Clifford Flows

Francesco Alesiani[*]
NEC Laboratories Europe
francesco.alesiani@neclab.eu

Takashi Maruyama[*]
NEC Laboratories Europe
takashi.maruyama@neclab.eu

Abstract

Geometric machine learning incorporates geometric priors when modeling physical systems, as particle or molecular systems. Clifford Algebra extends Euclidean vector space by introducing algebraic structure and thus represents an appealing tool to model geometrical features. An example of this model is the Clifford neural network, an equivariant neural network based on Clifford Algebra. When modeling distributions over geometric objects using Clifford Algebra, we need to define how these distributions transform. We thus introduce probability density function over Clifford algebra and their transformation based on gradients of functions defined over Clifford Algebra. Here we show that the gradient of functions between Clifford algebras on Euclidean spaces induces the canonical gradient of the functions restricted to the base vector spaces. This ensures that the gradient of Clifford neural networks coincides with that obtained through widely adopted automatic differentiation modules such as Autograd. We empirically evaluate the benefit of the gradient of Clifford neural networks and the transformation of distribution over Clifford Algebra for the problem of sampling from distributions in scientific discovery.

1 Introduction

Clifford neural networks [4, 6, 31, 36, 45, 48, 51], a class of geometric deep learning models [7], have made promising progress in modeling the inherent interactions of physical systems, such as fluid dynamics [4] and multibody interaction systems [45, 6], or geometrical quantities [5]. Clifford neural networks have been applied to solve physical systems described by partial differential equations (PDEs) [4, 51] or ordinary differential equations (ODEs) [6, 31, 45, 48]. The Clifford neural networks are extremely effective in solving these equations since only a gradient with respect to the parameters of the neural network is required, or *forward problem*. Other classes of problems associated with physical systems require computing the gradient of the neural network with respect to their *input*. Example applications include inverse-design [2, 50], flow-matching [8, 27], and normalizing flow [39, 46]. We have a proper understanding of the forward manipulation of elements in the Clifford Algebra, however, we believe that the notion of *differentiability* of Clifford neural networks with respect to the Clifford Algebra and the definition of *probability distributions* over Clifford Algebras have not been sufficiently understood.

Contributions. As a first step towards the application of Clifford neural networks to transformation of probability distributions, i) we propose to interpret functions between Clifford algebras as continuous functions between metric spaces; ii) we elucidate the differentiability of the functions by observing the gradient of the functions is equivalent to the natural and canonical gradient of functions on Euclidean spaces. As a corollary of the observation, iii) we also show that the gradient of Clifford neural networks is compatible with that of the functions restricted on their base vector spaces, which eventually ensures the validity of the usage of automatic differentiation modules such as Autograd [41] to obtain the gradient of Clifford neural networks; iv) we introduce the definition of a probability

[*]Equal contribution

Machine Learning and the Physical Sciences Workshop, NeurIPS 2024.

distribution over Clifford Algebra based on this relationship; v) we introduce a new transformation between probability distribution over the Clifford Algebras and the correspondent architecture, *Clifford NVP*; and vi) we empirically validate the use of the gradient of Clifford neural networks to model distribution changes with two experiments. The code for the experiments is provided in `https://github.com/nec-research`.

2 Background

Clifford Algebra. We start by introducing Clifford algebra [35], also known as geometric algebra [23], over a real vector space V of finite dimension n, and some of its key properties. We follow similar notation and definition as in [45, 51]. The Clifford algebra $Cl(V, \mathfrak{q})$ with a quadratic form $\mathfrak{q} : V \to \mathbb{R}$ is a vector space generated by the l-fold tensor product of a basis $\{\boldsymbol{e}_i\}_{i=1}^n$ of V with an equivalence relation $\mathfrak{q}(\boldsymbol{v}) = \boldsymbol{v} \otimes \boldsymbol{v}$ $(\forall \boldsymbol{v} \in V)$. Then, every element $\boldsymbol{x} \in Cl(V, \mathfrak{q})$ may be written with finite indices $I_m = \{i_1 < \cdots < i_m\} \subset \{1, 2, \cdots, n\}$

$$\boldsymbol{x} = \sum_{m=0}^{n} \sum_{I_m} x_{I_m} \boldsymbol{e}_{i_1} \otimes_{\mathfrak{q}} \cdots \otimes_{\mathfrak{q}} \boldsymbol{e}_{i_m}, \quad x_{I_m} \in \mathbb{R}. \tag{1}$$

