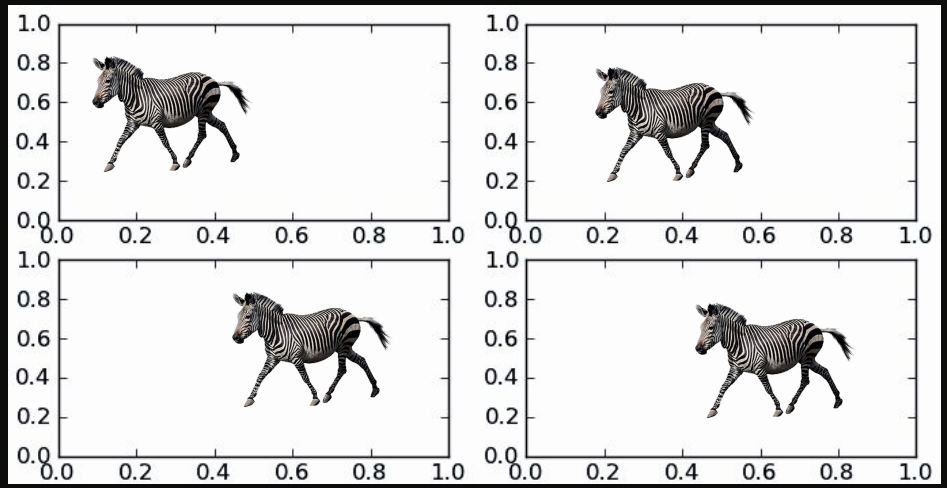
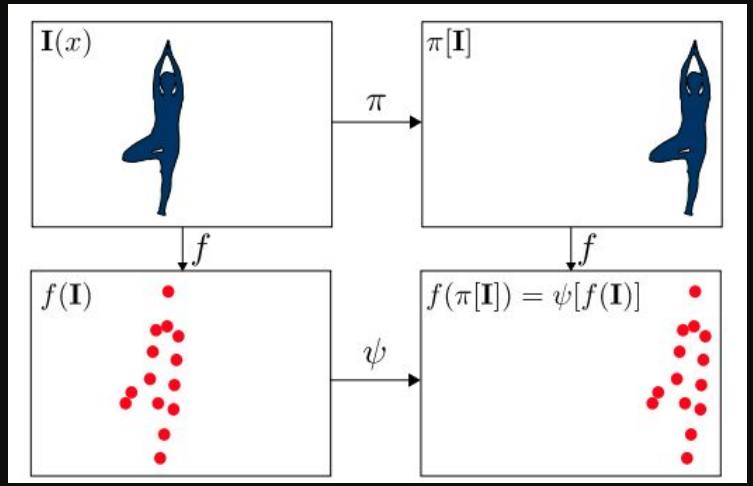
Geometric algebra transformers

Recap - equivariance

- Equivariance





…a feature mapping f: X → Y is equivariant to a group of transformations if we can associate every transformation π∈Π of the input x∈X with a transformation ψ∈Ψ of the features; that is,

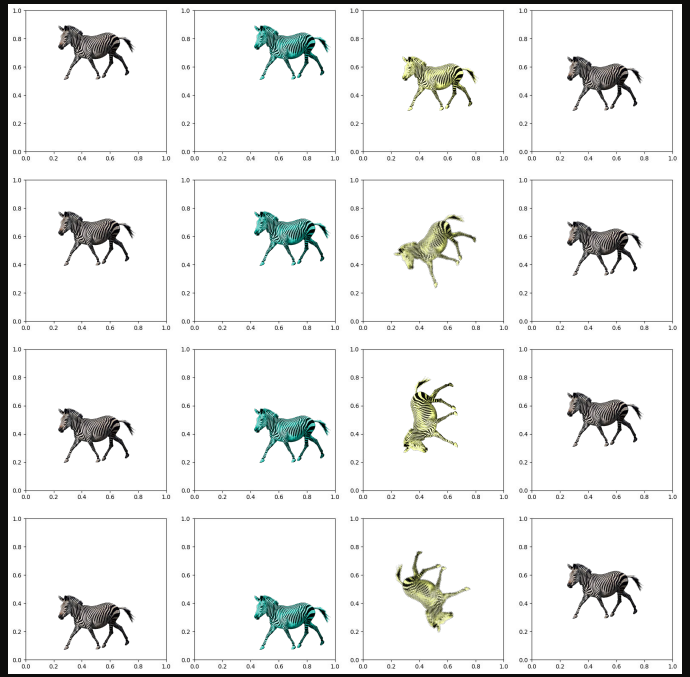
$$\psi[f(\mathbf{x})]=f(\pi[\mathbf{x}])$$

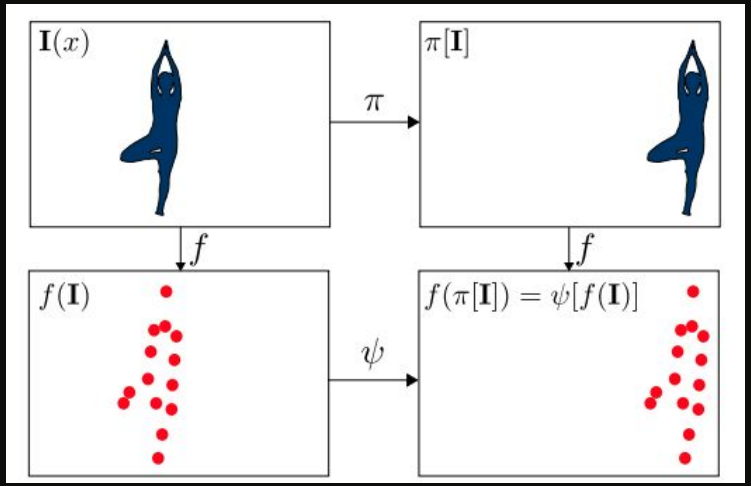
we have LaTeX at home

LaTeX at home:

Recap - equivariance

- Equivariance





…a feature mapping f: X → Y is equivariant to a group of transformations if we can associate every transformation π∈Π of the input x∈X with a transformation ψ∈Ψ of the features; that is,

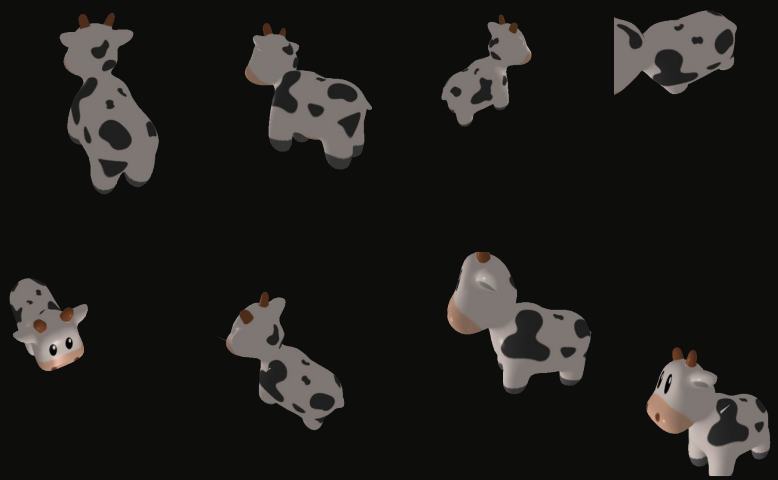
$$\psi[f(\mathbf{x})]=f(\pi[\mathbf{x}])$$

we have LaTeX at home

LaTeX at home:

