Geometric Algebra Transformer Classifier

Paola Carboni
carboni.1811419@studenti.uniroma1.it

Federico Raschiatore
raschiatore.2071132@studenti.uniroma1.it

May 30, 2024

1 Introduction

Deep learning has been effectively applied to a wide range of tasks in various fields. This success is due to deep learning models' ability to capture spatial, temporal, and other important features of the data. However, the learning process can be computationally intensive and time-consuming. By integrating mathematical tools tailored to specific tasks, we can significantly improve the learning process. While traditional vector algebra is effective for many tasks and data types, there are cases where using a mathematical framework designed specifically for geometric data can be advantageous. Geometric algebra provides a concise and efficient way to handle geometric data and perform geometric operations, while preserving rich spatial information and maintaining a clear geometric interpretation. In this paper, we use the expressive power of geometric algebra to develop a transformer (GATr) for a classification task of medical interest. We compare the model's results with those of a standard transformer and a simple CNN network.

2 Data Exploration

The task consists in the classification of two types of artery models: *single* and *bifurcating*. The dataset, first introduced in [2], is composed by 2000 samples for each class, each described by a 3D point cloud `pos`, the indexes of the points at the inlet of the artery segment `inlet_idcs`, the set of triplets of point indexes that define a surfaces of the mesh denoted as `face`, the wall sheer stress vector `wss` and the pressure value associated to each point, `pressure`.

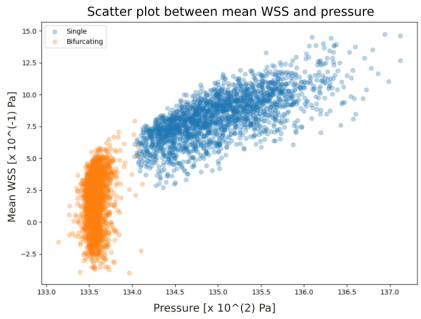

Figure 1: Scatter plot of the data sample between mean WSS and pressure.

Given that the dataset is almost linearly separable based on the `pressure` and mean `wss` features, as shown in Figure 1, we have decided to leave out the `inlet_idcs` feature. This is because `inlet_idcs` is much smaller in size compared to the other features, and it sets a limit on the number of points that can be included in the embedding of each sample.

3 Geometric Algebra Data Embedding

Geometric algebra $G_{3,0,1}$ is defined by a set of four orthogonal generators e_0, e_1, e_2, e_3 which satisfy the property $e_i e_i = 1$ for $i = 1, 2, 3$ and $e_i e_i = 0$ for $i = 0$. The set of all possible exterior product combinations between the

generators define the whole geometric algebra, which is thus characterized by a 16-dimensional space. Each of these dimensions is said to be of grade k if it is generated by the outer product of k different generators. To encode geometric entities defined in the $\mathbb{R}^3$ Euclidean space in this new space, the embedding table provided in [1] and shown in Figure 2 was used. In particular, the `pos` feautures of the dataset were encoded as 3D points, `inlet_idcs` were encoded as scalars, the `wss` vectors were encoded as translations in $\mathbb{R}^3$, and the `pressure` feature was encoded as a scalar. The feature `face`, which contains the indexes of the points that correspond to vertexes of a surface of the mesh, was encoded as an oriented plane. Given the indexes idx_1, idx_2, idx_3, the vertexes of the surface are represented by $P_1 = pos(idx_1)$, $P_2 = pos(idx_2)$, $P_3 = pos(idx_3)$. The normal $\vec{n}$ to the plane is thus computed as

$$\vec{n} = \frac{\overrightarrow{P_1P_2} \times \overrightarrow{P_2P_3}}{||\overrightarrow{P_1P_2} \times \overrightarrow{P_2P_3}||}, \tag{1}$$

where

$$\begin{aligned} \overrightarrow{P_1P_2} &= pos(idx_1) - pos(idx_2) = (x_1 - x_2, y_1 - y_2, z_1 - z_2), \\ \overrightarrow{P_2P_3} &= pos(idx_2) - pos(idx_3) = (x_2 - x_3, y_2 - y_3, z_2 - z_3). \end{aligned} \tag{2}$$