Note that $I_m = \emptyset$ for $m = 0$. The expression $\boldsymbol{v} \otimes_{\mathfrak{q}} \boldsymbol{w}$ of elements $\boldsymbol{v}, \boldsymbol{w} \in V$ represents the *geometric product* of $\boldsymbol{v}, \boldsymbol{w}$, which defines a product on $Cl(V, \mathfrak{q})$ and charactrizes $Cl(V, \mathfrak{q})$ as an algebra. The product of $\boldsymbol{x}, \boldsymbol{y} \in Cl(V, \mathfrak{q})$ runs all the pair of $\boldsymbol{e}_{i_1} \otimes_{\mathfrak{q}} \cdots \otimes_{\mathfrak{q}} \boldsymbol{e}_{i_m}$ composing respective $\boldsymbol{x}$ and $\boldsymbol{y}$, but some of the basis elements $\boldsymbol{e}_i$ is reduced to a scalar because of the relation $\mathfrak{q}(\boldsymbol{e}_i) = \boldsymbol{e}_i \otimes_{\mathfrak{q}} \boldsymbol{e}_i$,

$$(\boldsymbol{e}_{i_1} \otimes_{\mathfrak{q}} \cdots \otimes_{\mathfrak{q}} \boldsymbol{e}_{i_r}) \otimes_{\mathfrak{q}} (\boldsymbol{e}_{j_1} \otimes_{\mathfrak{q}} \cdots \otimes_{\mathfrak{q}} \boldsymbol{e}_{j_s}) = \prod_{u=0}^{t-1} \mathfrak{q}(e_{k_{r+s-u}})(\boldsymbol{e}_{k_1} \otimes_{\mathfrak{q}} \cdots \otimes_{\mathfrak{q}} \boldsymbol{e}_{k_{r+s-t}}). \tag{2}$$

Clifford Neural Networks. Taking advantage of the flexible manipulation of geometric quantities through the algebraic representation, Clifford algebra is incorporated into various kinds of machine-learning models. Such models include Fourier neural operators [4], message passing neural networks (MPNNs) [45], simplicial MPNNs [31], multilayer perceptron models [36], convolutional neural networks [51], and transformers [6]. Typical building blocks of these neural networks form the algebra $\mathbb{R}[X_1, X_2, \cdots, X_c]$ of polynomials in coefficients of $\mathbb{R}$ (of any order) with c variables. The sum and product of $\mathbb{R}[X_1, X_2, \cdots, X_c]$ are defined as those of $Cl(\mathbb{R}^n, \mathfrak{q})$, which also serve as a map from the product space of Clifford algebras (of channel dimension c) to the Clifford algebra:

$$\underbrace{Cl(\mathbb{R}^n, \mathfrak{q}) \times \cdots \times Cl(\mathbb{R}^n, \mathfrak{q})}_{c} \xrightarrow{\boldsymbol{F}} Cl(\mathbb{R}^n, \mathfrak{q}), \quad F \in \mathbb{R}[X_1, X_2, \cdots, X_c].$$

3 Transformation of probability density functions over Clifford algebras

To introduce a transformation of distributions between Clifford Spaces, we define the differentiability of functions between Clifford algebras. **Differentiable function on Clifford algebra**. Let $\mathfrak{g}_V$ be an Euclidean metric for V, i.e., a symmetric, non-degenerate and positive bilinear form $\mathfrak{g}_V : V \times V \to \mathbb{R}$. The metric induces a metric $\mathfrak{g}_{Cl(V,\mathfrak{q})}$ on $Cl(V, \mathfrak{q})$ of dimension 2^n. With this induced metric, the $\boldsymbol{a}$-directional gradient of $\boldsymbol{F}$ at $\boldsymbol{x}_0 \in Cl(V, \mathfrak{q})$ in the direction $\boldsymbol{a} \in Cl(V, \mathfrak{q})$ is defined as

$$\boldsymbol{F}'_{\boldsymbol{a}}(\boldsymbol{x}_0) = \lim_{\lambda \to 0} \frac{\boldsymbol{F}(\boldsymbol{x}_0 + \lambda \boldsymbol{a}) - \boldsymbol{F}(\boldsymbol{x}_0)}{\lambda}, \quad \boldsymbol{a} \in Cl(V, \mathfrak{q}). \tag{3}$$

