Embed Me If You Can: A Geometric Perceptron

Pavlo Melnyk, Michael Felsberg, Mårten Wadenbäck
Computer Vision Laboratory, Department of Electrical Engineering, Linköping University
{pavlo.melnyk, michael.felsberg, marten.wadenback}@liu.se

Abstract

Solving geometric tasks involving point clouds by using machine learning is a challenging problem. Standard feed-forward neural networks combine linear or, if the bias parameter is included, affine layers and activation functions. Their geometric modeling is limited, which motivated the prior work introducing the multilayer hypersphere perceptron (MLHP). Its constituent part, i.e.*, the hypersphere neuron, is obtained by applying a conformal embedding of Euclidean space. By virtue of Clifford algebra, it can be implemented as the Cartesian dot product of inputs and weights. If the embedding is applied in a manner consistent with the dimensionality of the input space geometry, the decision surfaces of the model units become combinations of hyperspheres and make the decision-making process geometrically interpretable for humans. Our extension of the MLHP model, the multilayer geometric perceptron (MLGP), and its respective layer units,* i.e.*, geometric neurons, are consistent with the 3D geometry and provide a geometric handle of the learned coefficients. In particular, the geometric neuron activations are isometric in 3D, which is necessary for rotation and translation equivariance. When classifying the 3D Tetris shapes, we quantitatively show that our model requires no activation function in the hidden layers other than the embedding to outperform the vanilla multilayer perceptron. In the presence of noise in the data, our model is also superior to the MLHP.*

1. Introduction

Understanding the geometry of a neuron is a crucial prerequisite to successfully performing geometric deep learning [4]. Owing to their inherent connection to the notion of *distance*, circles (or spheres) are fundamental atomic structures for defining geometric constraints. Note, e.g., how the third corner of a triangle is constrained by specifying the radii of two circles centered in each of the other two corners.

This motivates us to consider geometries of decision surfaces beyond hyperplanes, such as hyperspheres, in order to properly represent isometries.

Going beyond planar decision surfaces to non-planar ones increases the complexity of the model, but this has proved to be beneficial performance-wise [5, 3, 14]. To construct and represent such surfaces, one needs to consider more general spaces, in which case, Klein geometries [20] provide a useful theoretical framework.

In this paper, we employ Clifford algebra as a tool to perform computations in conformal geometry. Building on top of the previous works on Clifford neurons [6] and spherical decision surfaces [18], we explore the multilayer hypersphere perceptron (MLHP) [2] model applied to the problem of classifying the 3D Tetris shapes, essentially a collection of point clouds, see Section 5.1 and Fig. 3. To the best of our knowledge, MLHP has not been previously investigated in this context. We focus specifically on 3D geometry, which is important for tasks such as pose estimation, which in turn is a prerequisite for grasping, 3D inpainting, and augmented reality.

Striving to make the decision-making process more intuitive and be consistent with the dimensionality of the input space geometry, we perform the conformal embedding in a Minkowski space. Consequently, we observe that the decision surfaces of the MLHP units become combinations of hyperspheres. We call such an extension of the MLHP model the *multilayer geometric perceptron* (MLGP) and refer to its respective layer units as *geometric neurons*. Owing to the homogeneous representation [8], we provide an interpretation of our model parameters *directly* in the Euclidean space, which is only intuitive if the embedding agrees with the input geometry.

We summarize our contributions as follows:

- (**a**) We demonstrate how spherical decision surfaces improve the understanding of the decision-making process of neural networks, provided that their construction is done adhering to the input space geometry.
- (**b**) We propose an extension to the MLHP, the MLGP model[1] with *geometric neurons*, and show that, apart from being more geometrically explainable, it achieves favorable quantitative results when classifying the 3D Tetris shapes, and is superior when they are perturbed.

[1] The code is available at github.com/pavlo-melnyk/mlgp-embedme.

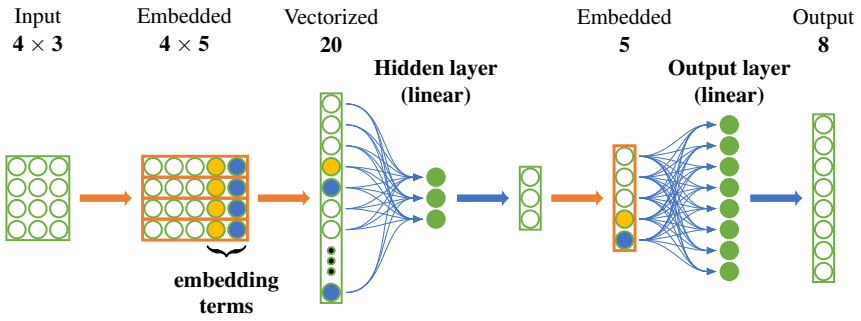

Figure 1. The proposed (feed-forward) MLGP model — our modification of the baseline MLHP. The embedding of the input array is performed *point-wise* and at each subsequent layer vector-wise: the first embedding term (yellow) is always set to -1, the second (blue) is the scaled magnitude of the vector being embedded, i.e., $-\frac{1}{2}||\mathbf{x}||^2$. Since the embedding of the model input is point-wise, the hidden layer consists of *geometric neurons* that represent combinations of hyperspheres. The output layer consists of hypersphere neurons [3].

(**c**) By introducing a matrix operator, we derive the isomorphism between the sandwich product of the motor (rotation followed by translation) with a general geometric object in the conformal $\mathbb{ME}^3 \equiv \mathbb{R}^{3+1,1}$ space (see Section 3.1 for notation) and the corresponding matrix–vector product in the Euclidean $\mathbb{R}^5$ space. Using this result, we prove that the geometric neuron activations are isometric in $\mathbb{R}^3$ by construction, which is necessary for rotation and translation equivariance. We further demonstrate it experimentally.