Since `inlet_idcs` was excluded from the final embedding, as discussed in Section 2, each artery sample is represented by 4*`data_length` 16-dimensional multivectors, where `data_length` is the number of points considered for each artery sample. After various trials, the value `data_length` = 50 was selected as the best trade-off between computational time and performance. Each feature of the data has been standardized as

$$std(x) = \frac{x - \bar{x}}{\sigma(x)}, \tag{3}$$

where $\bar{x}$ represents the mean and $\sigma(x)$ represents the standard deviation of feature x for the considered sample.

Object / operator	Scalar	Vector		Bivector		Trivector		PS
	1	e_0	e_i	e_{0i}	e_{ij}	e_{0ij}	e_{123}	e_{0123}
Scalar $\lambda \in \mathbb{R}$	λ	0	0	0	0	0	0	0
Plane w/ normal $n \in \mathbb{R}^3$, origin shift $d \in \mathbb{R}$	0	d	n	0	0	0	0	0
Line w/ direction $n \in \mathbb{R}^3$, orthogonal shift $s \in \mathbb{R}^3$	0	0	0	s	n	0	0	0
Point $p \in \mathbb{R}^3$	0	0	0	0	0	p	1	0
Pseudoscalar $\mu \in \mathbb{R}$	0	0	0	0	0	0	0	μ
Reflection through plane w/ normal $n \in \mathbb{R}^3$, origin shift $d \in \mathbb{R}$	0	d	n	0	0	0	0	0
Translation $t \in \mathbb{R}^3$	1	0	0	$\frac{1}{2}t$	0	0	0	0
Rotation expressed as quaternion $q \in \mathbb{R}^4$	q_0	0	0	0	q_i	0	0	0
Point reflection through $p \in \mathbb{R}^3$	0	0	0	0	0	p	1	0

Figure 2: Embedding table used to convert $\mathbb{R}^3$ geometric entities into $G_{3,0,1}$ multivectors.

4 Classification architectures

4.1 1D CNN

The first classification model is a 1D CNN. It consists of a feature extraction block made up of 1D convolutional layers with batch normalization and ReLU activation, followed by a max pooling stage. There is also a fully connected block with batch normalization, ReLU activation, and dropout after each fully connected layer. Since the task is binary classification, the output activation function used is the sigmoid.

Each artery sample is represented by `data_length` tensors of size 10. Here, `data_length` refers to the number of points considered for each sample, and each tensor of size 10 is obtained by concatenating the point features: `pos`, `face`, `wss` and `pressure`. The chosen architecture and embedding allow for the extraction of information from the different types of available data. All the features, except `face` which is a set of point indexes, have been standardized using the equation (3).

The class label l_i associated to sample x_i was encoded by an index (0, 1). The whole dataset is thus represented by tuples as

$$\mathbb{D} = \{(x_i, l_i)\}_{i=1}^{N} \tag{4}$$

where N is the total number of samples in the dataset. The architecture is trained on 70% of the data, validated on 20% of data and tested on the remaining 10% of the samples.

4.2 Standard Transformer

The second model chosen as the baseline for the classification task is a standard transformer, as described in [2]. It takes the input data embedded using geometric algebra, as described in Section 3. Due to the classification nature of the task, we only used the encoder block and added some fully connected layers to produce a single value as the output.

The main blocks of the architecture are the embedding layer, the encoder, and the output layer. The input is passed to an embedding layer that produces an embedding of the input tensor and passes it to the following layer. The Encoder block is composed of a Multi-Head Attention layer, which is used to compute the attention value as specified in the paper [2], and a feedforward layer that computes the output of the encoder. The output layer is used to compute the output of the network. It is composed of two linear layers and a ReLU activation function, and is scaled using a sigmoid function in the range [0, 1].

4.2.1 Hyperparameters

The hyperparameters of the model were tuned so that they would guarantee the best trade-off between high performance on the test set and low training time. In Table 1 the accuracy of the model on the test set and training time per epoch are compared for different choices of the hidden dimension of the multi-head attention layer, highlighting that the best trade-off is achieved for $hidden\_dimension = 8$.