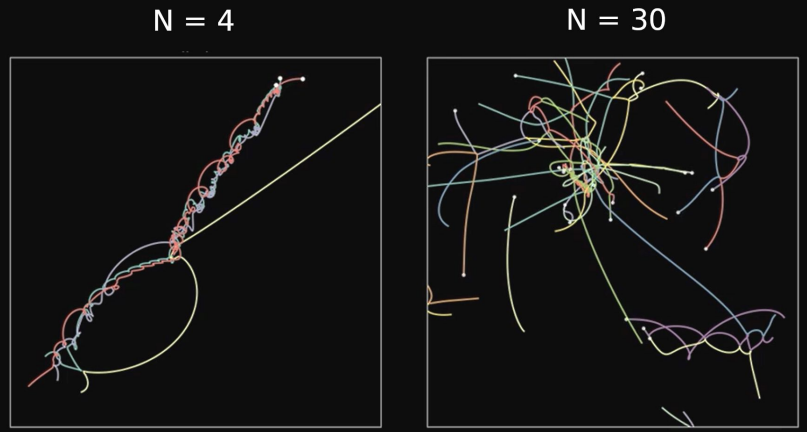
Recap - equivariance

- Equivariance
- E(3) symmetry
  - SE(3)



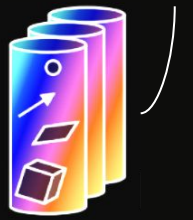
Recap - equivariance

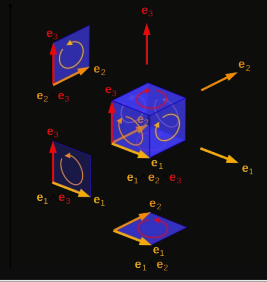
- Equivariance
- E(3) symmetry
  - SE(3)
- Example task: n-body



Big-picture view of GATr

- Equivariant module
- Acts on *3D multivectors*: “arrays” representing points/lines/planes/volumes





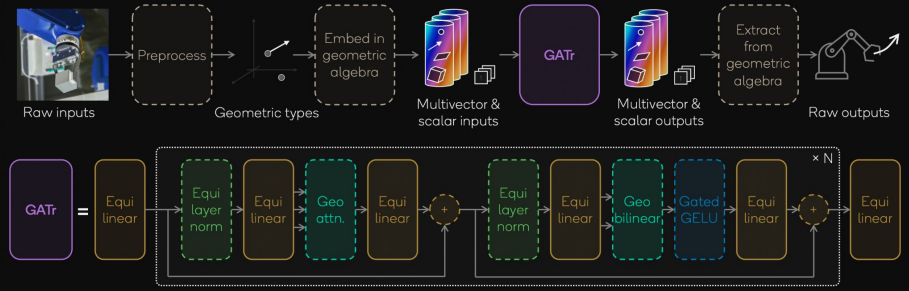


Figure 1: Overview over the GATr architecture. Boxes with solid lines are learnable components, those with dashed lines are fixed.

Object / operator	Scalar 1	Vector e_0	Vector e_i	Bivector e_{0i}	Bivector e_{ij}	Trivector e_{0ij}	Trivector e_{123}	PS 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

Table 1: Embeddings of common geometric objects and transformations into the projective geometric algebra $\mathbb{G}_{3,0,1}$. 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. $ij \in \{12, 13, 23\}$. For simplicity, we fix gauge ambiguities (the weight of the multivectors) and leave out signs (which depend on the ordering of indices in the basis elements).

Big-picture view of GATr

- Equivariant module
- Acts on *3D multivectors*: “arrays” representing points/lines/planes/volumes
- Linear (bilinears), attention, normalization, scalar nonlinearity

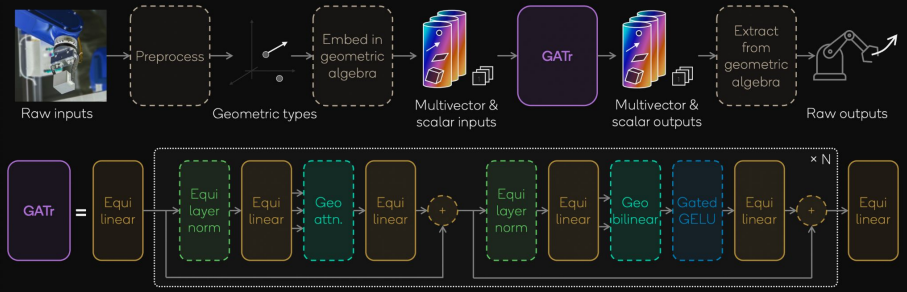


Figure 1: Overview over the GATr architecture. Boxes with solid lines are learnable components, those with dashed lines are fixed.

Typical input for GATr

Big-picture view of GATr

- Equivariant module
- Acts on *3D multivectors*: “arrays” representing points/lines/planes/volumes
- Linear (bilinears), attention, normalization, scalar nonlinearity
- Additional scalar vectors for the main sequence

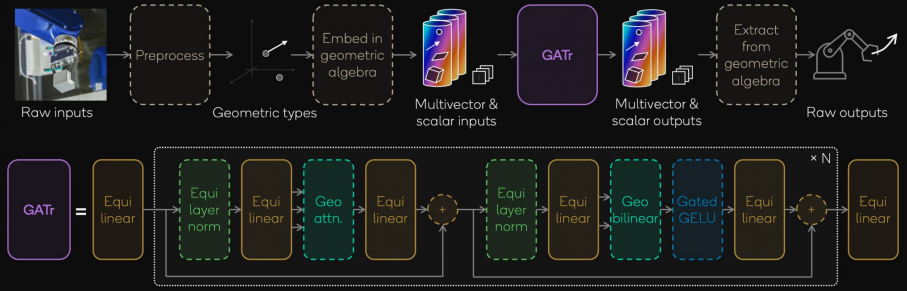


Figure 1: Overview over the GATr architecture. Boxes with solid lines are learnable components, those with dashed lines are fixed.