2. Related work

Research on neural networks equivariant to certain symmetry groups has been expanding over the past few years, e.g., SE(3)-equivariant models [23], [21], and the SO(3)-equivariant network [1]. To introduce to the reader a broader picture of concepts related to our work, we start by noting that a Klein geometry can be viewed as a homogeneous space together with a transformation (symmetry) group acting on that space. Conformal geometry on the sphere is modeled as a Klein geometry with the underlying space being the sphere $\mathcal{S}^n$ and the *Lorentz group* [12] of an $(n+2)$-dimensional space (e.g., $\mathbb{R}^{n+1,1}$) acting as the transformation group. The main computational mechanism in conformal geometry — Clifford (geometric) algebras — is utilized in all the following methods.

2.1. Hyperspherical decision surfaces

We draw the main inspiration for our work from the idea of modeling hyperspherical surfaces using a conformal space representation introduced in [13] and exploited in [18]. The hypersphere neuron with, as the name suggests, a hypersphere as decision surface is proposed as a variant of a Clifford neuron in [3]. Therein, a hyperspherical surface is shown as a generalization of a hyperplane. A multilayer feed-forward neural network based on hypersphere neurons is designed in [2] and is referred to as MLHP. The authors describe how a certain amount of reduction in computational complexity can be achieved when using the MLHP model for some types of learning tasks. However, the prior work did not consider the problem of classifying point clouds.

2.2. Clifford neural networks

A multilayer Clifford neural network, as well as the corresponding back-propagation derivation, is first proposed and discussed in [16, 17]. As per [6], work on another multilayer model, two key concepts for Clifford neural machinery originate and are easy to analyze at the neuron level: (*i*) the ability to process various geometric entities, and (*ii*) the concept of the geometric model. The latter acts as certain transformations on the processed data and becomes inherent by choosing a particular Clifford algebra. The paper [6] also introduces the spinor Clifford neurons (SCN) with weights acting like rotors from two sides. It is demonstrated how a single SCN can be used to compute Möbius transformations in a linear way: something unattainable by any real-valued network. Additionally, the paper describes Clifford-valued activation functions for all two-dimensional Clifford algebras.

The work [19] introduces the hyperconic multilayer perceptron with quadratic hypersurfaces spawned by the hyperconic neurons in the hidden layer. The output units in their model are hypersphere neurons. The model is trained using the particle swarm optimization (PSO) algorithm [7]. The authors of [22] use the generalization of the geometric algebra of quadratic surfaces $\mathcal{G}_{6,3}$ [24] and propose a new Clifford neuron — the hyperellipsoidal neuron. It is shown how decision surfaces of different geometric shapes, derived as special cases of an ellipsoid, can be obtained, depending on the input data: spherical decision surface, ellipsoidal, cylindrical, or a pair of planes. They use some hybrid training algorithm, in which the center of the hyperellipsoid is updated by unsupervised learning and the radii by a supervised learning method.

Less related to geometric problems are models of recurrent Clifford NNs originally discussed in [11], where their

dynamics are studied from the perspective of the existence of energy functions.

We focus on spherical decision surfaces and feed-forward-like models trainable with backpropagation, but in contrast to prior work, we exploit a different embedding scheme that is consistent with the 3D geometry of the input space. Consequently, the first hidden layer units in our model can be seen as combinations of hypersphere neurons and, along with the rest of the layers, do not necessarily require an activation function. Moreover, such units activations are isometric to ridig body transformations of the input. Utilizing the chosen embedding strategy results in a more explainable decision-making process and can even be superior performance-wise when dealing with noisy data. We explain the details and discuss the advantages of our method in Section 4.

3. Background

The proposed spherical model MLGP, which we will present in detail in Section 4, is based on a particular embedding of the Euclidean vector space $\mathbb{R}^3$ into a Minkowski space, which is isomorphic to the Euclidean space $\mathbb{R}^5$. The construction of this embedding is reminiscent of the way in which projective n-space is embedded in $\mathbb{R}^{n+1}$ by means of homogeneous coordinates. Many key aspects pertaining to the geometric interpretation will also be similar for these two embeddings, and we therefore encourage the reader to keep the more familiar projective case in mind when studying our discussion of the conformal embedding in Section 3.1.

In particular, we wish to emphasize the fact that successful interpretation of various geometric entities, given in their embedding representation, requires appropriate normalization to be applied. To make sense of the homogeneous coordinates of a (proper) point, for example, the appropriate *point normalization* is achieved by simply dividing the whole coordinate vector by its final coordinate. Correspondingly, recall that the dual projective entities (i.e., hyperplanes) require a fundamentally different type of normalization; here we require that the final coordinate is ≤ 0 and that the squares of the others sum to one. If $\mathbf{p} = (p_1, \ldots, p_n, -\Delta)$ is normalized in this way, it is possible to directly interpret $(p_1, \ldots, p_n)$ as the outward pointing unit normal of the hyperplane, and $\Delta \geq 0$ as its distance to the origin.

Slightly related, we also wish to highlight the role of scalar products between objects in the embedding space. In the projective embedding, we recall that incidence relations between points and hyperplanes are expressed as orthogonality of the embedding vectors, i.e., $\mathbf{p}^\top \mathbf{x} = 0$. If this scalar product is not zero, it does not have a direct geometric interpretation, *unless the representations have been properly normalized*. If proper normalization has been applied, then of course $\left|\mathbf{p}^\top \mathbf{x}\right|$ is simply the Euclidean distance between the point and the hyperplane.

3.1. Minkowski space and conformal embedding

Taking one step further from homogeneous coordinates opens up a whole new world in the form of *conformal geometry* [13], i.e., angle-preserving transformations on a space. *Minkowski space* [12], named after H. Minkowski who introduced $\mathbb{R}^{3,1}$ as a model of space-time, is the real vector space $\mathbb{ME}^n \equiv \mathbb{R}^{n+1,1}$ where the first $n+1$ basis vectors square to $+1$ and the last of them squares to -1.