The original definition is given in [11, 22]. Here, the distance of the space, used when taking infinitely small λ, is defined by a norm $||\boldsymbol{x}|| = \sqrt{\mathfrak{g}_{Cl(V,\mathfrak{q})}(\boldsymbol{x}, \boldsymbol{x})}$. We call $\boldsymbol{F}$ *differentiable* when the limit exists for any directional vector $\boldsymbol{a} \in Cl(V, \mathfrak{q})$ and $\boldsymbol{x}_0$ (and its associated gradient is continuous). Another prerequisite for this notion is detailed in Appendix B.

Connection on gradients between base space and associated Clifford algebra The quadratic form $\mathfrak{q}$, as defined in Section 2, naturally defines a bilinear form $\mathfrak{b}(v, w) = \frac{1}{2}(\mathfrak{q}(\boldsymbol{v} + \boldsymbol{w}) - \mathfrak{q}(\boldsymbol{v}) - \mathfrak{q}(\boldsymbol{w})) : V \times V \to \mathbb{R}$. Throughout the rest of this paper, we assume the bilinear form $\mathfrak{b}$ to be an inner product of signature (p, q, r), i.e., $\mathfrak{b}(\boldsymbol{v}, \boldsymbol{w}) = \mathfrak{b}^{p,q,r}(\boldsymbol{v}, \boldsymbol{w}) = \boldsymbol{v}^{\mathrm{T}} \Delta^{p,q,r} \boldsymbol{w}$ with matrix $\Delta^{p,q,r} = \mathrm{diag}(\underbrace{1, \cdots, 1}_{p}, \underbrace{-1, \cdots, -1}_{q}, \underbrace{0, \cdots, 0}_{r})$, which leads to the following equivalent relations; if $1 \leq i \leq p$, then $\mathfrak{q}(\boldsymbol{e}_i) = +1$, if $p+1 \leq i \leq p+q$, then $\mathfrak{q}(\boldsymbol{e}_i) = -1$, while if $p+q+1 \leq i \leq p+q+r$ we have that $\mathfrak{q}(\boldsymbol{e}_i) = 0$. We also denote $\mathbb{R}[X_1, \cdots, X_c]_{p,q,r}$ as the set of polynomial functions on the Clifford algebra whose geometric product $\otimes_{\mathfrak{q}}$ is associated with the bilinaer form with signature (p, q, r). We claim that all functions in $\mathbb{R}[X_1, \cdots, X_c]_{p,q,r}$ are differentiable for **any** signatures (p, q, r). The claim can be seen as an extension of results in [11, 22]. The formal claim is given as Proposition D.1 in Appendix D. With Proposition D.1, we further elaborate the connection between gradients of functions between Clifford spaces and its base spaces: Suppose that we have the following embedding (inc) and projection (proj) maps between V and $Cl(V, \mathfrak{q})$:

$$\text{inc} : V \hookrightarrow Cl(V, \mathfrak{q}),\ \boldsymbol{v} \mapsto \sum_{k=1}^{n} \mathfrak{g}_V(\boldsymbol{v}, \boldsymbol{e}_k)\boldsymbol{e}_k,\ \ \text{proj} : Cl(V, \mathfrak{q}) \twoheadrightarrow V,\ \sum_{m=0}^{n} \sum_{I_m} v_{I_m} \boldsymbol{e}_{I_m} \mapsto \sum_{I_1} v_{I_1} \boldsymbol{e}_{I_1}.$$

Corollary 3.1. *For $\forall \boldsymbol{F} \in \mathbb{R}[X]_{p,q,r}$, its restriction to the base space V by* inc *and* proj

$$V \xhookrightarrow{\text{inc}} Cl(V, \mathfrak{q}) \xrightarrow{\boldsymbol{F}} Cl(V, \mathfrak{q}) \xrightarrow{\text{proj}} V$$

is a differentiable function between V with respect to the metric g_V. In particular, when $V = \mathbb{R}^n$ and its basis is the standard orthonormal basis, the function $\text{proj} \circ F \circ \text{inc}$ is differentiable on $\mathbb{R}^n$ with respect to the canonical differentiable structure on Euclidean space.