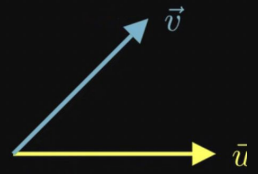
Typical input for GATr

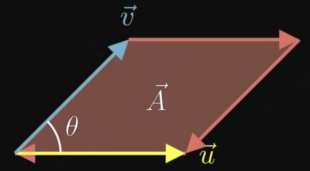
Geometric Algebra

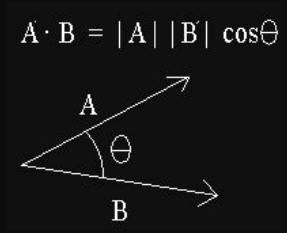
- Basis vectors e_1, e_2, e_3
- Products
  - Square must equal squared norm $vv = \langle v, v \rangle$
  - Antisymmetry $e_i e_j = -e_j e_i \;\; e_i e_j + e_j e_i = (e_i + e_j)^2 - e_i^2 - e_j^2 = 0$
- Multivectors $\sum_{i=0}^{d} \binom{d}{k} = 2^d$
- Operations $x = x_s + x_1 e_1 + x_2 e_2 + x_3 e_3 + x_{12} e_1 e_2 + x_{13} e_1 e_3 + x_{23} e_2 e_3 + x_{123} e_1 e_2 e_3$
  - Geometric product
  - Inner/interior product $\langle x, y \rangle = (xy + yx)/2$
  - Exterior/wedge/outer product $x \wedge y \equiv (xy - yx)/2$
  - Dual $x \mapsto x^* \;\; e_1 \mapsto e_{23}$
  - Sandwich product For odd u, $u[x] = u\hat{x}u^{-1}$, while for even u, $u[x] = uxu^{-1}$
  - Left contraction $a \rfloor b := \langle ab \rangle_{l-k}$
  - “Join” $x, y \mapsto (x^* \wedge y^*)^*$
- Blades $x = x_1 \wedge x_2 \wedge \ldots \wedge x_k$

2D Geometric Algebra

- Vectors
- Inner product
- Bivectors and wedge product
- Geometric product
- Reflection - uvu
- Rotation as reflection

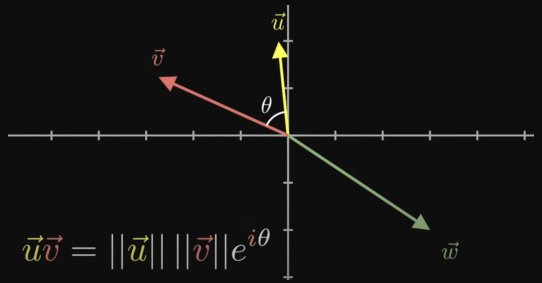


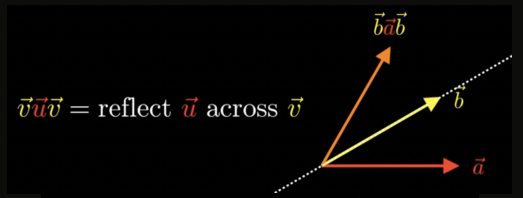


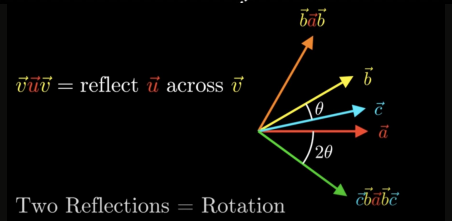


$$\vec{v} \wedge \vec{u} = \vec{A}$$

$$||\vec{u} \wedge \vec{v}|| = ||\vec{u}||\,||\vec{v}|| \sin(\theta)$$



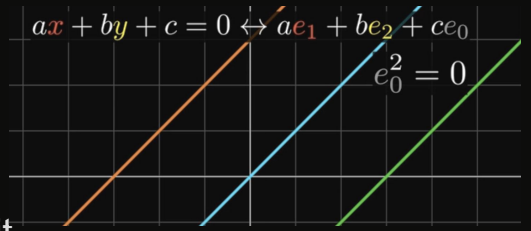




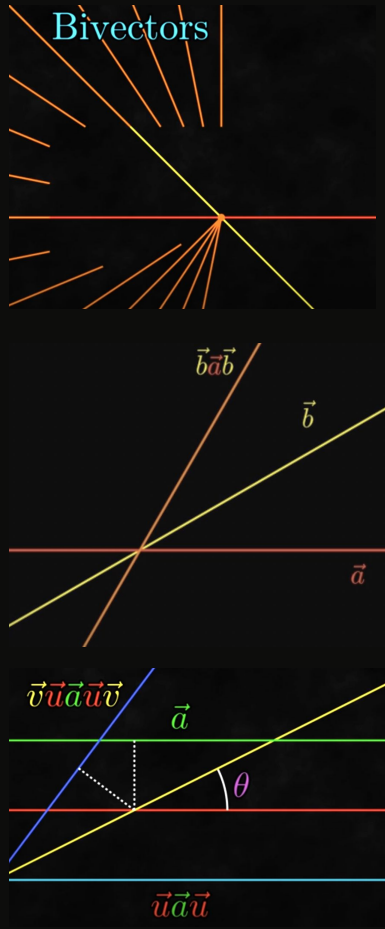
[4], [5]

Projective 2D Geometric Algebra

- Vectors as lines
- 0^{th} basis element
  - Squares to 0
  - Destroys bivectors containing it
- Points/bivectors as intersections
- Reflections as sandwiches
- Rotations as reflections



$$\vec{a} = a_1 e_1 + a_2 e_2 + a_0 e_0$$
$$\vec{b} = b_1 e_1 + b_2 e_2 + b_0 e_0$$
Intersection point: $\left(\frac{a_2 b_0 - a_0 b_2}{a_1 b_2 - a_2 b_1}, \frac{a_0 b_1 - a_1 b_0}{a_1 b_2 - a_2 b_1}\right)$
$$e_{12} + \frac{a_2 b_0 - a_0 b_2}{a_1 b_2 - a_2 b_1} e_{20} + \frac{a_0 b_1 - a_1 b_0}{a_1 b_2 - a_2 b_1} e_{01}$$



[5]

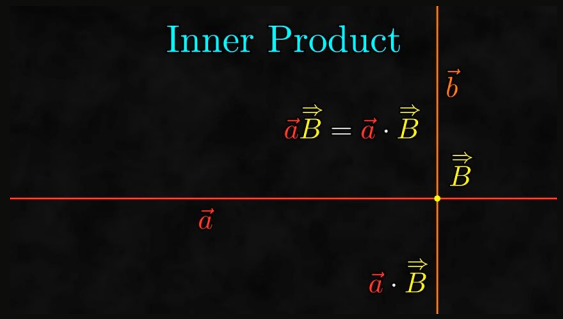
Projective 2D Geometric Algebra

- Vectors as lines
- 0th basis element e_0
  - Squares to 0
  - Destroys bivectors containing it
- Points/bivectors as intersections
- Reflections as sandwiches
- Rotations as reflections
- Join
- Inner product
  - = geometric product for point on line

$(x_1, y_1) \quad (x_2, y_2)$

$(y_1 - y_2)x + (x_2 - x_1)y + x_1 y_2 - y_1 x_2 = 0$

Join of $\vec{\vec{A}}$ and $\vec{\vec{B}}$: $\vec{\vec{A}} \vee \vec{\vec{B}}$



Projective 3D Geometric Algebra

- Vectors correspond to planes
- Bivectors are lines
- Trivectors are points
- Wedge = “meet”
  - = geometric product for orthogonal planes

$$|ae_1 + be_2 + ce_3 + de_0| = \sqrt{a^2 + b^2 + c^2}$$

$e_i e_j = -e_j e_i \quad (i \neq j)$

$e_1^2 = 1$

$e_2^2 = 1$

$e_3^2 = 1$

$e_0^2 = 0$

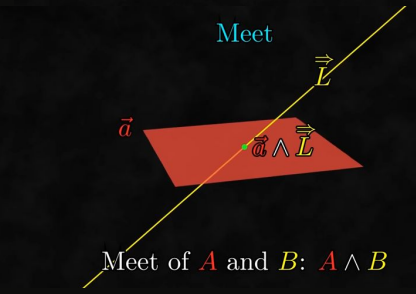
1

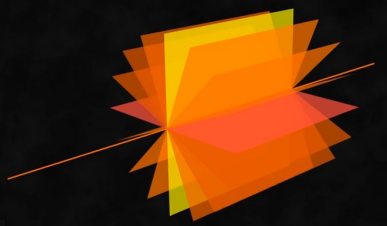
$e_1 \quad e_2 \quad e_3 \quad e_0$

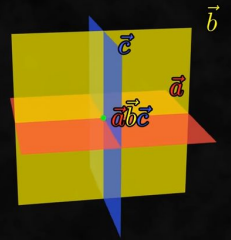
$e_{01} \quad e_{02} \quad e_{03} \quad e_{12} \quad e_{13} \quad e_{23}$