Minkowski $\mathbb{R}^{1,1}$ space. The relevance of Minkowski spaces for Euclidean geometry is well-described in terms of the Minkowski $\mathbb{R}^{1,1}$ plane in [12]. Its *orthonormal basis* is defined as $\{e_+,\ e_-\}$, where $e_+^2 = 1,\ e_-^2 = -1$, and $e_+ \cdot e_- = 0$.

A *null basis* can then be constructed as the two vectors $\{e_0,\ e_\infty\}$, where $e_0 = \frac{1}{2}(e_- - e_+)$ is the origin and $e_\infty = e_- + e_+$ is the point at infinity. Note the properties $e_0^2 = e_\infty^2 = 0$ and $e_0 \cdot e_\infty = -1$, which follow from the signature properties of e_+ and e_- and the fact that they are orthogonal.

Conformal embedding. Given a vector in Euclidean space, $\mathbf{x} \in \mathbb{R}^n$, one can construct the conformal space as $\mathbb{ME}^n \equiv \mathbb{R}^{n+1,1} = \mathbb{R}^n \oplus \mathbb{R}^{1,1}$. The embedding of $\mathbf{x}$ in the conformal space $\mathbb{ME}^n$ represents the stereographic projection of $\mathbf{x}$ onto a projection sphere defined in $\mathbb{ME}^n$ as

$$X = \mathcal{C}(\mathbf{x}) = \mathbf{x} + \frac{1}{2}\mathbf{x}^2\, e_\infty + e_0\,, \tag{1}$$

where $X \in \mathbb{ME}^n$ is called *normalized* and $\mathbf{x}^2 = \mathbf{x}\cdot\mathbf{x} = ||\mathbf{x}||^2$. Observe that $X^2 = 0$. Note in this context that the *Clifford (geometric) product* of vectors (see equation (1) in [9]) is denoted as concatenation of literals. The standard inner product is related to the geometric product as

$$\mathbf{x} \cdot \mathbf{y} = \frac{\mathbf{xy} + \mathbf{yx}}{2} \quad \text{and} \quad X \cdot Y = \frac{XY + YX}{2}\,. \tag{2}$$

From (1), we obtain the naming of the two null vectors:

$$e_0 = \mathcal{C}(\mathbf{0}) \quad \text{and} \quad e_\infty = \lim_{|\mathbf{x}|\to\infty} \frac{2}{\mathbf{x}^2}\,\mathcal{C}(\mathbf{x}). \tag{3}$$

The embedding (1) is homogeneous, i.e., all embedding vectors in the equivalence class

$$[X] = \left\{Z \in \mathbb{R}^{n+1,1} : Z = \gamma X,\ \gamma \in \mathbb{R} \setminus \{0\}\right\} \tag{4}$$

are taken to represent the same vector $\mathbf{x}$. This property is fundamental for the remainder of the paper.

Scalar products. Given $Y = \mathbf{y} + \frac{1}{2}\mathbf{y}^2\, e_\infty + e_0$, the scalar product of two embeddings in conformal space turns out to be the Euclidean distance, which constitutes the basis for deriving the hypersphere neuron [3]:

$$X \cdot Y = -\frac{1}{2}(\mathbf{x} - \mathbf{y})^2\,. \tag{5}$$

3.2. Hypersphere as classifier

A *normalized hypersphere* in $\mathbb{ME}^n$ is a hypersphere $S \in \mathbb{ME}^n$ with center $\mathbf{c} = (c_1, \ldots, c_n) \in \mathbb{R}^n$ embedded as $C \in \mathbb{ME}^n$, radius $r \in \mathbb{R}$, and the coefficient for e_0 set to 1. It is defined in the conformal space as

$$S = \mathbf{c} + \frac{1}{2}(\mathbf{c}^2 - r^2)\, e_\infty + e_0 = C - \frac{1}{2} r^2 e_\infty \,. \quad (6)$$

Keeping in mind that $\mathbb{R}^n$ has a basis $(e_1, \ldots, e_n)$, we obtain the scalar product of an embedded data vector X and a hypersphere S in $\mathbb{ME}^n$:

$$X \cdot S = X \cdot C - \frac{1}{2} r^2 X \cdot e_\infty = -\frac{1}{2}(\mathbf{x} - \mathbf{c})^2 + \frac{1}{2} r^2 \,. \quad (7)$$

Thus, it follows that $X \cdot S = 0 \iff |\mathbf{x} - \mathbf{c}| = |r|$. Specifically, the scalar product shows where the input vector is relative to the hypersphere: inside (positive product), on (zero), or outside (negative product) of the hypersphere.

It has been shown in [3] that by embedding a data vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ and the hypersphere $S \in \mathbb{ME}^n$ in $\mathbb{R}^{n+2}$ as

$$\begin{aligned} \boldsymbol{X} &= (x_1, \ldots, x_n, -1, -\frac{1}{2}\mathbf{x}^2) \in \mathbb{R}^{n+2}, \\ \boldsymbol{S} &= (c_1, \ldots, c_n, \frac{1}{2}(\mathbf{c}^2 - r^2), 1) \in \mathbb{R}^{n+2}, \end{aligned} \quad (8)$$

with $\boldsymbol{S}$ further referred to as a *normalized hypersphere* (in $\mathbb{R}^{n+2}$), one can implement a hypersphere neuron in $\mathbb{ME}^n$ as a standard dot product in $\mathbb{R}^{n+2}$ since

$$\begin{aligned} \boldsymbol{X} \cdot \boldsymbol{S} &= \mathbf{x} \cdot \mathbf{c} \; - \frac{1}{2}(\mathbf{c}^2 - r^2) - \frac{1}{2}\mathbf{x}^2 = \\ &= -\frac{1}{2}(\mathbf{x} - \mathbf{c})^2 + \frac{1}{2} r^2 = X \cdot S \,. \end{aligned} \quad (9)$$

4. The proposed MLGP model

From now on and depending on the context, *hypersphere* refers to either a decision surface (geometric entity), or the (scaled) embedded vector $\boldsymbol{S} \in \mathbb{R}^{n+2}$ (8), or a classifier (the hypersphere neuron).

