# Geometric Algebra Transformers

**Claudio Schiavella,**
Sapienza, University of Rome
Rome
schiavella.1884561@studenti.uniroma1.it

**Lorenzo Cirillo**
Sapienza, University of Rome
Rome
cirillo.1895955@studenti.uniroma1.it

**Jacopo Tedeschi**
Sapienza, University of Rome
Rome
tedeschi.1882789@studenti.uniroma1.it

## 1 Introduction

Deep learning models can achieve a lot of tasks while handling various types of data, like texts, images, videos, and audio. However, modeling geometric objects while preserving their properties is an ongoing task. Indeed, geometric data are characterized by symmetries, topologies, shapes, and other structural attributes that are difficult to manage with deep learning. Therefore, we reimplement the Geometric Algebra Transformer [1], a transformer network [2] capable of modeling geometric types. Its main characteristic is the equivariance of the layers composing the network, a crucial key to preserving the geometric properties of the input data.

In particular, we focus on a simple binary classification task and implement a classical transformer architecture to make a reliable comparison. Moreover, to apply some novelty, we realize two additional versions (i.e. one for the baseline and one for the GATr) characterized by a lighter attention module consisting of a single pooling layer [3]. This attention module is extremely simple and less heavy than the classical one so the inference time is reduced during the testing phase.

Hence, we implement four models (Transformer, MetaFormer, GATr, MetaGATr) and test them on a geometric dataset and the classical evaluation metrics for a binary classification problem (i.e., accuracy, recall, precision, F1-score). Finally, as anticipated, we measure the inference times to check the speed improvements of our proposed versions. More detailed information on all topics in this report can be found in the project notebook.

## 2 Dataset

The starting dataset [4] consists of 2000 samples of single outlet arteries and 2000 samples of bifurcation arteries. Each sample has 5 properties: position, wall shear stress (wss), face, pressure, and inlet index. Because its size is very high, it was necessary to split the whole dataset into a certain number of partitions to load it on Colab. The partitions are successively unified allowing us to have a unique large file with all the samples mixed. After that, each sample is labeled based on its belonging class (i.e., 0 if "single", 1 if "bifurcating"). Finally, this obtained dataset is divided into a train set (80%) and a test set (20%).

To fully understand the significance of the data, the decision was made to have a consultation with a medical expert, Dr. Flavio Milana of Humanitas Research Hospital LinkedIn profile. Some design choices depended on the knowledge gained from the meeting. Thanks to the information acquired, among the sample characteristics we consider only the first four, excluding the input index. The latter is only an indication of the starting point of the artery, more related to the visualization of the artery 3D mesh. In addition, because of the limited resources on which the project was carried out, it was necessary to take only part of the measurements of each sample, and if we had also considered the inlet index, the number of data would have been too low.

An additional source of data information that enabled us to approach architecture design in a more informed way was the results of exploratory data analysis. A very important result coming from this phase was to see how the dataset is, excluding some outliers, linearly separable, a symptom that the binary classification task could have good results.

### 2.1 Samples preprocessing

GATr is a transformer architecture that takes as input geometric types (like points, lines, planes, ...) [1]. A dataset containing geometric samples is required to work with this architecture. Therefore, each sample has to be represented by geometric types. In the projective geometric algebra (GA) $\mathbb{G}_{3,0,1}$ [1, 5], geometric types are organized in multivectors. Geometric algebra is provided with fundamental operations that manipulate them (geometric, inner, outer product, join, blade, ...) but for reasons of brevity, we cannot go into this report.

A multivector can be thought of as an array of 16 cells, since $\mathbb{G}_{3,0,1}$ has 4 bases, hence $2^4 = 16$ which is the $GA\_dimension$. Basis combination can describe different geometric types. Multivectors are also characterized by their grade based on the number of combinations of bases involved in describing an object, as summarized in Figure 1.

| Object / operator | Scalar | Vector | | Bivector | | Trivector | | PS |
|---|---|---|---|---|---|---|---|---|
| | $1$ | $e_0$ | $e_i$ | $e_{0i}$ | $e_{ij}$ | $e_{0ij}$ | $e_{123}$ | $e_{0123}$ |
| Scalar $\lambda \in \mathbb{R}$ | $\lambda$ | 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 | $\mu$ |
| 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 1: Embeddings of common geometric objects and transformations into the projective geometric algebra $G_{3,0,1}$. The columns show different components of the multivectors with the corresponding basis elements, with $i, j \in \{1, 2, 3\}$, $j \neq i$, i.e. $ij \in \{12, 13, 23\}$.