$e_{012} \quad e_{013} \quad e_{023} \quad e_{123}$

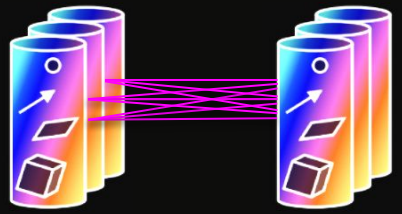
e_{0123}

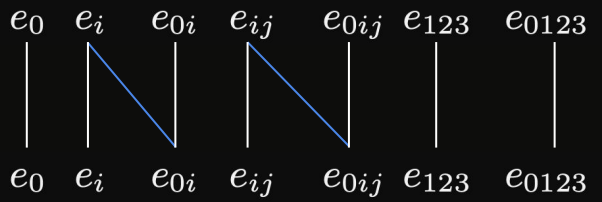




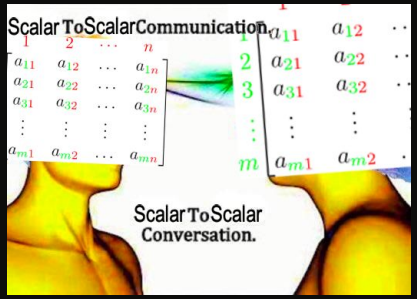


Equivariant linear layer





- Blade projection
- For each multivector-multivector connection
  - Sum of
    - weighted sum of blade projections
    - weighted sum of blade projections multiplied by e_0
- Scalars are processed normally (and with bias)



Proposition 1. *Any linear map* $\phi : \mathbb{G}_{d,0,1} \to \mathbb{G}_{d,0,1}$ *that is equivariant to* $\mathrm{Pin}(d,0,1)$ *is of the form*

$$\phi(x) = \sum_{k=0}^{d+1} w_k \langle x \rangle_k + \sum_{k=0}^{d} v_k e_0 \langle x \rangle_k \tag{4}$$

for parameters $w \in \mathbb{R}^{d+2}, v \in \mathbb{R}^{d+1}$. *Here* $\langle x \rangle_k$ *is the blade projection of a multivector, which sets all non-grade-k elements to zero.*

Equivariant linear layer

Lemma 2. *In the Euclidean ($r = 0$) or projective ($r = 1$) geometric algebra $\mathbb{G}_{n,0,r}$, let x be a k-blade. Let u be a 1-versor. Then* $u[x] = x \iff u \rfloor x = 0$ *and* $u[x] = -x \iff u \wedge x = 0$.

- Decompose versor into a part tangential to x and a part normal to x

In the projective algebra, a blade x is defined to be *ideal* if it can be written as $x = e_0 \wedge y$ for another blade y.

Proof. Let x be a k-blade and u a vector of unit norm. We can decompose u into $u = t + v$ with $t \wedge x = 0$ (the part tangential to the subspace of x) and $v \rfloor x = 0$ (the normal part). This decomposition is unique unless x is ideal in the projective GA, in which case the e_0 component of u is both normal and tangential, and we choose t Euclidean.

Let $l \geq k$. Given a k-vector a and l-vector b, define the *left contraction* as $a \rfloor b := \langle ab \rangle_{l-k}$, which is a $l - k$-vector. For $k = 1$, and b a blade $b = b_1 \wedge \ldots \wedge b_l$. Geometrically, $a \rfloor b$ is the projection of a to the space spanned by the vectors b_i. Thus we have that $a \rfloor b = 0 \iff \forall i, \langle a, b_i \rangle = 0$ [18, Sec 3.2.3], in which case we define a and b to be *orthogonal*. In particular, two vectors a, b are orthogonal if their inner product is zero. Futhermore, we define a vector a to be *tangential* to blade b if $a \wedge b = 0$.

Equivariant linear layer

Lemma 2. *In the Euclidean ($r = 0$) or projective ($r = 1$) geometric algebra $\mathbb{G}_{n,0,r}$, let x be a k-blade. Let u be a 1-versor. Then $u[x] = x \iff u \rfloor x = 0$ and $u[x] = -x \iff u \wedge x = 0$.*

- Decompose versor into a part tangential to x and a part normal to x
- Expand sandwich using properties

In the projective algebra, a blade x is defined to be *ideal* if it can be written as $x = e_0 \wedge y$ for another blade y.

Proof. Let x be a k-blade and u a vector of unit norm. We can decompose u into $u = t + v$ with $t \wedge x = 0$ (the part tangential to the subspace of x) and $v \rfloor x = 0$ (the normal part). This decomposition is unique unless x is ideal in the projective GA, in which case the e_0 component of u is both normal and tangential, and we choose t Euclidean.

In either case, note the following equalities: $xt = (-1)^{k-1}tx; xv = (-1)^k vx; vt = -tv$ and note $\nexists \lambda \neq 0$ such that $vtx = \lambda x$, which can be shown e.g. by picking a basis. Then:

$$u[x] = (-1)^k(t+v)x(t+v) = (t+v)(-t+v)x = (-\|t\|^2 + \|v\|^2)x - 2vtx$$

We have $u[x] \propto x \iff vtx = 0$. If x is not ideal, this implies that either $v = 0$ (thus $u \wedge x = 0$ and $u[x] = -x$) or $t = 0$ (thus $u \rfloor x = 0$ and $u[x] = x$). If x is ideal, this implies that either $v \propto e_0$ (thus $u \wedge x = 0$ and $u[x] = -x$) or $t = 0$ (thus $u \rfloor x = 0$ and $u[x] = x$). □

Equivariant linear layer

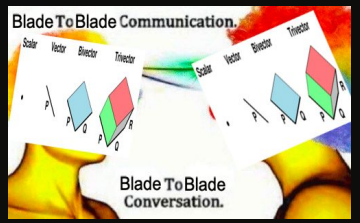
Lemma 3. *Let $r \in \{0, 1\}$. Any linear $\mathrm{Pin}(n, 0, r)$-equivariant map $\phi : \mathbb{G}_{n,0,r} \to \mathbb{G}_{n,0,r}$ can be decomposed into a sum of equivariant maps $\phi = \sum_{lkm} \phi_{lkm}$, with ϕ_{lkm} equivariantly mapping k-blades to l-blades. If $r = 0$ (Euclidean algebra) or $k < n + 1$, such a map ϕ_{lkm} is defined by the image of any one non-ideal k-blade, like $e_{12\ldots k}$. Instead, if $r = 1$ (projective algebra) and $k = n + 1$, then such a map is defined by the image of a pseudoscalar, like $e_{01\ldots n}$.*