4.1. Geometric neuron: point-wise embedding

In the MLHP [2], the model input is treated as a single real n-vector that is subsequently embedded in $\mathbb{R}^{n+2}$, as discussed in Section 3.2. However, such an embedding scheme is not the most intuitive choice for all types of learning problems.

We propose to apply the conformal embedding to the input *point-wise*, in contrast to performing the embedding on a vectorized input as in MLHP [2]. We motivate this choice by reasoning from a geometric perspective.

For the simplicity of argument, consider 3D geometry. Take, e.g., the geometry problem of classifying point clouds, each consisting of k points, to which random 3D rigid transformations are applied. Suppose one applies the transformation to the points $\mathbb{R}^{k\times 3}$ and then either **(a)** stacks these transformed points in a single $\mathbb{R}^{3k}$ vector or **(b)** keeps the transformed points as $\mathbb{R}^{k\times 3}$. In **(a)**, the original rigid transformation in $\mathbb{R}^3$ is not equivalent to a rigid body transformation in this $\mathbb{R}^{3k}$ space. Therefore, the embedding in the corresponding $\mathbb{ME}^{3k} \cong \mathbb{R}^{3k+2}$ conformal space will not be injective under 3D rigid body transformations — the embedding will be *invariant* to a *too large* class of transformations. Whereas in **(b)**, one embeds these transformed $\mathbb{R}^{k\times 3}$ points point-wise in $\mathbb{ME}^3 \cong \mathbb{R}^5$, and the transformation maps one-to-one onto the conformal space (resulting in $\mathbb{R}^{k\times 5}$).

We perform this point-wise embedding only in the first layer, but it can be done in the remaining layers as well. In order to propagate the embedded input through the initial linear layer, we vectorize the $\mathbb{R}^{k\times 5}$ array row-wise into $\boldsymbol{X} \in \mathbb{R}^{5k}$. For the proof of concept, we assume that the points are ordered. The case of point sets, i.e., including permutations, can be addressed by heuristics or max-pooling over permutations, but is not further considered here. Each intermediate layer output is a one-dimensional array, $\mathbf{z} \in \mathbb{R}^m$, that we embed in $\mathbb{R}^{m+2}$, the same as in MLHP. We illustrate the embeddings in Fig. 1.

We discover that this change of the embedding scheme affects the decision surfaces of the corresponding layer units. Namely, for a given (embedded and vectorized) input point cloud $\boldsymbol{X} \in \mathbb{R}^{5k}$, a single unit in the first layer with weights $\widetilde{\boldsymbol{S}} \in \mathbb{R}^{5k}$ represents k hyperspheres: one for each 3D point in the original $\mathbb{R}^{k\times 3}$ input. We call such units *geometric neurons* (for a mathematical formulation, see Section 4.2) and our model the *multilayer geometric perceptron* (MLGP). Note that the proposed method works for any dimension other than three and, given a single 3D point as input, the first layer unit is identical to the hypersphere neuron [3].

The embedding, which is non-linear and present at each layer, may eliminate the need for activation functions implied by MLPs. The choice of the final layer activation function depends solely on the application and is no different from the standard MLP case.

4.2. Learned parameter interpretation

Since we regard the model parameters as independent during training, as proposed by [3], our model learns *non-normalized* hyperspheres (parameter vectors) of the form $\widetilde{\boldsymbol{S}} = (s_1, s_2, \ldots, s_{n+2}) \in \mathbb{R}^{n+2}$. We recall that due to the homogeneity of the hypersphere representation (4), both normalized and non-normalized hyperspheres represent the same decision surface.

To analyze the learned decision surfaces in the respective Euclidean space, we need to obtain normalized vectors $\boldsymbol{S}$ as in (8). To achieve this, we perform *point normalization*, i.e., divide all elements in the learned parameter vector $\widetilde{\boldsymbol{S}}$ by the

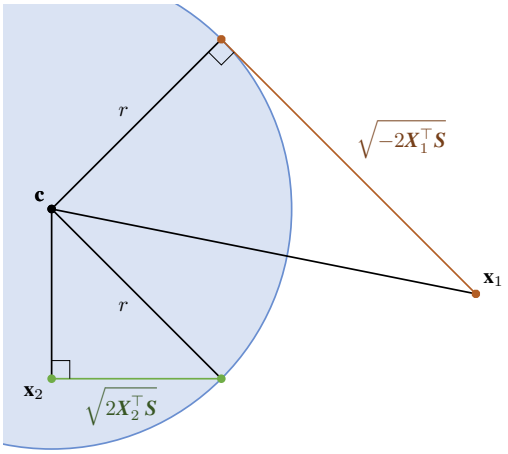

Figure 2. The 2D interpretation of the scalar product of a point X and a sphere S in the conformal space (7). $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ are the Euclidean points $\mathbf{x}_1$ and $\mathbf{x}_2$ embedded in $\mathbb{R}^4 \cong \mathbb{ME}^2$, respectively, and $\boldsymbol{S}$ is the conformal representation of the circle centered at $\mathbf{c}$ and with radius r, described by (6). The scalar product $\boldsymbol{X}^\top \boldsymbol{S}$ determines the cathetus length.

last one. We refer to this last element, s_{n+2}, as the *scale factor*, $\gamma \in \mathbb{R}$. The scale factors can take arbitrary values.

As a result, after training, we can alternatively decompose the geometric neuron output as a weighted sum of the scalar products of k embedded input points and k learned hyperspheres that the neuron represents and that are normalized. This decomposition allows interpreting the decision-making process in the Euclidean space. To fully appreciate this mental picture, the reader is encouraged to refer to Fig. 1.