This corollary ensures that the gradient of $\text{proj} \circ \boldsymbol{F} \circ \text{inc}$ is the "standard" gradient defined on the Euclidean spaces, as obtained through an automatic differentiation module such as Autograd [41].

Coordinate system and Jacobian matrix of differentiable Clifford functions When modeling continuous normalizing flow, we need the definition of a probability distribution over the Clifford Algebra. We consider thus the special case of Corollary 3.1 with $V = \mathbb{R}^{2^n}$, which defines an isomorphism with $Cl(\mathbb{R}^n, \mathfrak{q})$, with $\text{inc}_{\text{coord}}$ and $\text{proj}_{\text{coord}}$ the corresponding mapping operators. The function $\boldsymbol{f} = \text{proj}_{\text{coord}} \circ \boldsymbol{F} \circ \text{inc}_{\text{coord}}$ is defined implicitly from the differentiable function on Clifford algebra $\boldsymbol{F}$. We define the Jacobian of functions between Clifford algebras via the directional gradient. Given a differentiable function $\boldsymbol{F} : Cl(\mathbb{R}^n, \mathfrak{q}) \to Cl(\mathbb{R}^n, \mathfrak{q})$, the directional gradient of the J-th component $\boldsymbol{F}^{(J)}$ in the output space along the direction $\boldsymbol{e}_I$ in the input space is defined as follows:

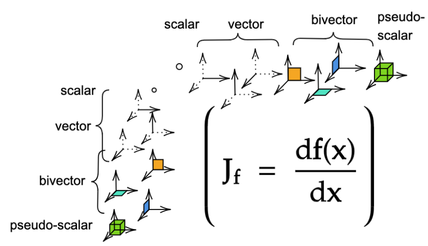

Figure 1: *Clifford Jacobian* ($\boldsymbol{J} = \{\partial_I \boldsymbol{F}^{(J)}\}_{I,J}$) with respect to the coordinate system $\{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3\}$.

$$\partial_I \boldsymbol{F} = \sum_{J} \partial_I \boldsymbol{F}^{(J)} \in Cl(\mathbb{R}^n, \mathfrak{q}),\ \partial_I \boldsymbol{F}^{(J)} = \lim_{\lambda \to 0} \frac{\boldsymbol{F}^{(J)}(\boldsymbol{x} + \lambda \boldsymbol{e}_I) - \boldsymbol{F}^{(J)}(\boldsymbol{x})}{\lambda} \in Cl(\mathbb{R}^n, \mathfrak{q}).$$

We define the Jacobian of $\boldsymbol{F}$ as $\boldsymbol{J}_{\boldsymbol{F}} = (\partial_I \boldsymbol{F})_I = (\partial_1 \boldsymbol{F}, \ldots, \partial_{2^n} \boldsymbol{F})^T \in Cl(\mathbb{R}^n, \mathfrak{q})^{2^n}$, which is related to the the Jacobian of $\boldsymbol{f}$ in the coordinate system through $\text{proj}_{\text{coord}}$ and $\text{inc}_{\text{coord}}$ via $\boldsymbol{J}_{\boldsymbol{f}} = \boldsymbol{J}_{\text{inc}_{\text{coord}}} \boldsymbol{J}_{\boldsymbol{F}} \boldsymbol{J}_{\text{proj}_{\text{coord}}}$, where $\boldsymbol{J}_{\boldsymbol{f}} = \frac{\mathrm{d}\boldsymbol{f}(\boldsymbol{x})}{\mathrm{d}\boldsymbol{x}} = \Big(\big(\partial_1 \boldsymbol{f}^{(1)}, \ldots, \partial_1 \boldsymbol{f}^{(2^n)}\big)^T \ldots \big(\partial_{2^n} \boldsymbol{f}^{(1)}, \ldots, \partial_{2^n} \boldsymbol{f}^{(2^n)}\big)^T \Big) \in \mathbb{R}^{2^n \times 2^n}$.