Hidden Dimension	**# of Parameters**	**Training Time/Epoch [s]**	**Accuracy**
8	1k	7.1	0.998
16	3k	20	0.996
32	5.7k	31	0.995

Table 1: Hyperparameter exploration for the standard transformer.

4.3 GATr

The GATr model is a transformer architecture specifically designed for the elaboration of 3-dimensional data. It takes in input multivectors encoded in geometric algebra as specified in Section 3.

The model, implemented as specified in [1], is composed by the following blocks:

1. **EquiLinear Layer:** Each block is equivariant to E(3) transformations (plane reflection, rotation, translation, and point reflection). This is ensured by the use of a consistent linear mapping:

$$\varphi(x) = \sum_{k=0}^{d+1} w_k \langle x \rangle_k + \sum_{k=0}^{d} v_k \langle x \rangle_k$$

2. **EquiLayer Norm:** Similar to classical Layer Norm but adapted to geometric algebra, computing the mean between the channels of the multivector.

3. **Geometric-Attention:** This layer is akin to standard attention [3], but it uses the following equivariant inner product:

$$\text{Attention}(q, k, v)_{i'c'} = \sum_i \text{Softmax}_i \left(\frac{\sum_c \langle q_{i'c}, k_{ic} \rangle}{\sqrt{8n_c}} \right) v_{ic'} \tag{5}$$

4. **Geometric Bilinear Layer:** It is designed to combine the geometric and join products. The inputs pass through an Equilinear Layer before the geometric and join products are computed, enhancing the network's ability to understand distances between multivectors.

5. **Gated GeLU:** Non-linearities are introduced via the activation function $GatedGELU(x) = GELU(x_1)x$, where x_1 is the scalar component of the multivector x.

6. **Output Layer:** It computes the network output using two feed-forward Linear layers and a ReLU activation, scaled by a sigmoid function in the range [0, 1].

4.3.1 GATr hyperparameter exploration

The GATr architecture underwent training using different hyperparameter configurations in order to determine the best balance between computational time and performance. In Table 2, the accuracy of the model on the test set and the training time per epoch are compared across various options for the hidden dimension of the equilinear layer (8, 16, 32) and the dropout ratio (0.1, 0.3).

The accuracy of the model decreases as the number of trainable parameters in the model increases. This suggests that the model may be overfitting the data during training, resulting in reduced predictive power on unseen data. The computational time per epoch increases with the size of the model, as expected, while the dropout ratio does not appear to have a significant impact on performance or computational time. From these observations, the final choices for the two hyperparameters are: $hidden\_dimension = 8$, $dropout = 0.1$.

Hidden Dimension	# of Parameters	Dropout	Training Time/Epoch [s]	Accuracy
8	7.9k	0.1	30.1	0.988
8	7.9k	0.3	31.4	0.978
16	30.7k	0.1	78.8	0.962
16	30.7k	0.3	80.1	0.960
32	121k	0.1	126	0.948
32	121k	0.3	125	0.942

Table 2: Hyperparameter exploration for GATr trained on 70% of the data for 10 epochs.

4.4 Loss, Optimizer and Scheduler

For all three models used in the experiments, Binary Cross Entropy Loss was employed due to the binary classification nature of the task. The Adam optimizer was selected for its efficiency and adaptive learning rate capabilities. Additionally, the Reduce LR On Plateau scheduler was implemented to dynamically adjust the learning rate based on validation performance, ensuring a more effective training process.

5 Experimental Results

The models were trained on 70% of the data, validated on 20% on the data and tested on the remaining 10% of the samples. Table 3 shows the number of parameters that characterizes each architecture as well as the performance metrics accuracy, precision, recall and f1-score evaluated on the test set. All three models achieve good results on the test set, in particular the baseline transformer achieves better performances than GATr for each of the evaluation metrics considered. Since the task is almost linearly separable, it is possible that introducing a large number of weights has led to some residual overfitting phenomena.

