q_0)/|\log_p(q_0)|$ and $e_{p,2} := n \times e_{p,1}$. The angle θ_{pq} is then defined as the angle of $\log_p(q)$ in polar coordinates corresponding to this reference frame. Numerically, it can be computed by

$$\theta_{pq} \;:=\; \operatorname{atan2}\left(e_{p,2}^\top \log_p(q),\; e_{p,1}^\top \log_p(q)\right).$$

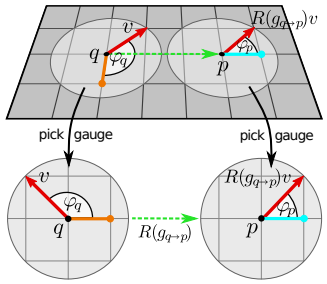
Given the reference frame $(e_{p,1}, e_{p,2})$, a 2-tuple of coefficients $(v_1, v_2) \in \mathbb{R}^2$ specifies an (embedded) tangent vector $v_1 e_{p,1} + v_2 e_{p,2} \in T_pM \subset \mathbb{R}^3$. This assignment is formally given by the *gauge map* $E_p : \mathbb{R}^2 \to T_pM \subset \mathbb{R}^3$ which is a vector space isomorphism. In our case, it can be identified with the matrix

$$E_p = \begin{bmatrix} | & | \\ e_{p,1} & e_{p,2} \\ | & | \end{bmatrix} \in \mathbb{R}^{3\times 2}. \tag{7}$$

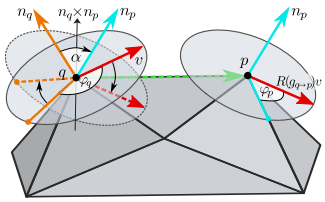
A.2 PARALLEL EDGE TRANSPORTERS

On curved meshes, feature vectors f_q and f_p at different locations q and p are expressed in different gauges, which makes it geometrically invalid to accumulate their information directly. Instead, when computing a new feature at p, the neighboring feature vectors at $q \in \mathcal{N}_p$ first have to be parallel transported into the feature space at p before they can be processed. The parallel transport along the edges of a mesh is determined by the (discrete) Levi-Civita connection corresponding to the metric induced by the ambient space $\mathbb{R}^3$. This connection is given by parallel transporters $g_{q\to p} \in [0, 2\pi)$ on the mesh edges which map tangent vectors $v_q \in T_qM$ at q to tangent vectors $R(g_{q\to p})v_q \in T_pM$ at p. Feature vectors f_q of type ρ are similarly transported to $\rho(g_{q\to p})f_q$ by applying the corresponding feature vector transporter $\rho(g_{q\to p})$.

In order to build some intuition, it is illustrative to first consider transporters on a planar mesh. In this case the parallel transport can be thought of as moving a vector along an edge without rotating it. The



(a) Parallel transport on a flat mesh.



(b) Parallel transport along an edge of a general mesh.

Figure 4: Parallel transport of tangent vectors $v \in T_qM$ at q to $R(g_{q\to p})v \in T_pM$ at p on meshes. On a flat mesh, visualized in figure 4(a), parallel transport moves a vector such that it stays parallel in the usual sense on flat spaces. The parallel transporter $g_{q\to p} = \varphi_p - \varphi_q$ corrects the transported vector *coefficients* for differing gauges at q and p. When transporting along the edge of a general mesh, the tangent spaces at q and p might not be aligned, see figure 4(b). Before correcting for the relative frame orientation via $g_{q\to p}$, the tangent space T_qM, and thus $v \in T_qM$, is rotated by an angle α around $n_q \times n_p$ such that its normal n_q coincides with that of n_p.

resulting abstract vector is then parallel to the original vector in the usual sense on flat spaces, see figure 4(a). However, if the (transported) source frame at q disagrees with the target frame at p, the *coefficients* of the transported vector have to be transformed to the target coordinates. This coordinate transformation from polar angles φ_q of v to φ_p of $R(g_{q\to p})v$ defines the transporter $g_{q\to p} = \varphi_p - \varphi_q$.

On general meshes one additionally has to account for the fact that the tangent spaces $T_qM \subset \mathbb{R}^3$ and $T_pM \subset \mathbb{R}^3$ are usually not parallel in the ambient space $\mathbb{R}^3$. The parallel transport therefore includes the additional step of first aligning the tangent space at q to be parallel to that at p, before translating a vector between them, see figure 4(b). In particular, given the normals n_q and n_p of the source and target tangent spaces T_qM and T_pM, the source space is being aligned by rotating it via $R_\alpha \in \mathrm{SO}(3)$ by an angle $\alpha = \arccos(n_q^\top n_p)$ around the axis $n_q \times n_p$ in the ambient space. Denote the rotated source frame by $(R_\alpha e_{q,1}, R_\alpha e_{q,2})$ and the target frame by $(e_{p,1}, e_{p,2})$. The angle to account for the parallel transport between the two frames, defining the discrete Levi-Civita connection on mesh edges, is then found by computing

$$g_{q\to p} \;=\; \mathrm{atan2}\left((R_\alpha e_{q,2})^\top e_{p,1},\; (R_\alpha e_{q,1})^\top e_{p,1}\right). \tag{8}$$

In practice we precompute these connections before training a model.