Density functions over Clifford algebra. Since the Clifford algebra $Cl(\mathbb{R}^n, \mathfrak{q})$ is equipped with the Euclidean scalar metric $\mathfrak{g}_{Cl(\mathbb{R}^n, \mathfrak{q})}$, we have a measure $\mu(\boldsymbol{x})$ on $Cl(\mathbb{R}^n, \mathfrak{q})$, that is equivalent to the canonical measure on $\mathbb{R}^{2^n}$. Through this measure, we define a probability density function $p(\boldsymbol{x})$ on $Cl(\mathbb{R}^n, \mathfrak{q})$ such that $\int_{Cl(\mathbb{R}^n, \mathfrak{q})} p(\boldsymbol{x}) d\mu(\boldsymbol{x}) = 1$. We can then also build the same probability theory on the space of $Cl(\mathbb{R}^n, \mathfrak{q})$ as the Euclidean space, $p_{Cl(\mathbb{R}^n, \mathfrak{q})}(\boldsymbol{x}) = p_{\text{coord}}(\text{proj}_{\text{coord}} \circ \boldsymbol{F}(\boldsymbol{x}))$, and the corresponding change in the probability distribution $\ln p(\boldsymbol{x}_1) = \ln p(\boldsymbol{x}_0) - \ln|\det \boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{x}_0)|$.

Clifford-valued non-volume preserving (Clifford-NVP). Inspired by [10], we propose an extension of Real-NVP to the Clifford Algebra. We therefore propose to transform probability distributions over the Clifford Algebra, and use the algebraic structure of the gradient as we have presented, where the Jacobian matrix for the proposed architecture has closed form. We first split the $2m$ input variables as $\boldsymbol{x}^0, \boldsymbol{y}^0 \in Cl(\mathbb{R}^n, \mathfrak{q})^m$, we then define the transformation element-wise, at the l step as

$$\boldsymbol{x}_i^{l+1} = \boldsymbol{x}_i^l, \quad \boldsymbol{y}_i^{l+1} = \boldsymbol{y}_i^l \exp\{s_{i,\theta}(\boldsymbol{x}^l)\} + \boldsymbol{t}_{i,\psi},(\boldsymbol{x}^l), \quad i = 1, \ldots, m \tag{4}$$

with $s_{i,\theta} : Cl(\mathbb{R}^{(p,q,r)})^m \to \mathbb{R} \subset \mathbb{R}^{(p,q,r)}$, a trainable scalar function of $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)$ and $\boldsymbol{t}_{i,\psi} : Cl(\mathbb{R}^{(p,q,r)})^m \to Cl(\mathbb{R}^{(p,q,r)})$ a trainable translation function. The determinant of the change of variable is therefore $\ln|\det J_{\boldsymbol{x},\boldsymbol{y}}| = \sum_{i=1}^{m} s_{i,\theta}(\boldsymbol{x})$.

4 Experiments

Sampling from distribution. Having introduced the gradients in the Clifford Algebra, we consider continuous normalizing flow, where $\boldsymbol{x}_t$ satisfies the gradient flow equation (Eq.5). (Eq.7):

$$\frac{\partial \boldsymbol{x}_t}{\partial t} = \boldsymbol{f}_t(\boldsymbol{x}_t), \tag{5}$$

$$\frac{\partial \ln p(\boldsymbol{x}_t)}{\partial t} = -\operatorname{tr}\left\{\frac{\partial \boldsymbol{f}_t}{\partial \boldsymbol{x}_t}\right\}, \tag{6}$$

$$\ln p(\boldsymbol{x}_1) = \ln\ p(\boldsymbol{x}_0) - \int_0^1 \operatorname{tr}\left\{\frac{\partial \boldsymbol{f}_t}{\partial \boldsymbol{x}_t}\right\}. \tag{7}$$