Model	# of Parameters	Accuracy	Precision	Recall	F1 Score
1D CNN	4.9k	0.998	0.996	1.00	0.998
Transformer	846	0.998	1.0	0.996	0.998
GATr	7.9k	0.988	0.995	0.961	0.978

Table 3: Performance metrics comparison of different architectures

Table 4 presents a comparison of the computational time for training and inference across the three architectures. The inference time refers to the time interval required to predict the class for all samples in the test set. Despite GATr exhibiting the longest training time, its inference time is comparable to that of the baseline transformer. Notably, GATr contains over nine times the number of parameters compared to the baseline

transformer. This indicates that GATr's architecture, though more complex and parameter-rich, is designed to facilitate efficient inference, potentially due to a more effective utilization of computational resources during inference.

Model	**# of Epochs**	**Training Time [s]**	**Training Time/Epoch [s]**	**Inference Time [s]**
Baseline	10	30.2	3.02	8.28
Transformer	10	49.6	4.96	27.2
GATr	10	255	25.5	30.9

Table 4: Time metrics comparison of different architectures

5.1 GATr equivariance tests

One of the main properties of GATr is the equivariance of its layers with respect to geometric operations of translation, rotation and reflection applied on an element x of the geometric algebra $G_{3,01}$ [1]. More formally, given f a mapping $G_{3,0,1} \longrightarrow G_{3,0,1}$, $x \in G_{3,0,1}$ and a representation ρ_u, then f is said to be equilinear with respect to ρ_u if

$$\rho_u(f(x)) = f(\rho_u(x)), \tag{6}$$

where ρ_u is the sandwich product that applies an operator u encoded by an element of $G_{3,0,1}$ to x. Table 5 shows the discrepancy between the two sides of equation (6) for each of the layers of GATr and four different geometric operations: plane reflection, rotation, translation and point reflection. The equation has been evaluated on a sample x of the dataset and a set of four representations ρ_u, one for each type, obtained by encoding randomly generated geometric operations in $E(3)$ in the geometric algebra $G_{3,0,1}$ using the definitions provided in Table 2. The layer that shows the highest residual values is the geometric attention layer, with the two reflection operations being the worst scoring transformations. This could be due to the design of the layer itself, although this problem may be mitigated by choosing appropriate hyperparameters. An in depth investigation on this issue could be the subject of future works.

	Equilin. Layer	**Equilayer Norm**	**Geometric Att.**	**Geom. Bilinear Layer**
Plane Reflection	3.42×10^{0}	2.09×10^{-7}	2.59×10^{1}	3.81×10^{-4}
Rotation	2.56×10^{-1}	1.13×10^{-7}	2.42×10^{0}	1.23×10^{-5}
Translation	1.91×10^{-1}	5.00×10^{-8}	1.70×10^{0}	1.46×10^{-5}
Point Reflection	5.90×10^{0}	2.44×10^{-8}	2.61×10^{1}	6.68×10^{-4}

Table 5: Equivariance test for the GATr layers

6 Conclusions

In this work, the Geometric Algebra Transformer (GATr) was introduced for classifying artery models, leveraging geometric algebra to enhance the understanding of spatial relationships and distances between multivectors. GATr was compared against a 1D CNN and a standard transformer, demonstrating comparable performance metrics despite its higher complexity. The standard transformer slightly outperformed GATr, potentially due to residual overfitting in the GATr model.

GATr maintained competitive inference times, indicating efficient parallelization. Its layers preserved equivariance to geometric transformations, which is crucial for robust geometric data manipulation.

GATr represents a significant advancement in integrating geometric algebra with deep learning, offering improved processing of geometric data. Future work could optimize GATr's equivariance properties and extend its application to other tasks.

References

[1] Johann Brehmer, Pim de Haan, Sönke Behrends, and Taco Cohen. Geometric algebra transformer, 2023.

[2] Julian Suk, Pim de Haan, Phillip Lippe, Christoph Brune, and Jelmer M. Wolterink. Mesh neural networks for se(3)-equivariant hemodynamics estimation on the artery wall, 2022.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.