Thus, we define the geometric neuron mathematically as

$$z = \sum_{i=1}^{k} \gamma_i \, \boldsymbol{X}_i^\top \boldsymbol{S}_i \;, \tag{10}$$

where $z \in \mathbb{R}$ is the output of the geometric neuron, $\boldsymbol{X}_i \in \mathbb{R}^5$ is the i^{th} row in the point-wise embedded input $\mathbf{X} \in \mathbb{R}^{k \times 5}$, and $\boldsymbol{S}_i = \widetilde{\boldsymbol{S}}_i / \gamma_i \in \mathbb{R}^5$ are the corresponding normalized learned parameters (hyperspheres). Each $\boldsymbol{X}_i$ and $\boldsymbol{S}_i$ are of the form shown by (8).

We discuss the important effect of having negative scale factors in a later section. Also, radii of hyperspheres can be extracted from their normalized form (8), namely from the second last element. However, as all other parameters, this element can be learned freely. It can even become negative, representing a hypersphere with an imaginary radius. Although lacking geometric interpretation, this can be beneficial for the learning process [3].

4.3. Activation isometry in 3D

An exciting property of the geometric neuron is that the rigid body transformations are isometries for its activations. There are two ways to show this.

First, consider transformations in the conformal space. We refer to the summary of motion operators and related conformal geometric algebra computations provided in [9].

Consider a motor M (equation (35) in [9]), i.e., rotation followed by translation in the conformal space, operating on a general conformal object O, e.g., a point X or a sphere S. To describe this transformation, we need to use the *sandwich product*:

$$O' = M^{-1} O M \;, \tag{11}$$

where M^{-1} is the inverse of the motor such that $M^{-1}M = MM^{-1} = 1$, and O' is the transformed object.

Suppose we now want to apply a motor M to both a point X and a sphere S in the conformal $\mathbb{ME}^3$ space and compute their scalar product (7). By plugging (11) into (2), we get

$$\begin{aligned} X' \cdot S' &= \frac{X'S' + S'X'}{2} \\ &= \frac{M^{-1}XM\,M^{-1}SM + M^{-1}SM\,M^{-1}XM}{2} \\ &= \frac{M^{-1}XSM + M^{-1}SXM}{2} \\ &= M^{-1}\frac{XS + SX}{2}M = X \cdot S \;. \end{aligned} \tag{12}$$

Given a rigid body transformation defined by the rotation $\mathbf{R} \in \mathrm{SO}(3)$ and translation $\mathbf{t} \in \mathbb{R}^3$, the sandwich product (11) of the corresponding motor M and a sphere S in the conformal $\mathbb{ME}^3$ space is isomorphic to the matrix-vector product with the matrix operator $\boldsymbol{M}_S$ on $\mathbb{R}^5$ defined as

$$\boldsymbol{M}_S = \begin{bmatrix} \mathbf{R} & \mathbf{0} & \mathbf{t} \\ \mathbf{t}^\top \mathbf{R} & 1 & \frac{1}{2}\mathbf{t}^2 \\ \mathbf{0}^\top & 0 & 1 \end{bmatrix} \;. \tag{13}$$

The correctness of the isomorphism follows from computing

$$\boldsymbol{M}_S \boldsymbol{S} = \begin{bmatrix} \mathbf{Rc} + \mathbf{t} \\ \mathbf{t}^\top \mathbf{Rc} + \frac{1}{2}(\mathbf{c}^2 - r^2) + \frac{1}{2}\mathbf{t}^2 \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{Rc} + \mathbf{t} \\ \frac{1}{2}((\mathbf{Rc} + \mathbf{t})^2 - r^2) \\ 1 \end{bmatrix} \;. \tag{14}$$

To the best of our knowledge, this result has not been presented in any related work.

For the point X embedded in the conformal $\mathbb{ME}^3$ space as shown in (8), the corresponding operator $\boldsymbol{M}_X$ on $\mathbb{R}^5$ differs slightly, as elements 4 and 5 are swapped and negated

$$\boldsymbol{M}_X = \begin{bmatrix} \mathbf{R} & -\mathbf{t} & \mathbf{0} \\ \mathbf{0}^\top & 1 & 0 \\ -\mathbf{t}^\top \mathbf{R} & \frac{1}{2}\mathbf{t}^2 & 1 \end{bmatrix} \;, \tag{15}$$

such that $\boldsymbol{M}_X$ is the adjoint of $\boldsymbol{M}_S$, i.e., $\boldsymbol{M}_X^\top \boldsymbol{M}_S = \boldsymbol{I}_5$.

Recalling the scalar product isomorphism between $\mathbb{ME}^n$ and $\mathbb{R}^{n+2}$ (9) and using the result (12), we can see that applying the isomorphic operators $\boldsymbol{M}_X$ and $\boldsymbol{M}_S$ to both each

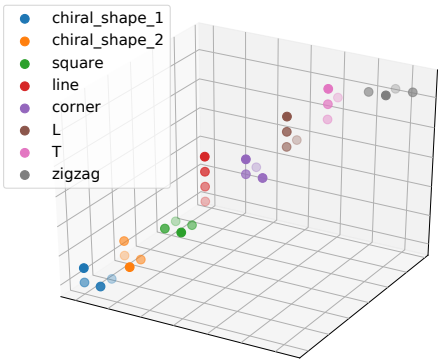

Figure 3. The 3D Tetris data.

point $\boldsymbol{X}_i$ and the respective sphere $\boldsymbol{S}_i$ in (10) does not change the output of the geometric neuron since

$$(\boldsymbol{M}_X\boldsymbol{X})^\top(\boldsymbol{M}_S\boldsymbol{S}) = X'\cdot S' = X\cdot S = \boldsymbol{X}^\top\boldsymbol{S} = \boldsymbol{X}^\top\boldsymbol{M}_X^\top\boldsymbol{M}_S\boldsymbol{S}\,. \tag{16}$$

Another way to show this is to recall the Euclidean space interpretation of the scalar product in the conformal space (7), which we illustrate by Fig. 2 for the 2D geometry. Considering the cases when a point is either inside or outside the circle, we can construct the triangles accordingly. Applying rigid body transformations to both the circle and the point will preserve the triangles and hence the distance determined by the scalar product $\boldsymbol{X}^\top\boldsymbol{S}$ (i.e., the cathetus length). Thus, the geometric neuron decomposed as the sum of the scalar products (10) is indeed isometric in the corresponding Euclidean space. We are not aware of any such graphical explanation previously presented in the literature.