Under gauge transformations by angles g_p at p and g_q at q the parallel transporters transform according to

$$g_{q\to p} \;\mapsto\; g_p + g_{q\to p} - g_q\,. \tag{9}$$

Intuitively, this transformation states that a transporter in a transformed gauge is given by a gauge transformation back to the original gauge via $-g_q$ followed by the original transport by $g_{q\to p}$ and a transformation back to the new gauge via g_p.

For more details on discrete connections and transporters, extending to arbitrary paths e.g. over faces, we refer to (Lai et al., 2009; Crane et al., 2010; 2013).

B Deriving the Kernel Constraint

Given an input type ρ_{in}, corresponding to vector space V_{in} of dimension C_{in} and output type ρ_{out}, corresponding to vector space V_{out} of dimension C_{out}, we have kernels $K_{\text{self}} \in \mathbb{R}^{C_{\text{out}}\times C_{\text{in}}}$ and $K_{\text{neigh}} : [0, 2\pi) \to \mathbb{R}^{C_{\text{out}}\times C_{\text{in}}}$. Following Cohen et al. (2019b), we can derive a constraint on these kernels such that the convolution is invariant.

First, note that for vertex $p \in M$ and neighbour $q \in \mathcal{N}_p$, the coefficients of a feature vector f_p at p of type ρ transforms under gauge transformation $f_p \mapsto \rho(-g)f_p$. The angle θ_{pq} gauge transforms to $\theta_{pq} - g$.

Next, note that $\hat{f}_q := \rho_{\text{in}}(g_{q\to p})f_q$ is the input feature at q parallel transported to p. Hence, it transforms as a vector at p. The output of the convolution f'_p is also a feature at p, transforming as $\rho_{\text{out}}(-g)f'_p$.

The convolution then simply becomes:

$$f'_p = K_{\text{self}} f_p + \sum_q K_{\text{neigh}}(\theta_{pq})\hat{f}_q$$

Gauge transforming the left and right hand side, and substituting the equation in the left hand side, we obtain:

$$\rho_{\text{out}}(-g)f'_p =$$
$$\rho_{\text{out}}(-g)\left(K_{\text{self}} f_p + \sum_q K_{\text{neigh}}(\theta_{pq})\hat{f}_q\right) =$$
$$K_{\text{self}}\rho_{\text{in}}(-g)f_p + \sum_q K_{\text{neigh}}(\theta_{pq} - g)\rho_{\text{in}}(-g)\hat{f}_q$$

Which is true for any features, if $\forall g \in [0, 2\pi), \theta \in [0, 2\pi)$:

$$K_{\text{neigh}}(\theta - g) = \rho_{\text{out}}(-g)\ K_{\text{neigh}}(\theta)\ \rho_{\text{in}}(g), \tag{10}$$
$$K_{\text{self}} = \rho_{\text{out}}(-g)\ K_{\text{self}}\ \rho_{\text{in}}(g). \tag{11}$$

where we used the orthogonality of the representations $\rho(-g) = \rho(g)^{-1}$.

C Solving the Kernel Constraint

As also derived in (Weiler & Cesa, 2019; Lang & Weiler, 2020), we find all angle-parametrized linear maps between C_{in} dimensional feature vector of type ρ_{in} to a C_{out} dimensional feature vector of type ρ_{out}, that is, $K : S^1 \to \mathbb{R}^{C_{\text{out}}\times C_{\text{in}}}$, such that the above equivariance constraint holds. We will solve for $K_{\text{neigh}}(\theta)$ and discuss K_{self} afterwards.

The irreducible real representations (irreps) of $\text{SO}(2)$ are the one dimensional trivial representation $\rho_0(g) = 1$ of order zero and $\forall n \in \mathbb{N}$ the two dimensional representations of order n:

$$\rho_n : \text{SO}(2) \to \text{GL}(2, \mathbb{R}) : g \mapsto \begin{pmatrix} \cos ng & -\sin ng \\ \sin ng & \cos ng \end{pmatrix}.$$

Any representation ρ of $\text{SO}(2)$ of D dimensions can be written as a direct sum of irreducible representations

$$\rho \cong \rho_{l_1} \oplus \rho_{l_2} \oplus ...$$
$$\rho(g) = A(\rho_{l_1} \oplus \rho_{l_2} \oplus ...)(g)A^{-1}.$$

where l_i denotes the order of the irrep, $A \in \mathbb{R}^{D\times D}$ is some invertible matrix and the direct sum $\oplus$ is the block diagonal concatenations of the one or two dimensional irreps. Hence, if we solve the kernel constraint for all irrep pairs for the in and out representations, the solution for arbitrary representations, can be constructed. We let the input representation be irrep ρ_n and the output representation be irrep ρ_m. Note that $K(g^{-1}\theta) = (\rho_{\text{reg}}(g)[K])(\theta)$ for the infinite dimensional regular representation of $\text{SO}(2)$, which by the Peter-Weyl theorem is equal to the infinite direct sum $\rho_0 \oplus \rho_1 \oplus ....$