Each property of a sample has been converted into a multivector. In particular, position as a point (trivector, grade 3), face as a plane (bivector, grade 2), wss as a translation (bivector, grade 2), pressure and inlet index as scalars (grade 0). For practical reasons, it is more convenient to map each sample to a single multivector. For this purpose, we perform a concatenation between the 5 sample features multivectors. To do that, the same $num\_items$ in each multivector is needed to concatenate them. The inlet index one has this number very small than the other ones. For this reason, and also as anticipated in Section 2, this property is completely discarded. Therefore, by concatenating the 4 (which is the number of $channels$) multivectors into a single one, we get a unique multivector representing the geometric types for each sample. Finally, we organize batches of a certain $batch\_size$ to give them as input to our architecture during the training and testing phases. Consequently, the input has size $(batch\_size, num\_items, channels, GA\_dimension)$.

## 3 Architectures

For our project, we compared four different types of architecture:

- GATr: the architecture proposed in [1]
- Transformer: a baseline architecture [2] to compare results.
- MetaGATr: proposed version of the GATr with a modified attention module [3, 6].
- MetaFormer: version of baseline with a modified attention module [3, 6].

In the following subsections, we describe the above-mentioned architectures in detail, by mainly focusing on the GATr and MetaGATr.

## 3.1 GATr

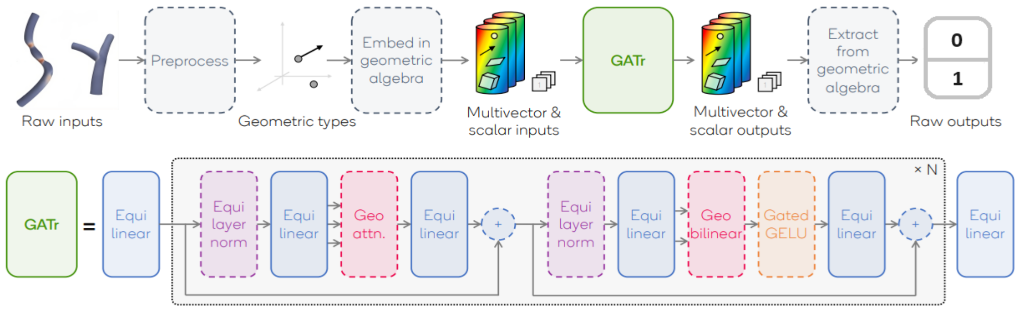

Figure 2: GATr architecture and workflow

GATr takes as input some multivectors representing the geometric properties of the data, as explained in Section 2.1. As mentioned, dealing with geometric data is not a trivial task and the involved network must have a particular property to succeed well in this: equivariance. Each layer of the GATr is equivariant to E(3) transformations, namely, rotations, translations, and reflections. Indeed, a layer $f$ is equivariant to an operator, $u$ if:

$$f(\rho_u(x)) = \rho_u(f(x)). \tag{1}$$

In Equation 1, the $\rho_u(x)$ represents the sandwich product, a particular way to apply an operator in geometric algebra.

In Figure 2, it has been reported the general workflow and the GATr architecture with all its layers, presented now step by step. The Equilinear layer is the most important GATr layer and the only one with learnable weights [1]. It processes a batch of multivectors, similarly to how a classical linear layer would do. However, some precautions are necessary to obtain a layer that is equilinear as well as linear. A first special feature is surely that each neuron in the equilinear layer is a multivector in turn. By doing so, a consistent linear mapping between input multivectors and neurons will be possible.

$$\phi(x) = \sum_{k=0}^{d+1} w_k \langle x \rangle_k + \sum_{k=0}^{d} v_k \langle x \rangle_k. \tag{2}$$

As we can see in Figure 3, the learnable weights $w_k$ map each component of the input multivector to the respective components of the neuron multivector. The learnable weights $v_k$ map each component of the input multivector without base $e_0$ to the respective components of the neuron multivector with base $e_0$.