In the following sections, we experimentally demonstrate this property and discuss its implications.

5. Experiments

To demonstrate the explainability of our method, we test it on a geometry classification problem. We also compare its performance with those of analogous baseline MLHP and vanilla MLP.

5.1. 3D shape classification data

We use the 3D Tetris dataset proposed in [21]. It consists of 8 shapes, displayed in Fig. 3. Each data sample is a 4×3 array, containing the 3D coordinates of four points in a certain order. We refer to these 3D coordinates as canonical. Note that the dataset includes two chiral shapes that are reflections of each other.

Main dataset. For the first experiment, we augment the Tetris data by performing uniform random rotation in $[0, 2\pi)$ (about random axis) and translation in $(-3, 3)$, i.e., rigid body transformation of the canonical shapes. This way, we form a training set consisting of 1000 shapes, a validation set containing 9000 samples, and a test set of size 90000.

Theta-split. Additionally, we create a *theta-split* dataset of the same size to see if the models in our comparison can generalize over rigid transformations. The rotation angle, θ, in the dataset construction differs for the training and validation/test sets: θ_{train} is drawn from the uniform distribution over the joint interval $\left[0, \frac{\pi}{4}\right) \cup \left[\pi, \frac{5\pi}{4}\right)$ and θ_{val} and θ_{test} from the antipodal interval. The translation vector is drawn as in the main dataset construction.

Data with noise. An important practical consideration for model comparison is that real-world data often contain a certain amount of noise. Therefore, we add distortion, $\boldsymbol{n}$, of different levels to the shape coordinates in the main and theta-split datasets: $\boldsymbol{n} \sim U(-a, a)$ with $a \in \{0.1,\, 0.2\}$. We thus obtain four additional datasets.

5.2. Setup

When building models for the experiments, we want the total number of parameters to be comparable, even though a quantitative comparison of the methods is beyond the main focus of our work. The decision surfaces in our (MLGP) and the baseline (MLHP) models are of a higher order of complexity than the vanilla MLP case. As a consequence, they have a different number of hidden units. We select the vanilla model with 6 hidden units (134 parameters), the baseline MLHP model with 5 hidden units (126 parameters), and our MLGP with 4 hidden units (128 parameters). Note that the vanilla model includes bias parameters, whereas the other two do not.

We try different activation functions for all models: the sigmoid, hyperbolic tangent (tanh), ReLU, and identity, i.e., no activation function. In the case of MLGP and MLHP, identity means that the only source of non-linearity is the embedding. The final layer of all models in our experiments is equipped with the softmax activation function. We implement all models in PyTorch [15] and use the default parameter initialization for linear layers. We train the models for 20000 epochs by minimizing the cross-entropy loss function with the Adam optimizer [10] supplied with the default hyperparameters: the learning rate is set to 0.001, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. We run each experiment 50 times. At each run, we generate the datasets (training and validation sets) described in Section 5.1. The test data are generated once for each experiment.

We train and test the models on the main and theta-split datasets, both with different levels of noise. Since the three vanilla models with the identity, sigmoid, and tanh activation functions, respectively, are inferior to that with ReLU, we show only the latter case and proceed with ReLU as an activation function for the vanilla model. Our MLGP method and the baseline model perform much better without activation functions, which motivates us to use this configuration. The performances of the models on the test data and in all experiments are presented in Table 1.

Table 1. Model accuracies on the *test* data (mean and std over 50 runs, %); values in parentheses represent the accuracy of the 10 best models selected based on the *validation* accuracy.

	Main dataset			Theta-split		
Noise	$a = 0.0$	$a = 0.1$	$a = 0.2$	$a = 0.0$	$a = 0.1$	$a = 0.2$
MLP	71.7 ± 4.0 (78.0 ± 2.6)	65.3 ± 4.3 (71.7 ± 4.4)	60.7 ± 3.2 (65.9 ± 2.8)	55.3 ± 5.8 (63.8 ± 3.1)	51.8 ± 3.4 (56.5 ± 2.2)	48.4 ± 3.6 (53.3 ± 1.9)
Baseline [2]	$\mathbf{92.5 \pm 0.4}$ $(\mathbf{92.8 \pm 0.2})$	81.3 ± 3.4 (86.9 ± 0.4)	68.2 ± 3.2 (72.2 ± 0.5)	87.6 ± 0.6 (88.3 ± 0.6)	78.4 ± 3.8 (83.8 ± 1.8)	65.7 ± 3.7 (70.4 ± 0.7)
Ours	91.8 ± 2.2 (92.5 ± 0.2)	$\mathbf{81.8 \pm 5.9}$ $(\mathbf{89.3 \pm 0.4})$	$\mathbf{69.7 \pm 8.3}$ $(\mathbf{79.6 \pm 0.2})$	$\mathbf{87.9 \pm 0.8}$ $(\mathbf{89.0 \pm 0.4})$	$\mathbf{80.3 \pm 5.1}$ $(\mathbf{87.5 \pm 0.5})$	$\mathbf{68.2 \pm 8.4}$ $(\mathbf{79.4 \pm 0.3})$

Table 2. Isometry test: the pretrained (original) and transformed MLGP accuracies on the original and transformed *test* split from the main dataset (mean and std over 10000 runs, %).

	MLGP	Transformed MLGP
Original data	$\mathbf{91.98 \pm 0.00}$	86.11 ± 3.46
Transformed data	86.09 ± 3.45	$\mathbf{91.98 \pm 0.00}$

5.3. Isometry test

To show that the rigid body transformations are isometries for the geometric neuron activations as claimed in Section 4.3, we design the following experiment. Given an instance of our MLGP model pretrained on the main data (see Section 5.1), we generate a rigid body transformation in $\mathbb{R}^3$ and apply it to the weights of the geometric neurons in the model by means of the isomorphism (13). This results in a *transformed* model. Subsequently, we compare the performance of the original model on the original test set and the performance of the transformed model on the test set modified by the generated transformation. Our hypothesis is that the two results will be indistinguishable up to a numerical precision if the geometric neuron activations are indeed isometric on $\mathbb{R}^3$. Table 2 summarizes the experiment outcome. For the sake of completeness, we also evaluate the original model on the transformed test set and the transformed model on the original test set.