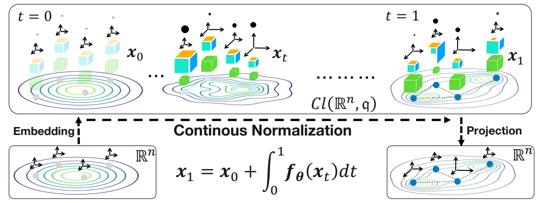

Figure 2: **Schematic of Continuous Normalizing Flow method**. The samples are generated starting from a random noise $\boldsymbol{x}_0 \sim N(0, \boldsymbol{I})$ and integrated using the vector field defined in the Clifford Algebra by the Clifford Neural Network $\boldsymbol{F}$. The log probability is computed using the integral of the Trace of the Jacobian of the transformation.

The associated infinitesimal change of variable (Eq.6) is given by [9] (Theorem.1), and the final sample probability is computed by integrating the infinitesimal change of variables. When the initial samples are drawn from a given distribution $\boldsymbol{x}_0 \sim p_0(\boldsymbol{x})$, the generation process of the final samples $\boldsymbol{x}_1 \sim p_1(\boldsymbol{x})$ is called the continuous normalizing flow. CNF thus requires to compute the trace of the Jacobian, i.e. the gradient with respect to the input variable, $\frac{\partial \boldsymbol{f}_t}{\partial \boldsymbol{x}_t}$.

Table 1: Comparison of the Negative Log Likelihood on the test partition on DW4 and LJ3 dataset.

	DW4 ($n = 2$)		**LJ13** ($n = 3$)	
# training samples	10^2	10^3	10	10^2
E-NF	$8.31^{\pm 0.05}$	$\mathbf{8.15}^{\pm \mathbf{0.10}}$	$33.12^{\pm 0.85}$	$\mathbf{30.99}^{\pm \mathbf{0.95}}$
E-NF (24×2^n)	$\mathbf{8.24}^{\pm \mathbf{0.06}}$	$8.33^{\pm 0.09}$	$\mathbf{31.33}^{\pm \mathbf{0.30}}$	$\mathbf{30.61}^{\pm \mathbf{0.16}}$
CGGNN (24)	$8.80^{\pm 0.32}$	$8.56^{\pm 0.04}$	$\mathbf{31.36}^{\pm \mathbf{0.55}}$	$\mathbf{30.35}^{\pm \mathbf{0.18}}$

We consider a Double Well (DW) and Lennard-Jones (LJ) particle systems, as presented in [28], which model the interactions among particles. **DW4** consists of four particles moving in a 2 dimensional space whose energy depends on a pair of particles. **LJ13** consists of 13 particles and models the potential between molecules as *Lennard-Jones* potentials. Following the experimental setup of [47], we use 10^3 samples for testing and validation, while the training is performed on $10, 10^2$ and 10^3 samples. We compare state-of-the-art E(n)-equivariant flow architectures, whose details are given in Appendix G. Table 1 shows the results of DW4 and LJ13 experiments. We observe that the performance of CNF with Clifford Group-Equivariant GNN (CGGNN) [45] models is better or comparable to the other baselines. We also compare the performance of CGGNN with that of Equivariant Normalizing Flow (E-NF), as proposed in [47], with the increased number of hidden-channel dimensions, to ensure that both of the models have a comparable number of hidden units for a fair comparison. The performance of CGGNN is still comparable to or better than those baselines. These results indicate that the back-propagation through Clifford neural networks can carry informative Jacobian to transform density functions across time.

Table 2: **Comparison of different Normalizing Flow models over the hard-sphere dataset**. Generalization power is highlighted by a lower drop in the log probability over the test data.