Using the fact that all $\text{SO}(2)$ irreps are orthogonal, and using that we can solve for $\theta = 0$ and from the kernel constraints we can obtain $K(\theta)$, we see that Eq. 10 is equivalent to

$$\hat{\rho}(g)K := (\rho_{\text{reg}} \otimes \rho_n \otimes \rho_m)(g)K = K$$

where $\otimes$ denotes the tensor product, we write $K := K(\theta)$ and filled in $\rho_{\text{out}} = \rho_m,\ \rho_{\text{in}} = \rho_n$. This constraint implies that the space of equivariant kernels is exactly the trivial subrepresentation of

$\hat{\rho}$. The representation $\hat{\rho}$ is infinite dimensional, though, and the subspace can not be immediately computed.

For $\mathrm{SO}(2)$, we have that for $n \geq 0$, $\rho_n \otimes \rho_0 = \rho_n$, and for $n, m > 0$, $\rho_n \otimes \rho_m \cong \rho_{n+m} \oplus \rho_{|n-m|}$. Hence, the trivial subrepresentation of $\hat{\rho}$ is a subrepresentation of the finite representation $\tilde{\rho} := (\rho_{n+m} \oplus \rho_{|n-m|}) \otimes \rho_n \otimes \rho_m$, itself a subrepresentation of $\hat{\rho}$.

As $\mathrm{SO}(2)$ is a connected Lie group, any $g \in \mathrm{SO}(2)$ can be written as $g = \exp tX$ for $t \in \mathbb{R}$, $X \in \mathfrak{so}(2)$, the Lie algebra of $\mathrm{SO}(2)$, and $\exp : \mathfrak{so}(2) \to \mathrm{SO}(2)$ the Lie exponential map. We can now find the trivial subrepresentation of $\tilde{\rho}$ looking infinitesimally, finding

$$\tilde{\rho}(\exp tX)K = K$$
$$\iff d\tilde{\rho}(X)K := \frac{\partial}{\partial t}\tilde{\rho}(\exp tX)|_{t=0}K = 0$$

where we denote $d\tilde{\rho}$ the Lie algebra representation corresponding to Lie group representation $\tilde{\rho}$. $\mathrm{SO}(2)$ is one dimensional, so for any single $X \in \mathfrak{so}(2)$, K is an equivariant map from ρ_m to ρ_n, if it is in the null space of matrix $d\tilde{\rho}(X)$. The null space can be easily found using a computer algebra system or numerically, leading to the results in table 1.

D EQUIVARIANCE

The GEM-CNN is by construction equivariant to gauge transformations, but additionally satisfies two important properties. Firstly, it only depends on the intrinsic shape of the 2D mesh, not how the mesh vertices are embedded in $\mathbb{R}^3$, since the geometric quantities like angles θ_{pq} and parallel transporters depend solely on the intrinsic properties of the mesh. This means that a simultaneous rotation or translation of all vertex coordinates, with the input signal *moving along* with the vertices, will leave the convolution output at the vertices unaffected.

The second property is that if a mesh has an orientation-preserving mesh isometry, meaning that we can map between the vertices preserving the mesh structure, orientations and all distances between vertices, the GEM-CNN is equivariant with respect to moving the signal along such a transformation. An (infinite) 2D grid graph is an example of a mesh with orientation-preserving isometries, which are the translations and rotations by 90 degrees. Thus a GEM-CNN applied to such a grid has the same equivariance properties a G-CNN (Cohen & Welling, 2016) applied to the grid.

D.1 PROOF OF MESH ISOMETRY EQUIVARIANCE

Throughout this section, we denote $p' = \phi(p), q' = \phi(q)$. An orientation-preserving mesh isometry is a bijection of mesh vertices $\phi : \mathcal{V} \to \mathcal{V}$, such that:

- Mesh faces are one-to-one mapped to mesh faces. As an implication, edges are one-to-one mapped to edges and neighbourhoods to neighbourhoods.
- For each point p, the differential $d\phi_p : T_pM \to T_{p'}M$ is orthogonal and orientation preserving, meaning that for two vectors $v_1, v_2 \in T_pM$, the tuple (v_1, v_2) forms a right-handed basis of T_pM, then $(d\phi_p(v_1), d\phi_p(v_2))$ forms a right-handed basis of $T_{p'}M$.