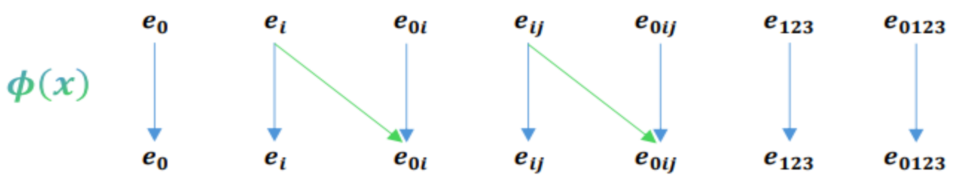

Figure 3: The conceptual idea under the equilinear mapping between multivectors

The Equilinear layer normalization (Equinorm) is very similar to its classical approach; however, due to the operations of geometric algebra, it retains an equivariant nature.

$$\text{Equinorm}(x) = \frac{x}{\sqrt{\mathbb{E}_c \langle x, x \rangle}}. \tag{3}$$

Where $\mathbb{E}_c$ is the average over the multivector channels.

The Geometric Attention takes inspiration from the one introduced in [2], but with the sum of the inner products across channels between multivectors instead of the dot-product. The chosen attention approach is a multi-head multi-query type, so each $Q, K, V$ comes from a different layer.

$$\text{Attention}(q, k, v) = \sum_i \text{Softmax}_i\left(\frac{\sum_c \langle q_i, k_i \rangle}{\sqrt{8 n_c}}\right) v_i \tag{4}$$

The Geometric Bilinear layer owes its name to the fact that it makes two operations, combining the geometric product and join. The latter is necessary because it gives the network the ability to distinguish distances.

The activation function of the GATRr is a gated GELU: a GELU function of the multivector scalar components that weights the multivector itself.

After processing the input multivector with the GATr block in inputs, the output is extracted from the geometric algebra framework, the multivector, retrieving the scalar components. This is because we have to perform a classification task and typically the input of the sigmoid function is a scalar. From that, we feed the embedded scalar, extracted from the multivector, in a linear layer, and then a sigmoid function gives the $\lambda \in [0, 1]$ needed for the predictions.

The training process of GATr, like all other architectures, is preceded by a hyperparameter search phase to identify the best choices for the network's hyperparameters.

### 3.2 MetaGATr

In this section, we describe our modification applied to the GATr architecture. It consists of replacing the attention module with a simpler token mixer, rearranging so the architecture in a MetaFormer structure [3]. The MetaFormer is an evolution of the Transformer that generalizes the attention module by replacing it with the most trivial of token mixers: a pooling layer. So the PoolFormer is implemented by relying on the following attention:

$$Attention(X) = Pooling(Norm(X)) + X. \tag{5}$$

Where the terms $X$ indicate the embedded input and $Norm(...)$ refers to the normalization technique chosen. In particular, we use the Layer Normalization technique [7], while $Pooling$ is a standard spatial pooling of a suitable size to fit the other blocks of the architecture.

Doing so greatly reduces the parameters and network inference time, with even better results than well-established models [3, 6]. Although this architecture is principally aimed at computer vision tasks[6], the authors of [3] specify how it can be extended to other types of tasks, so ours seemed an optimal field to test it.

### 3.3 Transformer and MetaFormer

Our baseline is represented by a transformer composed by:

- two MLP layer as input block used for embed data;
- one encoder constructed as the one in [2], is the only Transformer block needed due classification nature of the task;
- one MLP layer as output layer to do classification.

To maintain symmetry, the [3] approach was also applied to the baseline, resulting in a MetaFormer architecture useful for comparing MetaGATr results.

## 4 Results

In this section, we show the results of the testing phase evaluated on the four mentioned architectures. The evaluation metrics are the classical ones for a binary classification task: accuracy, precision, recall, and f1-score. In addition, we test also the inference time to check the speed improvements of the meta versions and check for the effective equivariance of the GATr layers. This evaluation phase is done on a batch of data never seen by any of the four architectures during their training.