	ODD Test	
Model/Algebra	(4, 1, 0)	(3, 0 ,1)
Real NVP [10]	-69.56%	-398.67%
NSF_CL [13]	-235.13%	-259.87%
Clifford NVP [new]	**-1.99%**	**-7.15%**

Normalizing Flow over Clifford Algebra To evaluate the ability to model transformations of distributions over Clifford algebra, we experiment with Normalizing Flows by extending the coupling layers of Real NVP [10] to the Clifford algebra, in which the Jacobian matrix has a closed form, as defined by Equation 4. To compare the new architecture, we consider generating a new dataset, based on hard sphere simulation. Figure 3 shows two snapshots at two different timesteps (sweep) of the Monte Carlo simulation of hard spheres, where some of the spheres move independently (single) and others (connected with lines) moves in rigid-distance pairs. In Table 2, we compare Clifford NVP with Euclidean RealNVP [10] and Neural Spline Flow [13] over the new dataset represented in geometric algebras with signatures $(3, 0, 1)$ and $(4, 1, 0)$ with

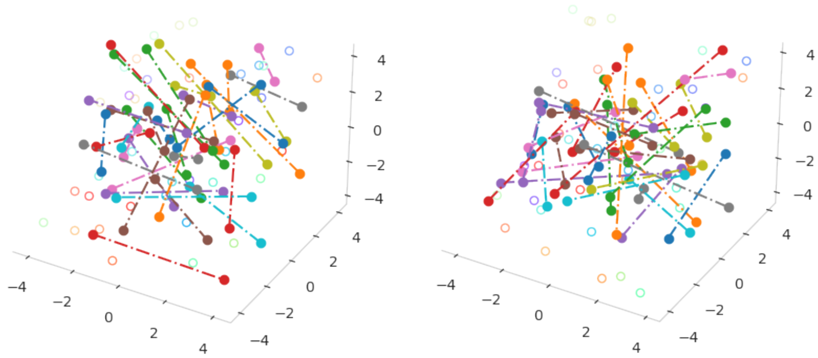

Figure 3: **Visualization of the hard sphere dataset.** Visualization of 100 single and pairs of spheres at different sweeps of the Monte Carlo simulation. We can represent these objects as elements of some Clifford Algebra, in particular spheres as points and pairs as lines (or their dual) in Conformal Geometric Algebra (CGA) with signature $(4,1,0)$ [11] or Projective Geometric Algebra (PGA) with signature $(3,0,1)$ [23].

3d hard sphere Monte-Carlo simulation [38] composed of either isolated spheres or pair of rigidly connected spheres. The task is to evaluate the test dataset in terms of the percentage drop of the log probability. Our results clearly show significant performance gain on the dataset that shows the advantage of Clifford algebra to represent geometric objects.

5 Conclusions

In this paper, we use the gradient of functions between Clifford spaces to model transformation of probability distributions defined over Clifford Algebra. We show that the gradient obtained through Autograd coincides with the analytical gradient. We also provide empirical evidence of the utility of using Clifford algebras in the context of sampling from probability distributions. We hope, that future research would take advantage of the tools defined in the present work and investigate alternative probability distributions properties that are now accessible.

References

[1] Francesco Alesiani. Pytorch geometric algebra, 2024.

[2] Kelsey R Allen, Tatiana Lopez-Guavara, Kim Stachenfeld, Alvaro Sanchez-Gonzalez, Peter Battaglia, Jessica B Hamrick, and Tobias Pfaff. Inverse design for fluid-structure interactions using graph network simulators. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[3] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[4] Johannes Brandstetter, Rianne van den Berg, Max Welling, and Jayesh K Gupta. Clifford neural layers for PDE modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

[5] Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022.

[6] Johann Brehmer, Pim de Haan, Sönke Behrends, and Taco Cohen. Geometric algebra transformer. In *Advances in Neural Information Processing Systems*, volume 37, 2023.

[7] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[8] Ricky T. Q. Chen and Yaron Lipman. Flow matching on general geometries. In *The Twelfth International Conference on Learning Representations*, 2024.

[9] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[10] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017.

[11] Leo Dorst, Daniel Fontijne, and Stephen Mann. *Geometric Algebra for Computer Science: An Object-Oriented Approach to Geometry*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2009.

[12] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[13] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[14] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. *Advances in neural information processing systems*, 32, 2019.

[15] Dumitru Erhan, Y. Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *Technical Report, Univeristé de Montréal*, 01 2009.

[16] Virginia V. Fernández, Antonio M. Moya, and Waldyr A. Rodrigues. Euclidean clifford algebra. *Advances in Applied Clifford Algebras*, 11(3):1–21, Oct 2001.