Lemma D.1. *Given a orientation-preserving isometry ϕ on mesh M, with on each vertex a chosen reference neighbour q_0^p, defining a frame on the tangent plane, so that the log-map $\log_p q$ has polar angle θ_q^p in that frame. For each vertex p, let $g_p = \theta^{p'}_{\phi(q_0^p)}$. Then for each neighbour $q \in \mathcal{N}_p$, we have $\theta^{p'}_{q'} = \theta^p_q + g_p$. Furthermore, we have for parallel transporters that $g_{q' \to p'} = g_{q \to p} - g_p + g_q$.*

Proof. For any $v \in T_pM$, we have that $\phi(\exp_p(v)) = \exp_{p'}(d\phi_p(v))$ (Tu, 2017, Theorem 15.2). Thus $\phi(\exp_p(\log_p q)) = q' = \exp_{p'}(d\phi_p(\log_p q))$. Taking the log-map at p' on the second and third expression and expressing in polar coordinates in the gauges, we get $(r^{p'}_{q'}, \theta^{p'}_{q'}) = d\phi_p(r^p_q, \theta^p_q)$. As ϕ is an orientation-preserving isometry, $d\phi_p$ is a special orthogonal linear map $\mathbb{R}^2 \to \mathbb{R}^2$ when expressed in the gauges. Hence $d\phi_p(r, \theta) = (r, \theta + z_p)$ for some angle z_p. Filling in $\theta^p_{q_0^p} = 0$, we find $z_p = g_p$,

proving the first statement. The second statement follows directly from the fact that parallel transport $q \to p$, then push-forward along ϕ to p' yields the same first pushing forward from q to q' along ϕ, then parallel transporting $q' \to p'$ (Gallier & Quaintance, 2020, Theorem 18.3 (2)). □

For any feature f of type ρ, we can define a push-forward along ϕ as $\phi_*(f)_{p'} = \rho(-g_p)f_p$.

Theorem D.1. *Given GEM-CNN convolution $K \star \cdot$ from a feature of type ρ_{in} to a feature of type ρ_{out}, we have that $K \star \phi_*(f) = \phi_*(K \star f)$.*

Proof.

$$\begin{aligned}
\phi_*(K \star f)_{p'} &= \rho_{\text{out}}(-g_p)\left(K_{\text{self}} f_p + \sum\nolimits_{q \in \mathcal{N}_p} K_{\text{neigh}}(\theta_{pq})\rho_{\text{in}}(g_{q\to p}) f_q\right) \\
&= \rho_{\text{out}}(-g_p)\left(K_{\text{self}} f_p + \sum\nolimits_{q' \in \mathcal{N}_{p'}} K_{\text{neigh}}(\theta_{p'q'} - g_p)\rho_{\text{in}}(g_{q'\to p'} + g_p - g_q) f_q\right) \\
&= \rho_{\text{out}}(-g_p)\left(K_{\text{self}} f_p + \sum\nolimits_{q' \in \mathcal{N}_{p'}} K_{\text{neigh}}(\theta_{p'q'} - g_p)\rho_{\text{in}}(g_p)\rho_{\text{in}}(g_{q'\to p'})\rho_{\text{in}}(-g_q) f_q\right) \\
&= K_{\text{self}}\rho_{\text{in}}(-g_p) f_p + \sum\nolimits_{q' \in \mathcal{N}_{p'}} K_{\text{neigh}}(\theta_{p'q'})\rho_{\text{in}}(g_{q'\to p'})\rho_{\text{in}}(-g_q) f_q \\
&= (K \star \phi_*(f))_{p'}
\end{aligned}$$

where in the second line we apply lemma D.1 and the fact that ϕ gives a bijection of neighbourhoods of p, in the third line we use the functoriality of ρ and in the fourth line we apply the kernel constraints on K_{self} and K_{neigh}. □

E Additional details on the experiments

E.1 Embedded MNIST

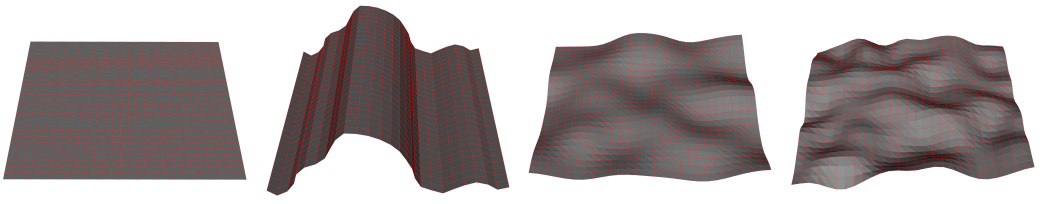
To create the intrinsically curved grids we start off with the flat, rectangular grid, shown in figure 5(a), which is embedded in the XY-plane. An independent displacement for each vertex in Z-direction is drawn from a uniform distribution. A subsequent smoothing step of the normal displacements with a Gaussian kernel of width σ yields geometries with different levels of curvature. Figures 5(c)-5(h) show the results for standard deviations of 2.5, 2, 1.5, 1, 0.75 and 0.5 pixels, which are denoted by their roughness $3 - \sigma$ as 0.5, 1, 1.5, 2, 2.25 and 2.5. In order to facilitate the generalization between different geometries we normalize the resulting average edge lengths.