## 4.1 Evaluation metrics

Table 1: Accuracy, precision, recall, F1-score, inference time on Transformer, MetaFormer, GATr, MetaGATr

| Model | Accuracy | Recall | Precision | F1-score | Inference Time [s] | Parameters [$10^3$] |
|---|---|---|---|---|---|---|
| Transformer | 0.99 | 0.99 | 0.99 | 0.99 | 4.85 | 3.7 |
| MetaFormer | **1.00** | 0.99 | 0.99 | 0.99 | **4.44** | 3.3 |
| GATr | **1.00** | 0.99 | 0.99 | 0.99 | 13.17 | 13.68 |
| MetaGATr | **1.00** | **1.00** | **1.00** | **1.00** | 10.54 | 5.8 |

In general, the results of each architecture were very good. This is because of a very simple task compared to the potential of the proposed networks. The closeness between the performance of the baseline Transformer and the GATr is not surprising, as even in [1], for an experiment on the same dataset as our project, they had results consistent with those obtained, with the Transformer and the GATr having good and similar performances. However, as previously introduced, the GATr allows to rely on the equivariance property. Finally, the results obtained from the MetaFormer variants are faster and sometimes even better than the Transformer baseline and GATr variants, confirming the trend of [3] also in a non-vision task.

The impact in inference time with the application of the MetaFormer approach is greater in the GATr than in the baseline Transformer because of the number of parameters. In both cases, the MetaFormer approach lowers the number of parameters more stoutly, however in the GATr, yet achieving the expected results.

## 4.2 Equivariant check

Table 2: Closness between the two equivariant equation sides

| Layer | Plane reflection | Quaternion Rotation | Translation | Point reflection |
|---|---|---|---|---|
| Equilinear | $8.61 \times 10^{-6}$ | $1.48 \times 10^{-7}$ | $2.32 \times 10^{-3}$ | $6.98 \times 10^{-1}$ |
| Equinorm | $4.85 \times 10^{-2}$ | $4.43 \times 10^{-9}$ | $7.93 \times 10^{-11}$ | $4.46 \times 10^{-9}$ |
| Geometric Attention | $6.26 \times 10^{-4}$ | $1.00 \times 10^{-4}$ | $2.97 \times 10^{-1}$ | $6.01 \times 10^{2}$ |
| Geometric Bilinear | $1.06 \times 10^{-7}$ | $1.60 \times 10^{-9}$ | $2.98 \times 10^{-5}$ | $2.20 \times 10^{-2}$ |
| Gated GELU | $7.05 \times 10^{-7}$ | $1.57 \times 10^{-8}$ | $2.48 \times 10^{-12}$ | $1.7 \times 10^{-1}$ |

The particularity of the GATr neural network of the project is its ability to be equivariant to E(3), the rigid transformations of 3D space: rotations, translations, and reflections. This particularity is an important element to achieve, as it differs from other Transformer-type architectures and will be able to be an important term of comparison.

The equivariant check described in Equation 1 will be applied for each GATr layer and each operator in Figure 1. However, there is to specify how due to some imperfections, such as numerical approximations and too deep details in the geometric algebra, it is very unlikely that the two sides of the Equation 1, passed in complex layers, and relatively imperfect as mentioned above, will be equal. Perhaps they may be very similar, with very little difference in terms of elements, but the latter would give a negative result if we applied a classical control such as *torch.equal*. Hence the design choice to analyze the distance between the two sides of the equation presented above, considering ourselves satisfied if the distance between these two values is very low.

In this context, the results shown in Table 2 are very good, indicating that the GATr, most of the time, maintains the equivariance property. Geometric Attention is the layer that performs worst. However, this could be associated with the fact that attention from [1] was implemented without distance awareness. Also in the same paper, this additional variant was implemented, which could be analyzed as future work.

## References

[1] Johann Brehmer, Pim De Haan, Sönke Behrends, and Taco Cohen. Geometric algebra transformers. *arXiv preprint arXiv:2305.18415*, 2023.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[3] Yu Weihao, Luo Mi, Zhou Pan, Si Chenyang, Zhou Yichen, Wang Xinchao, Feng Jiashi, and Yan Shuicheng. Metaformer is actually what you need for vision. *arXiv preprint arXiv:2111.11418*, 2022.

[4] 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.

[5] L. Dorst and S. Mann. Geometric algebra: A computational framework for geometrical applications part 1. *IEEE Computer Graphics and Applications*, 22(3):24–31, 2002.

[6] Claudio Schiavella, Lorenzo Cirillo, Lorenzo Papa, Paolo Russo, and Irene Amerini. Optimize vision transformer architecture via efficient attention modules: A study on the monocular depth estimation task. In Gian Luca Foresti, Andrea Fusiello, and Edwin Hancock, editors, *Image Analysis and Processing - ICIAP 2023 Workshops*, pages 383–394, Cham, 2024. Springer Nature Switzerland.

[7] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.