- Decompose linear map into per-image- and max-kernel- grade maps
- Decompose y as a sum of blades which are images of x blades
  - Lemmas 1/4 show that unit versors act transitively on blades with positive norm
- Still need to define the linear mapping between blades

Proof. The $\mathrm{Pin}(n, 0, r)$ group action maps k-vectors to k-vectors. Therefore, ϕ can be decomposed into equivariant maps from grade k to grade l: $\phi(x) = \sum_{lk} \phi_{lk}(\langle x \rangle_k)$, with ϕ_{lk} having l-vectors as image, and all k'-vectors in the kernel, for $k' \neq k$. Let x be an non-ideal k-blade (or pseudoscalar if $k = n + 1$). By lemmas 1 and 4, in both Euclidean and projective GA, the span of the k-vectors in the orbit of x contains any k-vector. So ϕ_{lk} is defined by the l-vector $y = \phi_{lk}(x)$. Any l-vector can be decomposed as a finite sum of l-blades: $y = y_1 + \ldots y_M$. We can define $\phi_{lkm}(x) = y_m$, extended to all l-vectors by equivariance, and note that $\phi_{lk} = \sum_m \phi_{lkm}$. □

Equivariant linear layer

Proposition 5. *For the projective geometric algebra $\mathbb{G}_{n,0,1}$, any linear endomorphism $\phi : \mathbb{G}_{n,0,1} \to \mathbb{G}_{n,0,1}$ that is equivariant to the group $\mathrm{Pin}(n,0,r)$ (equivalently to $E(n)$) is of the type $\phi(x) = \sum_{k=0}^{n+1} w_k \langle x \rangle_k + \sum_{k=0}^{n} v_k e_0 \langle x \rangle_k$, for parameters $w \in \mathbb{R}^{n+2}, v \in \mathbb{R}^{n+1}$.*



- Lemma 3, blade-to-blade mappings
- Lemma 2, wedge of basis vectors ≤ k with both x and ϕ(x) is equal to 0
  - ϕ is a wedge with an l-k blade of vectors orthogonal to x
    - wedges preserve orthogonality
- Lemma 2, left contraction of basis vectors > k with both x and ϕ(x) is equal to 0
  - y vectors are orthogonal to all basis vectors
  - blade is scalar or proportional to e_0
- Pseudoscalars have only the scalar blade left

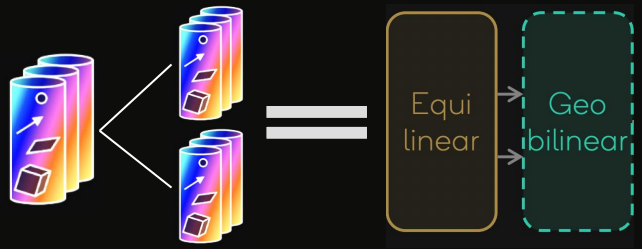
Proof. Following Lemma 3, decompose ϕ into a linear equivariant map from k-blades to l-blades. For $k < n+1$, let $x = e_{12\ldots k}$. Then following Lemma 2, for any $1 \leq i \leq k$, $e_i \wedge x = 0, e_i[x] = -x$, and $e_i[\phi(x)] = \phi(e_i[x]) = \phi(-x) = -\phi(x)$ and thus $e_i \wedge \phi(x) = 0$. Therefore, we can write $\phi(x) = x \wedge y_1 \wedge \ldots \wedge y_{l-k}$, for $l-k$ vectors y_j orthogonal to x.

Also, again using Lemma 2, for $k < i \leq n$, $e_i \rfloor x = 0 \implies e_i[\phi(x)] = \phi(x) \implies e_i \rfloor \phi(x) = 0 \implies \forall i, \langle e_i, y_j \rangle = 0$. Thus, y_j is orthogonal to all e_i with $1 \leq i \leq n$. Hence, $l = k$ or $l = k+1$ and $y_1 \propto e_0$.

For $k = n+1$, let $x = e_{012\ldots k}$. By a similar argument, all invertible vectors u tangent to x must be tangent to $\phi(x)$, thus we find that $\phi(x) = x \wedge y$ for some blade y. For any non-zero $\phi(x)$, $y \propto 1$, and thus $\phi(x) \propto x$. By Lemma 3, by equivariance and linearity, this fully defines ϕ. □

Bilinear layer

- This is the layer that does geometric operations on multivectors
  - Geometric product $x \wedge y \equiv (xy - yx)/2$
  - We can represent inner and wedge products $\langle x, y \rangle = (xy + yx)/2$
- But it can't distinguish distances!

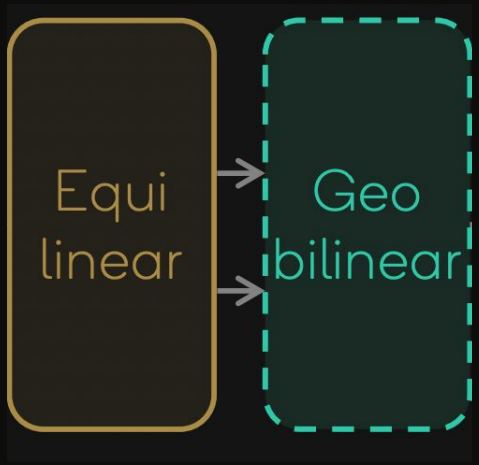


Lemma 7. *For the algebra $\mathbb{G}_{n,0,r}$, for multivectors x, y, we have $\|xy\| = \|x\| \, \|y\|$.*

Proof. $\|xy\|^2 = xy\widetilde{xy} = xy\tilde{y}\tilde{x} = x\|y\|^2\tilde{x} = x\tilde{x}\|y\|^2 = \|x\|^2\|y\|^2$

Bilinear layer

- This is the layer that does geometric operations on multivectors
  - Geometric product
  - We can represent inner and wedge products
- But it can't distinguish distances!
- Need to add joins
- z is the mean of all inputs *to the network* (._.)
  - Ugly hack to make multivectors mirror under mirrorings



$$x, y, z \mapsto \mathrm{EquiJoin}(x, y; z) = z_{0123}(x^* \wedge y^*)^*$$

Bilinear layer

For the geometric product, equivariance is straightforward: Any transformation $u \in \mathrm{Pin}(n, 0, r)$, gives a homomorphism of the geometric algebra, as for any multivectors x, y, $u[xy] = u\widehat{xy}u^{-1} = u\widehat{x}\widehat{y}u^{-1} = u\widehat{x}u^{-1}u\widehat{y}u^{-1} = u[x]u[y]$. The geometric product is thus equivariant.

- 😀 What about the join?

[12]The authors agree with the reader that there must be an easier way to prove this.