The same GEM-CNN is used on all geometries. It consists of seven convolution blocks, each of which applies a convolution, followed by a RegularNonlinearity with $N = 7$ orientations, batch normalization and dropout of 0.1. This depth is chosen since GEM-CNNs propagate information only between direct neighbors in each layer, such that the field of view after 7 layers is $2 \times 7 + 1 = 15$ pixel. The input and output types of the network are scalar fields of multiplicity 1 and 64, respectively, which transform under the trivial representation and ensure a gauge invariant prediction. All intermediate layers use feature spaces of types $M\rho_0 \oplus M\rho_1 \oplus M\rho_2 \oplus M\rho_3$) with $M = 4,\ 8,\ 12,\ 16,\ 24,\ 32$. After a spatial max pooling, a final linear layer maps the 64 resulting features to 10 neurons, on which a softmax function is applied. The model has 163k parameters. A baseline GCN, applying by isotropic kernels, is defined by replacing the irreps ρ_i of orders $i \geq 1$ with trivial irreps ρ_0 and rescaling the width of the model such that the number of parameters is preserved. All models are trained for 20 epochs with a weight decay of 1E-5 and an initial learning rate of 1E-2. The learning rate is automatically decayed by a factor of 2 when the validation loss did not improve for 3 epochs.

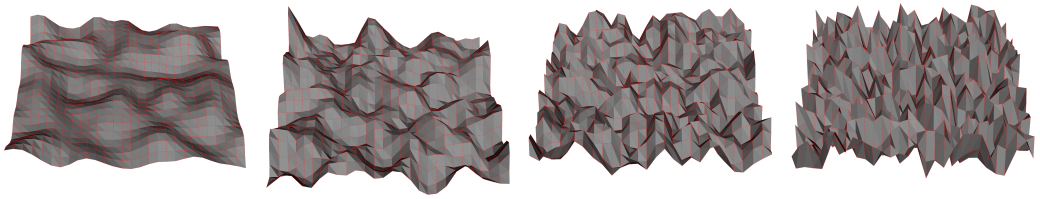
The experiments were run on a single TitanX GPU.

E.2 Shape Correspondence experiment

All experiments were ran on single RTX 2080TI GPUs, requiring 3 seconds / epoch.



(a) Flat embedding (b) Isometric embedding (c) Curved, roughness = 0.5 (d) Curved, roughness = 1



(e) Curved, roughness = 1.5 (f) Curved, roughness = 2 (g) Curved, roughness = 2.25 (h) Curved, roughness = 2.5

Figure 5: Examples of different grid geometries on which the MNIST dataset is evaluated. All grids have 28×28 vertices but are embedded differently in the ambient space. Figure 5(a) shows a flat embedding, corresponding to the usual pixel grid. The grid in Figure 5(b) is isometric to the flat embedding, its internal geometry is indistinguishable from that of the flat embedding. Figures 5(c)-5(h) show curved geometries which are not isometric to the flat grid. They are produced by a random displacement of each vertex in its normal direction, followed by a smoothing of displacements.

The non-gauge-equivariant CNN uses as gauges the SHOT local reference frames (Tombari et al., 2010). For one input and output channel, it has features $f_p \in \mathbb{R}$ convolution and weights $w \in \mathbb{R}^{2B+2}$, for $B \in \mathbb{N}$. The convolution is:

$$(K \star f)_p = w_0 f_p + \sum_{q \in \mathcal{N}_p} \left(w_1 + \sum_{n=1}^{B} (w_{2n} \cos(n\theta_{pq}) + w_{2n+1} \sin(n\theta_{pq}) \right) f_q. \tag{12}$$

This convolution kernel is an unconstranied band-limited spherical function. This is then done for C_{in} input channels and C_{out} output channels, giving $(2B+2)C_{\text{in}}C_{\text{out}}$ parameters per layer. In our experiments, we use $B = 2$ and 7 layers, with ReLU non-linearities and batch-norm, just as for the gauge equivariant convolution. After hyperparameter search in $\{16, 32, 64, 128, 256\}$, we found 128 channels to perform best.

F Additional experiments

F.1 RegularNonLinearity computational cost

Number of samples	Time / epoch (s)	Memory (GB)
none	21.2	1.22
1	21.9	1.22
5	21.6	1.23
10	21.5	1.24
20	22.0	1.27
50	21.7	1.35

Table 3: Run-time of one epoch training and validation and max memory usage of FAUST model without RegularNonLinearity of with varying number of samples used in the non-linearity. The hyperparameters are modified to have batch size 1.

In table 3, we show the computational cost of the RegularNonLinearity, computed by training and computing validation errors for 10 epochs. The run-time is not significantly affected, but memory usage is.

F.2 Equivariance Errors

In this experiment, we evaluate empirically equivariance to three kinds of transformations: gauge transformations, transformations of the vertex coordinates and transformations under isometries of the mesh, as introduced above in appenndix D. We do this on two data sets: the icosahedron, a platonic solid of 12 vertices, referred to in the plots as 'Ico'; and the deformed icosahedron, in which the vertices have been moved away from the origin by a factor of sampled from $\mathcal{N}(1, 0.01)$, referred to in the plots as 'Def. Ico'. We evaluate this on the GEM-CNN (7 layers, 101 regular samples, unless otherwise noted in the plots) and the Non-Equivariance CNN based on SHOT frames introduced above in Eq. 12 (7 layers unless otherwise noted in the plots). Both models have 16 channels input and 16 channels output. The equivariance model has scalar features as input and output and intermediate activations with band limit 2 with multiplicity 16. The non-equivariant model has hidden activations of 16 dimensions. If not for the finite samples of the RegularNonLinearity, the equivariant model should be exactly gauge invariant and invariant to isometries. Both models use batchnorm, in order to evaluate deeper models.