6. Discussion and conclusion

In this section, we discuss the major benefits of the proposed MLGP model: the exlplainability of coefficients and the improved quantitative results, in particular when extrapolating transformation parameters.

6.1. Model explainability

One of the main advantages of the embedding scheme utilized in our model is that it provides an intuitive geometrical interpretation of the learned coefficients. To visualize the learned decision surfaces in the Euclidean $\mathbb{R}^3$ space, we need to point-normalize them according to (8).

We use two shapes from the main dataset described in Section 5.1 as input to the trained MLGP model. We demonstrate the input shapes and the four spherical decision surfaces represented by the (fourth) hidden unit in Fig. 4, wherein each spherical decision surface classifies the corresponding point of the input shape. For clarity, we show how the fourth decomposed sphere of a single hidden unit in the same model distinguishes the respective points of the two input shapes in Fig. 5.

Note that each sphere may have a different scale factor. It can be negative and, thanks to the normalization step, turn the decision surface inside out for a given input. This phenomenon is discussed but not visualized in the prior work. Importantly, this swap is itself a conformal transformation. Suppose the sign of the scalar product (9) of an embedded input point $\boldsymbol{X}$ and a sphere $\boldsymbol{S}$ is the same regardless of the normalization of the latter. In that case, the input is categorized as class $\mathcal{I}$ if it is inside or on the surface of the sphere, and to class $\mathcal{O}$ if it is outside. We refer to such hyperspherical classifiers as $\mathcal{I}$-*hyperspheres* and $\mathcal{O}$-*hyperspheres*, respectively. We illustrate the idea of the inverted decision surfaces by drawing $\mathcal{O}$-spheres in red, whereas $\mathcal{I}$-spheres are shown in blue (see Fig. 4 and Fig. 5).

6.2. Quantitative results

The superiority of our MLGP model with no activation function other than the embedding and even fewer number of parameters (128 vs. 134) to the plain MLP is evident from Table 1. This result confirms the intuition from Section 4, given that our model units have higher-order decision surfaces. However, it comes at the cost of increased computational complexity. Considering that we have to evaluate the magnitude of m vectors at each layer in the embedding step, we can roughly compare it to adding m extra neurons to the respective layer in an analogous vanilla MLP, in accordance with the complexity analysis given in [3]. Compared to the MLP, our model has an increased computational complexity with a factor of typically between 1.6 and 2.

What is remarkable is the embedding being non-linear and present at each layer allows for successful learning without any activation function. To the best of our knowledge, this has not been observed in prior work.

In all noisy data experiments, our MLGP demonstrates, on average, better generalization than the baseline and vanilla models (see Table 1). We noted variations of the validation accuracy, presumably due to confusing two or three classes, and selected the ten models of each type with best validation

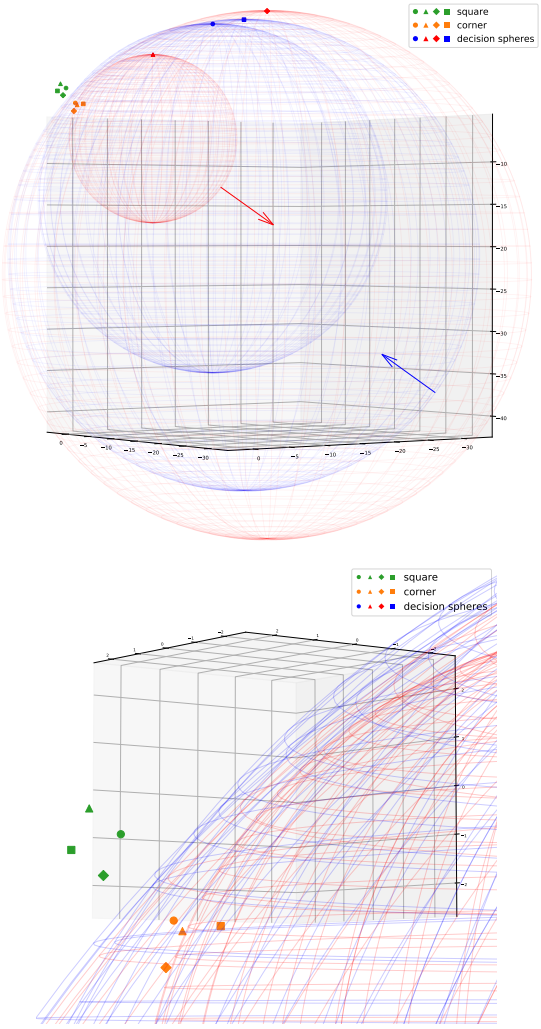

Figure 4. A single unit in the first layer (i.e., geometric neuron) in our MLGP model classifying the Tetris shapes: each spherical decision surface classifies the corresponding point of the input shape (specified by the markers); the unit output is then a linear combination of the scalar products (10). Top: the arrows specify the positive direction of the scalar product, i.e., inside or outside the sphere. Bottom: a zoomed-in view.

accuracy. Our method has the best correlation of validation accuracy and test accuracy, which further increases its advantage and reduces its variance.

We notice a major drop in the generalization performance of the vanilla MLP in the case of the theta-split data and a smaller, yet significant, drop for the baseline, whereas the generalization accuracy of our method decreases insignificantly, as indicated by Table 1. To a certain extent, this suggests that hyperspherical decision surfaces are better suited for such geometry tasks than standard hyperplanes. This has not been discussed in prior work.