Bilinear layer

The role of the dual is to have a bijection $\cdot^* : \mathbb{G}_{n,0,0} \to \mathbb{G}_{n,0,0}$ that maps k-vectors to $(n-k)$-vectors. For the Euclidean algebra, with a choice of pseudoscalar $\mathcal{I}$, we can define a dual as:

$$x^* = x\mathcal{I}^{-1} = x\tilde{\mathcal{I}} \tag{6}$$

Lemma 5. *In Euclidean algebra $\mathbb{G}_{n,0,0}$, the join is $\mathrm{Spin}(n,0,0)$ equivariant. Furthermore, it is $\mathrm{Pin}(n,0,0)$ equivariant if and only if n is even.*

Proof. The join is equivariant to the transformations from the group $\mathrm{Spin}(n,0,0)$, which consists of the product of an even amount of unit vectors, because such transformations leave the pseudoscalar $\mathcal{I}$ invariant, and the operation consists otherwise of equivariant geometric and wedge products.

Bilinear layer

Proposition 7. *In the algebra* $\mathbb{G}_{n,0,1}$*, the join* $a \vee b = (a^* \wedge b^*)^{-*}$ *is equivariant to* $\mathrm{Spin}(n, 0, 1)$.

Proof. Even though the dual is not a $\mathbb{G}_{n,0,1}$ operation, we can express the join in the algebra as follows. We decompose a k-vector x as $x = t_x + e_0 p_x$ into a Euclidean k-vector t_x and a Euclidean $(k-1)$-vector p_x. Then Dorst [17, Eq (35)] computes the following expression

$$\begin{aligned}(t_x + e_0 p_x) \vee (t_y + e_0 p_y) &= ((t_x + e_0 p_x)^* \wedge (t_y + e_0 p_y)^*)^{-*} \\ &= t_x \vee_{\mathrm{Euc}} p_y + (-1)^n \widehat{p_x} \vee_{\mathrm{Euc}} t_y + e_0 (p_x \vee_{\mathrm{Euc}} p_y)\,, \end{aligned} \tag{10}$$

where the Euclidean join of vectors a, b in the projective algebra is defined to equal the join of the corresponding vectors in the Euclidean algebra:

$$a \vee_{\mathrm{Euc}} b := ((a \widetilde{e_{12\ldots n}}) \wedge (b \widetilde{e_{12\ldots n}}))e_{12\ldots n}$$

Bilinear layer

$$
\begin{aligned}
\tau[x] \vee \tau[y] &= (\tau[t_x] + e_0 p_x) \vee (\tau[t_y] + e_0 p_y) \\
&= (t_x + e_0(p_x - v \rfloor t)) \vee (t_y + e_0(p_y - v \rfloor t_y)) \quad \text{typo} \\
&= x \vee y - t_x \vee_{\text{Euc}} (v \rfloor t_y) - (-1)^n \widehat{v \rfloor t_x} \vee_{\text{Euc}} t_y \\
&\quad - e_0 \left(p_x \vee_{\text{Euc}} (v \rfloor t_y) + (v \rfloor t_x) \vee_{\text{Euc}} p_y\right) && \text{Used (10) \& linearity} \\
&= x \vee y - e_0 \left(p_x \vee_{\text{Euc}} (v \rfloor t_y) + (v \rfloor t_x) \vee_{\text{Euc}} p_y\right) && \text{Used (9)} \\
&= x \vee y - e_0 \left(-(-1)^n \widehat{v \rfloor p_x} \vee_{\text{Euc}} t_y + (v \rfloor t_x) \vee_{\text{Euc}} p_y\right) && \text{Used (9)} \\
&= x \vee y - e_0 \left((-1)^n (v \rfloor \widehat{p_x}) \vee_{\text{Euc}} t_y + (v \rfloor t_x) \vee_{\text{Euc}} p_y\right) \\
&= x \vee y - e_0 \left(v \rfloor \left\{(-1)^n \widehat{p_x} \vee_{\text{Euc}} t_y + t_x \vee_{\text{Euc}} p_y\right\}\right) && \text{Used (8)} \\
&= \tau[x \vee y]
\end{aligned}
$$

Normalization and activation

- GELU is only over the scalar component
- Layernorm normalizes by the RMS of the multivectors across channel dimension

$\mathrm{GatedGELU}(x) = \mathrm{GELU}(x_1)x$, where x_1 is the scalar component of the multivector x.

$$\mathrm{LayerNorm}(x) = x / \sqrt{\mathbb{E}_c \langle x, x \rangle}$$

Attention

- Instead of dot product, use sum of inner products between multivectors
  - Inner product is invariant to E(3)
- Take the 8-dimensionality into account
- Add attention weights from linear and scalar branch

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

or

$$\text{Attention}(q,k,v)_{i'c'} = \sum_i \text{Softmax}_i\left(\frac{\sum_c \langle q^{MV}_{i'c}, k^{MV}_{ic}\rangle + \sum_c q^s_{i'c} k^s_{ic}}{\sqrt{8n_{MV} + n_s}}\right) v_{ic'}$$

Attention

- Inner product doesn’t take e_0-containing elements into account
- Solution: add nonlinear features

$$\langle q_{i'c}, k_{ic} \rangle \to \langle q_{i'c}, k_{ic} \rangle + \phi(q_{i'c}) \cdot \psi(k_{ic})$$

Attention

- Inner product doesn't take e_0-containing elements into account
- Solution: add nonlinear features
- …

$$\phi(q) = \omega(q_{\backslash 0}) \begin{pmatrix} q_{\backslash 0}^2 \\ \sum_i q_{\backslash i}^2 \\ q_{\backslash 0}\, q_{\backslash 1} \\ q_{\backslash 0}\, q_{\backslash 2} \\ q_{\backslash 0}\, q_{\backslash 3} \end{pmatrix} \quad \text{and} \quad \psi(k) = \omega(k_{\backslash 0}) \begin{pmatrix} -\sum_i k_{\backslash i}^2 \\ -k_{\backslash 0}^2 \\ 2k_{\backslash 0}\, k_{\backslash 1} \\ 2k_{\backslash 0}\, k_{\backslash 2} \\ 2k_{\backslash 0}\, k_{\backslash 3} \end{pmatrix} \quad \text{with} \quad \omega(x) = \frac{x}{x^2 + \epsilon}\,.$$

Attention

- Inner product doesn't take e_0-containing elements into account
- Solution: add nonlinear features
- …Cool

Our choice of these nonlinear features not only maintains equivariance, but has a straightforward geometric interpretation. When the trivector components of queries and keys represent 3D points (see Tbl. 1), $\psi(q) \cdot \phi(k)$ is proportional to the pairwise negative squared Euclidean distance. GATr's attention mechanism is therefore directly sensitive to Euclidean distance, while still respecting the highly efficient dot-product attention format.