F.2.1 Gauge Equivariance

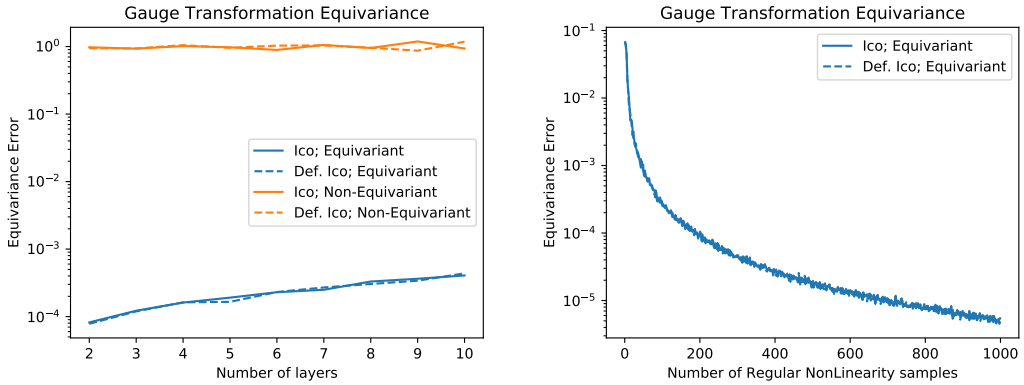


Figure 6: Equivariance error to gauge transformation.

We evaluate gauge equivariance by randomly initialising a model, randomly sampling input features. We also sample 16 random gauge transformations at each point. We compare the outputs of the model based on the different gauges. As the input and output features of the equivariant model are scalars, the outputs should coincide. This process is repeated 10 times. For the non-equivariant model, we compute frames based on SHOT and then randomly rotate these.

The equivariance error is quantified by as:

$$\sqrt{\frac{\mathbb{E}_{\Phi,f}\mathbb{E}_{p,c}\mathrm{Var}_g(\Phi_g(f)_{p,c})}{\mathrm{Var}_{\Phi,f,p,c}(\Phi_{g_0}(f)_{p,c})}} \tag{13}$$

where $\Phi_g(f)_c$ denotes the model Φ with gauge transformed by g applied to input f then taken the c-th channel, $\mathbb{E}_{\Phi,f}$ denotes the expectation over model initialisations and random inputs, for which we take 10 samples, $\mathbb{E}_{p,c}$ denotes averaging over the 12 vertices and 16 output channels, Var_g denotes the variance over the different gauge transformations, $\mathrm{Var}_{\Phi,f,c}$ takes the variance over the models, inputs and channels, and g_0 denotes one of the sampled gauge transformations. This quantity indicates how much the gauge transformation affects the output, normalized by how much the model initialisations and initial parameters affect the output.

Results are shown in Figure 6. As expected, the non-equivariant model is not equivariant to gauge transformations. The equivariant model approaches gauge equivariance as the number of samples of the Regular NonLinearity increases. As expected, the error to gauge equivariance accumulate as the number of layers increases. The icosahedron and deformed icosahedron behave the same.

F.2.2 AMBIENT EQUIVARIANCE

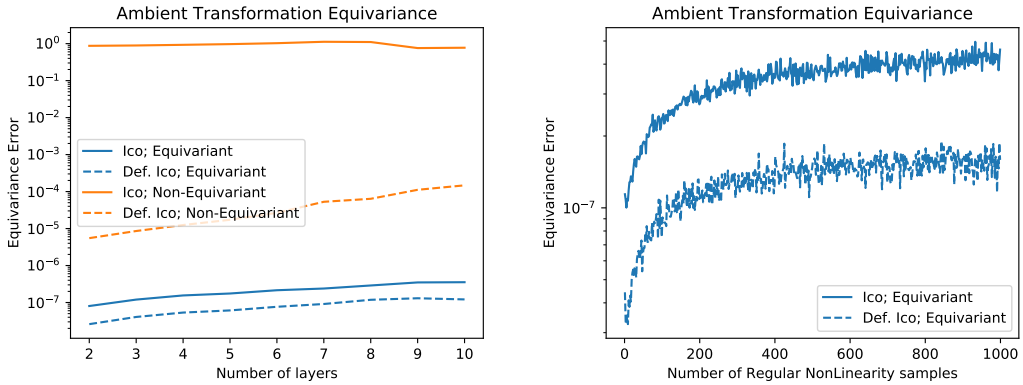


Figure 7: Equivariance error to ambient transformations of the vertex coordinates.

In this experiment, we measure whether the output is invariant to when all vertex coordinates are jointly transformed under rotations and translations. We perform the experiment as above, but sample as transformations g 300 translations and rotations of the ambient space $\mathbb{R}^3$. We evaluate again using Eq 13, where g now denotes a ambient transformation.