Overall, the experiments show that in addition to being more geometrically motivated and explainable, our modification to the baseline MLHP, the MLGP, produces favorable

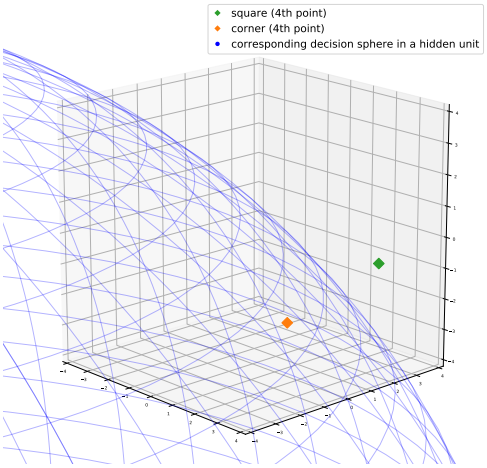

Figure 5. One of the spheres in a single geometric neuron; the sphere classifies the corresponding point in the input shapes: the scalar product of one point of the *corner* and the displayed normalized sphere ≈ 39.15 (positive), whereas that of the point of the *square* is ≈ -50.75 (negative).

quantitative results classifying the 3D Tetris shapes and is even superior when they are perturbed.

6.3. Isometry implications

The result shown in Section 4.3 is verified in Section 5.3: when the geometric neuron parameters are transformed the same way as the input, the neuron activations remain unchanged, hence the model output and the performance of the model, as presented in Table 2. Although this result may appear mathematically trivial, its beauty is that we can apply it to a very complex set of model parameters since we know they follow the geometrically correct algebraic structure (10) by construction with the proposed embedding scheme in our MLGP method. Needless to say, the baseline MLHP input embedding would not allow for such manipulations simply because the $\mathbb{R}^3$ transformations of the input shapes would not be applicable to the $\mathbb{R}^{12}$ hyperspheres. The same argument applies to the vanilla MLP, since a hyperplane can be seen as a special case of a hypersphere (i.e., with infinite radius) [18], and therefore, the MLHP is a generalization of the standard perceptron model.

The isometry property of the geometric neuron is a necessary condition to consider rotation and translation equivariance (discussed in, e.g., [21]) capabilities of our model. This is, however, outside the scope of this paper.

Acknowledgments

This work was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP), by the Swedish Research Council through a grant for the project Algebraically Constrained Convolutional Networks for Sparse Image Data (2018-04673), and the strategic research environment ELLIIT.

References

[1] Brandon Anderson, Truong Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. In *Advances in Neural Information Processing Systems*, pages 14510–14519, 2019.

[2] Vladimir Banarer, Christian Perwass, and Gerald Sommer. Design of a multilayered feed-forward neural network using hypersphere neurons. In *International Conference on Computer Analysis of Images and Patterns*, pages 571–578. Springer, 2003.

[3] Vladimir Banarer, Christian Perwass, and Gerald Sommer. The hypersphere neuron. In *ESANN*, pages 469–474, 2003.

[4] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[5] Sven Buchholz and Gerald Sommer. A hyperbolic multilayer perceptron. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 2, pages 129–133. IEEE, 2000.

[6] Sven Buchholz and Gerald Sommer. On Clifford neurons and Clifford multi-layer perceptrons. *Neural Networks*, 21(7):925–935, 2008.

[7] Russell Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43. Ieee, 1995.

[8] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[9] Eckhard Hitzer. Geometric operations implemented by conformal geometric algebra neural nodes. *Proc. SICE Symposium on Systems and Information 2008, 26-28 Nov. 2008, Himeji, Japan*, pages 357–362, 2008. arXiv preprint arXiv:1306.1358.

[10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[11] Yasuaki Kuroe. Models of Clifford recurrent neural networks and their dynamics. In *The 2011 International Joint Conference on Neural Networks*, pages 1035–1041. IEEE, 2011.

[12] Hongbo Li, David Hestenes, and Alyn Rockwood. Generalized homogeneous coordinates for computational geometry. In *Geometric Computing with Clifford Algebras*, pages 27–59. Springer, 2001.

[13] Hongbo Li, David Hestenes, and Alyn Rockwood. A universal model for conformal geometries of Euclidean, spherical and double-hyperbolic spaces. In *Geometric computing with Clifford algebras*, pages 77–104. Springer, 2001.

[14] Hod Lipson and Hava T. Siegelmann. Clustering irregular shapes using high-order neurons. *Neural Computation*, 12(10):2331–2353, 2000.

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[16] J.K. Pearson and D.L. Bisset. Back propagation in a Clifford algebra. In *Proc. Int. Conf. Artificial Neural Networks, I. Aleksander and J. Taylor (Ed.)*, volume 2, page 413–416, 1992.

[17] JK Pearson and DL Bisset. Neural networks in the Clifford domain. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 3, pages 1465–1469. IEEE, 1994.

[18] Christian Perwass, Vladimir Banarer, and Gerald Sommer. Spherical decision surfaces using conformal modelling. In *Joint Pattern Recognition Symposium*, pages 9–16. Springer, 2003.

[19] Juan Pablo Serrano-Rubio, Arturo Hernández-Aguirre, and Rafael Herrera-Guzmán. Hyperconic multilayer perceptron. *Neural Processing Letters*, 45(1):29–58, 2017.

[20] Richard W Sharpe. *Differential geometry: Cartan's generalization of Klein's Erlangen program*, volume 166. Springer Science & Business Media, 2000.

[21] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

[22] Carlos Villaseñor, Nancy Arana-Daniel, Alma Y Alanis, and Carlos Lopez-Franco. Hyperellipsoidal neuron. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 788–794. IEEE, 2017.

[23] Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco S Cohen. 3d steerable CNNs: Learning rotationally equivariant features in volumetric data. In *Advances in Neural Information Processing Systems*, pages 10381–10392, 2018.

[24] Julio Zamora-Esquivel. G 6, 3 geometric algebra; description and implementation. *Advances in Applied Clifford Algebras*, 24(2):493–514, 2014.