$$\phi(q) \cdot \psi(k) = -\omega(q_{\backslash 0})\omega(k_{\backslash 0})\|k_{\backslash 0}\,\vec{q} - q_{\backslash 0}\,\vec{k}\|^2_{\mathbb{R}^3}$$

$$\vec{x} = (x_{\backslash 1}, x_{\backslash 2}, x_{\backslash 3})^T$$

Attention

- Inner product doesn't take e_0-containing elements into account
- Solution: add nonlinear features
- …Cool - wait, this is just the square of a difference

$$\phi(q) = \omega(q_{\setminus 0}) \begin{pmatrix} q_{\setminus 0}^2 \\ \sum_i q_{\setminus i}^2 \\ q_{\setminus 0}\, q_{\setminus 1} \\ q_{\setminus 0}\, q_{\setminus 2} \\ q_{\setminus 0}\, q_{\setminus 3} \end{pmatrix} \quad \text{and} \quad \psi(k) = \omega(k_{\setminus 0}) \begin{pmatrix} -\sum_i k_{\setminus i}^2 \\ -k_{\setminus 0}^2 \\ 2k_{\setminus 0}\, k_{\setminus 1} \\ 2k_{\setminus 0}\, k_{\setminus 2} \\ 2k_{\setminus 0}\, k_{\setminus 3} \end{pmatrix} \quad \text{with} \quad \omega(x) = \frac{x}{x^2 + \epsilon}\,.$$

$$\phi(q) \cdot \psi(k) = -\omega(q_{\setminus 0})\omega(k_{\setminus 0}) \left\| k_{\setminus 0}\, \vec{q} - q_{\setminus 0}\, \vec{k} \right\|_{\mathbb{R}^3}^2$$

Here the index $\setminus i$ denotes the trivector component with all indices *but* i. $\vec{x} = (x_{\setminus 1}, x_{\setminus 2}, x_{\setminus 3})^T$

Attention

- Inner product doesn't take e_0-containing elements into account
- Solution: add nonlinear features
- …Cool - wait, this is just the square of a difference
- Transformation identity; rotation preserves norm; omega takes untransformed q/k

$$(q_{\backslash 0}, \vec{q}) \mapsto (q_{\backslash 0}, R\vec{q} + q_{\backslash 0}\vec{t})$$

$$\begin{aligned}
\phi(q) \cdot \psi(k) &\mapsto -\omega(q_{\backslash 0})\omega(k_{\backslash 0}) \| k_{\backslash 0}\,(R\vec{q} + q_{\backslash 0}\vec{t}) - q_{\backslash 0}\,(R\vec{k} + k_{\backslash 0}\vec{t}) \|^2_{\mathbb{R}^3} \\
&= -\omega(q_{\backslash 0})\omega(k_{\backslash 0}) \| R(k_{\backslash 0}\,\vec{q} - q_{\backslash 0}\,\vec{k}) + q_{\backslash 0}k_{\backslash 0}\vec{t} - q_{\backslash 0}k_{\backslash 0}\vec{t} \|^2_{\mathbb{R}^3} \\
&= -\omega(q_{\backslash 0})\omega(k_{\backslash 0}) \| k_{\backslash 0}\,\vec{q} - q_{\backslash 0}\,\vec{k} \|^2_{\mathbb{R}^3} \\
&= \phi(q) \cdot \psi(k)\,.
\end{aligned}$$

Attention

- Inner product doesn't take e_0-containing elements into account
- Solution: add nonlinear features
- …Cool - wait, this is just the square of a difference
- Transformation identity; rotation preserves norm; omega takes untransformed q/k
- Related to conformal geometric algebra
  - PGA is a subalgebra of CGA
  - CGA contains an infinity element

Attention

- Inner product doesn't take e_0-containing elements into account
- Solution: add nonlinear features
- …Cool - wait, this is just the square of a difference
- Transformation identity; rotation preserves norm; omega takes untransformed q/k
- Related to conformal geometric algebra
  - PGA is a subalgebra of CGA
  - CGA contains an infinity element
- Stability tricks
  - Omega could've been 1, but it has to cancel out the fourth-order polynomial

$$\phi(q) = \omega(q_{\backslash 0}) \begin{pmatrix} q_{\backslash 0}^2 \\ \sum_i q_{\backslash i}^2 \\ q_{\backslash 0}\, q_{\backslash 1} \\ q_{\backslash 0}\, q_{\backslash 2} \\ q_{\backslash 0}\, q_{\backslash 3} \end{pmatrix} \quad \text{and} \quad \psi(k) = \omega(k_{\backslash 0}) \begin{pmatrix} -\sum_i k_{\backslash i}^2 \\ -k_{\backslash 0}^2 \\ 2k_{\backslash 0}\, k_{\backslash 1} \\ 2k_{\backslash 0}\, k_{\backslash 2} \\ 2k_{\backslash 0}\, k_{\backslash 3} \end{pmatrix} \quad \text{with} \quad \omega(x) = \frac{x}{x^2 + \epsilon}\,.$$

Attention

- Inner product doesn't take e_0-containing elements into account
- Solution: add nonlinear features
- …Cool - wait, this is just the square of a difference
- Transformation identity; rotation preserves norm; omega takes untransformed q/k
- Related to conformal geometric algebra
  - PGA is a subalgebra of CGA
  - CGA contains an infinity element
- Stability tricks
  - Omega could've been 1, but it has to cancel out the fourth-order polynomial

We find it beneficial to add learnable weights as prefactors to each of these three terms. The attention weights are then given by

$$\mathrm{Softmax}_i\left(\frac{\alpha \sum_c \langle q_{i'c}^{MV}, k_{ic}^{MV}\rangle + \beta \sum_c \phi(q_{i'c}^{MV}) \cdot \psi(k_{ic}^{MV}) + \gamma \sum_c q_{i'c}^{s} k_{ic}^{s}}{\sqrt{13 n_{MV} + n_s}}\right)$$

Results: N-body

- 1-10 bodies
- 1000 timesteps
- Heliocentric generation
- MSE loss
- In summary, it's good

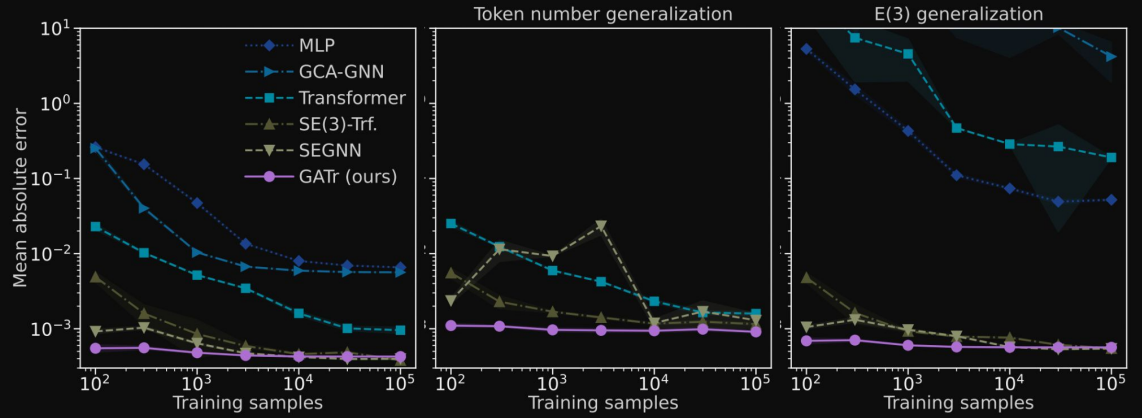


Figure 2: n-body dynamics experiments. We show the error in predicting future positions of planets as a function of the training dataset size. Out of five independent training runs, the mean and standard error are shown. **Left**: Evaluating without distribution shift. GATr (——)is more sample efficient than the equivariant SEGNN [6] (– –) and SE(3)-Transformer [20] (—·) and clearly outperforms non-equivariant baselines (·····, —·, – –). **Middle**: Evaluating on systems with more planets than trained on. Transformer architectures generalize well to different object counts. GCA-GNN has larger errors than the visible range. **Right**: Evaluating on translated data. Because GATr is E(3) equivariant, it generalizes under this domain shift.