Results are shown in Figure 7. We see that the equivariant GEM-CNN is invariant to these ambient transformations. Somewhat unexpectedly, we see that the non-equivariant model based on SHOT frames is not invariant. This is because of an significant failure mode of SHOT frames in particular and heuristically chosen gauges with a non-gauge-equivariant methods in general. On some meshes, the heuristic is unable to select a canonical frame, because the mesh is locally symmetric under (discrete subgroups of) planar rotations. This is the case for the icosahedron. Hence, SHOT can not disambiguate the X from the Y axis. The reason this happens in the SHOT local reference frame selection (Tombari et al., 2010) is the first two singular values of the M matrix are equal, making a choice between the first and second singular vectors ambiguous. This ambiguity breaks ambient invariance. For the non-symmetric deformed icosahedron, this problem for the non-equivariant method disappears.

F.2.3 ISOMETRY EQUIVARIANCE

The icosahedron has 60 orientation-preserving isometries. We evaluate equivariance using:

$$\sqrt{\frac{\mathbb{E}_{\Phi,f}\mathbb{E}_{p,c}(\Phi(g(f)_{p,c} - \Phi(f)_{g(p),c})^2}{\mathrm{Var}_{\Phi,f,p,c}(\Phi(f)_{p,c})}}$$

where $g : M \to M$ is an orientation-preserving isometry, sampled uniformly from all 60 and $g(f)$ is the transformation of a scalar input feature $f : M \to \mathbb{R}^{C_{\text{in}}}$ by pre-composing with g^{-1}.

As expected, the non-equivariant model is not equivariant to isometries. The GEM-CNN is not equivariant to the icosahedral isometries on the deformed icosahedron, as the deformation removes the symmetry. As the number of Regular NonLinearity samples increases, the GEM-CNN becomes more equivariant. Interestingly, the GEM-CNN is equivariant whenever the number of samples is a multiple of 5. This is because the stabilizer subgroup of the icosahedron at the vertices is the cyclic group of order 5. Whenever the RegularNonLinearity has a multiple of 5 samples, it is exactly equivariant to these transformations.

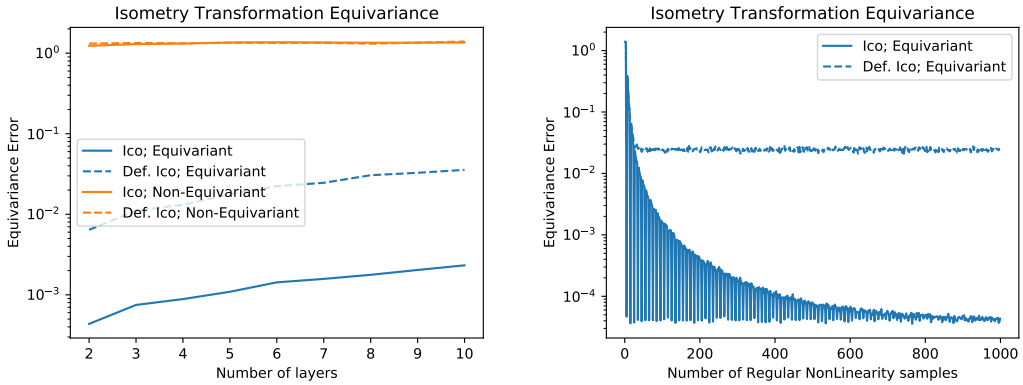


Figure 8: Equivariance error to isometry transformation.

G Equivariance Error Bounds on Regular Non-Linearity

The regular non-linearity acts on each point on the sphere in the following way. For simplicity, we assume that the representation is U copies of $\rho_0 \oplus \rho_1 \oplus ... \oplus \rho_M$. One such copy can be treated as the discrete Fourier modes of a circular signal with band limit M. We map these Fourier modes to N spatial samples with an inverse Discrete Fourier Transform (DFT) matrix. Then apply to those samples a point-wise non-linearity, like ReLU, and map back to the Fourier modes with a Discrete Fourier Transform Matrix.

This procedure is exactly equivariant for gauge transformation with angles multiple of $2\pi/N$, but approximately equivariant for small rotations in between.

In equations, we start with Fourier modes $x_0, (x_\alpha(m), x_\beta(m))_{m=1}^B$ at some point on the sphere and result in Fourier modes $z_0, (z_\alpha(m), z_\beta(m))_{m=1}^B$. We let $t = 0, ..., N-1$ index the spatial samples.

$$
\begin{aligned}
x(t) &= x_0 + \sum_m x_\alpha(m) \cos\left(\frac{2\pi}{N} mt\right) + \dots \\
&\qquad \sum_m x_\beta(m) \sin\left(\frac{2\pi}{N} mt\right) \\
y(t) &= f(x(t)) \\
z_0 &= \frac{1}{N} \sum_t y(t) \\
z_\alpha(m) &= \frac{2}{N} \sum_t \cos\left(\frac{2\pi}{N} mt\right) y(t) \\
z_\beta(m) &= \frac{2}{N} \sum_t \sin\left(\frac{2\pi}{N} mt\right) y(t)
\end{aligned}
\tag{14}
$$

Note that Nyquist's sampling theorem requires us to pick $N \geq 2B + 1$, as otherwise information is always lost. The normalization is chosen so that $z_\alpha(m) = x_\alpha(m)$ if f is the identity.