[17] Virginia V. Fernández, Antonio M. Moya, and Waldyr A. Rodrigues. Extensors. *Advances in Applied Clifford Algebras*, 11(3):23–40, Oct 2001.

[18] Virginia V. Fernández, Antonio M. Moya, and Waldyr A. Rodrigues. Multivector functions of a multivector variable. *Advances in Applied Clifford Algebras*, 11(3):79–91, Oct 2001.

[19] Amir Gholaminejad, Kurt Keutzer, and George Biros. Anode: Unconditionally accurate memory-efficient gradients for neural odes. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 730–736. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[20] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[21] David Hestenes. Hamiltonian mechanics with geometric calculus. In Zbigniew Oziewicz, Bernard Jancewicz, and Andrzej Borowiec, editors, *Spinors, Twistors, Clifford Algebras and Quantum Deformations*, pages 203–214, Dordrecht, 1993. Springer Netherlands.

[22] David Hestenes and Garret Sobczyk. *Clifford algebra to geometric calculus : a unified language for mathematics and physics*. D. Reidel ; Distributed in the U.S.A. and Canada by Kluwer Academic Publishers, Dordrecht; Boston; Hingham, MA, U.S.A., 1984.

[23] David Hestenes and Garret Sobczyk. *Clifford algebra to geometric calculus: a unified language for mathematics and physics*, volume 5. Springer Science & Business Media, 2012.

[24] Robin Kahlow. Jax geometric algebra, 2023.

[25] Robin Kahlow. Tensorflow geometric algebra, 2023.

[26] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[27] Leon Klein, Andreas Krämer, and Frank Noe. Equivariant flow matching. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[28] Jonas Köhler, Leon Klein, and Frank Noe. Equivariant flows: Exact likelihood generative learning for symmetric densities. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5361–5370. PMLR, 13–18 Jul 2020.

[29] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017.

[30] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.

[31] Cong Liu, David Ruhe, Floor Eijkelboom, and Patrick Forré. Clifford group equivariant simplicial message passing networks. In *The Twelfth International Conference on Learning Representations*, 2024.

[32] Jenny Liu, Aviral Kumar, Jimmy Ba, Jamie Kiros, and Kevin Swersky. Graph normalizing flows. *Advances in Neural Information Processing Systems*, 32, 2019.

[33] Yang Liu, Saeed Anwar, Liang Zheng, and Qi Tian. Gradnet image denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[34] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[35] Douglas Lundholm and Lars Svensson. Clifford algebra, geometric algebra, and applications. *arXiv preprint arXiv:0907.5356*, 2009.

[36] Pavlo Melnyk, Michael Felsberg, and Mårten Wadenbäck. Embed me if you can: A geometric perceptron. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1276–1284, October 2021.

[37] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, 2015.

[38] Ángel Mulero. *Theory and simulation of hard-sphere fluids and related systems*, volume 753. Springer, 2008.

[39] Frank Noé, Simon Olsson, Jonas Köhler, and Hao Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457):eaaw1147, 2019.

[40] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[41] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS 2017 Workshop on Autodiff*, 2017.

[42] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[43] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[44] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992.

[45] David Ruhe, Johannes Brandstetter, and Patrick Forré. Clifford group equivariant neural networks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[46] Victor Garcia Satorras, Emiel Hoogeboom, Fabian Bernd Fuchs, Ingmar Posner, and Max Welling. E(n) equivariant normalizing flows. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[47] Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pages 9323–9332. PMLR, 2021.

[48] Matthew Spellings. Geometric algebra attention networks for small point clouds, 2022.

[49] Gerhard Wanner and Ernst Hairer. *Solving ordinary differential equations II*, volume 375. Springer Berlin Heidelberg New York, 1996.

[50] Tailin Wu, Takashi Maruyama, and Jure Leskovec. Learning to accelerate partial differential equations via latent global evolution. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[51] Maksim Zhdanov, David Ruhe, Maurice Weiler, Ana Lucic, Johannes Brandstetter, and Patrick Forré. Clifford-steerable convolutional neural networks, 2024.