Results: arterial wall-shear-stress

- Meshes of arteries + estimated wall-shear-stress
  - Predictor of aneurysms
- Arteries can be oriented in different ways
- With relatively straight arteries, we can canonicalize
- Transformer performs the same then

Method	Mean approx. error [%]
GATr (default)	**6.1**
Without multivector representations (= Transformer)	10.5
Without aux. scalar representations	10.4
Without equivariance constraints on linear maps	9.7
Less wide (4 MV channels + 8 scalar channels)	12.0
Less deep (5 blocks)	7.8

Table 3: Ablation experiments on the arterial wall-shear-stress dataset. We report the mean approximation error (lower is better) on the validation set after training for 100 000 steps.

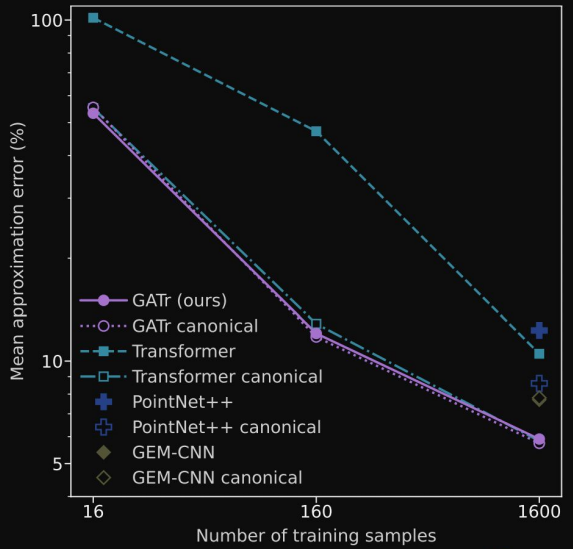
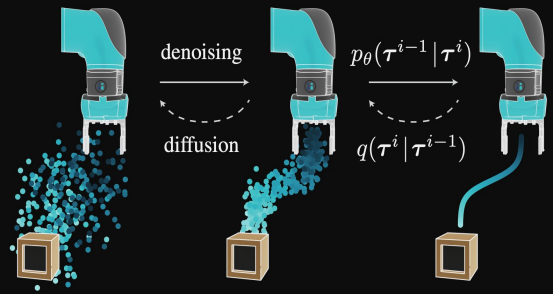


Figure 3: Arterial wall-shear-stress estimation [48]. We show the mean approximation error (lower is better) as a function of training dataset size, reporting results both on randomly oriented training and test samples (solid markers) and on a version of the dataset in which all artery meshes are canonically oriented (hollow markers). Without canonicalization, GATr (—) predicts wall shear stress more precisely and is more sample-efficient than the baselines.

Results: diffusion

- Diffusion-based robotic planning
- Block stacking
- Equivariant diffusion [2]



[3]

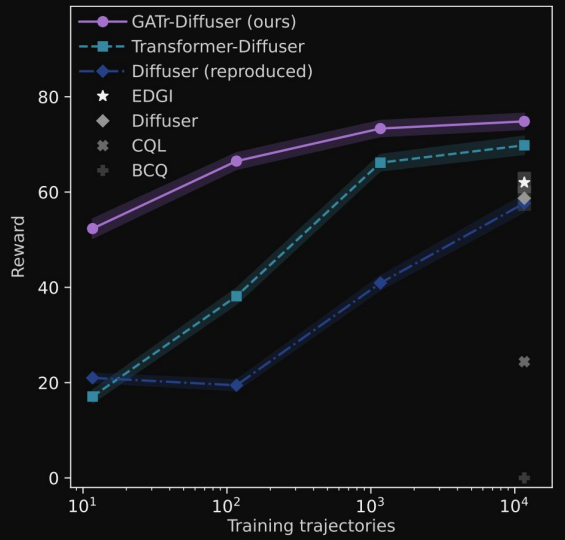


Figure 4: Diffusion-based robotic planning. We show normalized rewards (higher is better) as a function of training dataset size. GATr (——) is more successful at block stacking and more sample-efficient than the baselines, including the original Diffuser [27] (—·—) and our modification of it based on a Transformer (– –). In grey, we show results reported by Brehmer et al. [7] and Janner et al. [27].

Scaling

- Overhead from bigger latents
- Attention becomes the bottleneck
- Uses xformers
- SEGNN is fully connected

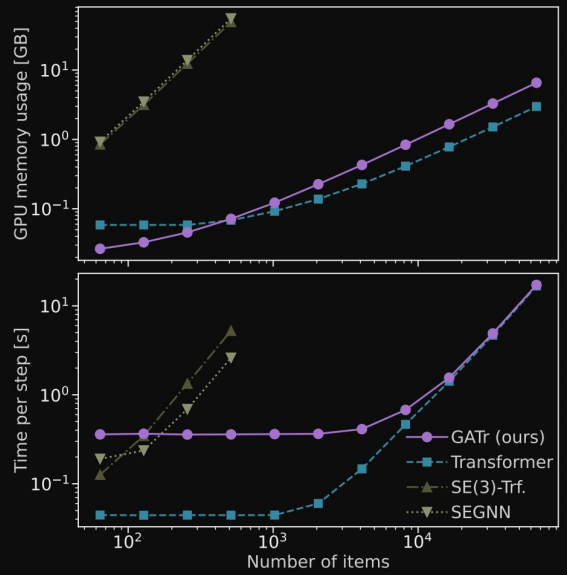


Figure 5: Computational cost and scaling. We measure peak GPU memory usage (top) and wall time (bottom) per combined forward and backward pass as a function of the number of items in synthetic data. Despite some compute overhead, GATr (—) scales just like the Transformer (– –) and orders of magnitude more favorably than the equivariant baselines (—·, – –).

My thoughts

- Nice setup
- Linear layer is great
- Bilinear layer looks hacky
  - Somewhat similar with attention
    - Waiting for the Conformal Geometric Algebra follow-up
- Try interpretability?
  - You can see the per-element activations in 3D
  - Do specific multivectors always represent the same geometric objects?

References

1. Geometric Algebra Transformers
   - ICML 2023, Johann Brehmer, Pim de Haan, Sönke Behrends, Taco Cohen
2. Equivariant flows: exact likelihood generative learning for symmetric densities.
   - ICML 2020, Jonas Köhler, Leon Klein, and Frank Noé.
3. Planning with Diffusion for Flexible Behavior Synthesis
   - ICML 2022, Michael Janner, Yilun Du, Joshua B. Tenenbaum, Sergey Levine
4. A Swift Introduction to Geometric Algebra
   - YouTube 2020, sudgylacmoe
5. Addendum to A Swift Introduction to Geometric Algebra
   - YouTube 2022, sudgylacmoe

$$x, y \mapsto (x^* \wedge y^*)^*$$

$$\mathbb{G}_{3,0,1}$$