Now we are interested in the equivariance error between the following two terms, for small rotation $\delta \in [0, 1)$. Any larger rotation can be expressed in a rotation by a multiple of $2\pi/N$, which is exactly equivariant, followed by a smaller rotation. We let $z_\alpha^{FT}(m)$ be the resulting Fourier mode if first the input is gauge-transformed and then the regular non-linearity is applied, and let $z_\alpha^{TF}(m)$ be the result

of first applying the regular non-linearity, followed by the gauge transformation.

$$\begin{aligned} z_\alpha^{FT}(m) &= \frac{2}{N}\sum_t \cos\left(\frac{2\pi}{N}mt\right) y(t+\delta) \\ &= \frac{2}{N}\sum_t c_m(t) y(t+\delta) \\ z_\alpha^{TF}(m) &= \frac{2}{N}\sum_t \cos\left(\frac{2\pi}{N}m(t-\delta)\right) y(t) \\ &= \frac{2}{N}\sum_t c_m(t-\delta) y(t) \end{aligned}$$

where we defined for convenience $c_m(t) = \cos(2\pi mt/N)$. We define norms $||x||_1 = |x_0| + \sum_m (|x_\alpha(m)| + |x_\beta(m)|)$ and $||\partial x||_1 = \sum_m m(|x_\alpha(m)| + |x_\beta(m)|)$.

Theorem G.1. *If the input x is band limited by B, the output z is band limited by B', N samples are used and the non-linearity has Lipschitz constant L_f, then the error to the gauge equivariance of the regular non-linearity bounded by:*

$$||z^{FT} - z^{TF}||_1 \leq \frac{4\pi L_f}{N}\left((2B'+\frac{1}{2})||\partial x||_1 + B'(B'+1)||x||_1\right)$$

which goes to zero as $N \to \infty$.

Proof. First, we note, since the Lipschitz constant of the cosine and sine is 1:

$$\begin{aligned} |c_m(t-\delta) - c_m(t)| &\leq \frac{2\pi m\delta}{N} \leq \frac{2\pi m}{N} \\ |x(t+\delta) - x(t)| &\leq \frac{2\pi}{N}\sum_m m(|x_\alpha(m)| + |x_\beta(m)|) \\ &\leq \frac{2\pi}{N}||\partial x||_1 \\ |y(t+\delta) - y(t)| &\leq L_f\frac{2\pi}{N}||\partial x||_1 \\ |c_m(t)| &\leq 1 \\ |x(t)| &\leq |x_0| + \sum_m (|x_\alpha(m)| + |x_\beta(m)|) \\ &\leq ||x||_1 \\ |y(t)| &\leq L_f||x||_1 \end{aligned}$$

Then:

$$\begin{aligned} &|c_m(t)y(t+\delta) - c_m(t-\delta)y(t)| \\ =&|c_m(t)\left[y(t+\delta) - y(t)\right] - y(t)\left[c_m(t-\delta) - c_m(t)\right]| \\ \leq&|c_m(t)||y(t+\delta) - y(t)| + |y(t)||c_m(t-\delta) - c_m(t)| \\ \leq&L_f\frac{2\pi}{N}||\partial x||_1 + L_f||x||_1\frac{2\pi m}{N} \\ =&\frac{2\pi L_f}{N}\left(||\partial x||_1 + m||x||_1\right) \end{aligned}$$

So that finally:

$$\begin{aligned}
&|z_\alpha^{FT}(m) - z_\alpha^{TF}(m)| \\
\leq &\frac{2}{N}\sum_t |c_m(t)y(t+\delta) - c_m(t-\delta)y(t)| \\
\leq &\frac{4\pi L_f}{N}\left(||\partial x||_1 + m||x||_1\right)
\end{aligned}$$

The sinus component $|z_\beta^{FT}(m) - z_\beta^{TF}(m)|$ has the same bound, while $|z_0^{FT} - z_0^{TF}| = |y(t+\delta) - y(t)|$, which is derived above. So if z is band-limited by B':

$$\begin{aligned}
||z^{FT} - z^{TF}||_1 = &|z_0^{FT} - z_0^{TF}| + \\
&\sum_{m=1}^{B'} |z_\alpha^{FT}(m) - z_\alpha^{TF}(m)| + |z_\beta^{FT}(m) - z_\beta^{TF}(m)| \\
&\leq \frac{4\pi L_f}{N}\left((2B' + \frac{1}{2})||\partial x||_1 + \sum_{m=1}^{B'} 2m||x||_1\right) \\
&= \frac{4\pi L_f}{N}\left((2B' + \frac{1}{2})||\partial x||_1 + B'(B'+1)||x||_1\right)
\end{aligned}$$

Since $||\partial x||_1 = \mathcal{O}(B||x||_1)$, we get $||z^{FT} - z^{TF}||_1 = \mathcal{O}(\frac{BB' + B'^2}{N}||x||_1)$, which obviously vanishes as $N \to \infty$. □