Efficient Equivariant Transformer for Self-Driving Agent Modeling

Scott Xu[1,2], **Dian Chen**[†], **Kelvin Wong**[1,2], **Chris Zhang**[1,2], **Kion Fallah**[†], **Raquel Urtasun**[1,2]
Waabi[1], University of Toronto[2]
{sxu, kwong, czhang, urtasun}@waabi.ai

Abstract

Accurately modeling agent behaviors is an important task in self-driving. It is also a task with many symmetries, such as equivariance to the order of agents and objects in the scene or equivariance to arbitrary roto-translations of the entire scene as a whole; i.e.*, SE(2)-equivariance. The transformer architecture is a ubiquitous tool for modeling these symmetries. While standard self-attention is inherently permutation equivariant, explicit pairwise relative positional encodings have been the standard for introducing SE(2)-equivariance. However, this approach introduces an additional cost that is quadratic in the number of agents, limiting its scalability to larger scenes and batch sizes. In this work, we propose DriveGATr, a novel transformer-based architecture for agent modeling that achieves SE(2)-equivariance without the computational cost of existing methods. Inspired by recent advances in geometric deep learning, DriveGATr encodes scene elements as multivectors in the 2D projective geometric algebra $\mathbb{R}^*_{2,0,1}$ and processes them with a stack of equivariant transformer blocks. Crucially, DriveGATr models geometric relationships using standard attention between multivectors, eliminating the need for costly explicit pairwise relative positional encodings. Experiments on the Waymo Open Motion Dataset demonstrate that DriveGATr is comparable to the state-of-the-art in traffic simulation and establishes a superior Pareto front for performance* vs. *computational cost.*

1. Introduction

Understanding how traffic agents behave has many important applications in self-driving, from online motion forecasting for autonomy to offline traffic modeling for simulation. In these applications, a traffic scene typically consists of a set of geometric objects representing traffic agents, like vehicles and pedestrians, and contextual elements, like the map lane graph and traffic signals [9, 10, 12]. Our goal is to learn an agent model that accurately predicts the actions of each agent in the scene from this context. This task has many symmetries, chief among which is equivariance to arbitrary 2D roto-translations of the scene; *i.e.*, SE(2)-equivariance. That is, if we apply a rigid transformation $\mathcal{T} \in \mathrm{SE}(2)$ to the input scene, the outputs of the model should also transform by that same rigid transformation, preserving their relative geometric relationships.

A natural question arises: how can we imbue our agent models with this equivariance? A common approach is to rely on data diversity. The hope is that with enough training examples, the model naturally learns a roughly equivariant function. While conceptually simple, this approach can be sample and compute inefficient and yet still struggle to generalize to new scenes as it is not truly equivariant. Alternatively, one can directly encode this property as an inductive bias into the model, designing the network architecture such that it is equivariant by construction. Approaches in this category typically explicitly model pairwise relationships between agents. For instance, one can transform the context around each agent into its own respective coordinate frame and process each agent's context independently [5, 14] or use pairwise relative positional encoding to process all agents simultaneously [4, 20]. Unfortunately, all of these approaches introduce an additional cost that scales quadratically with the number of agents, limiting its suitability to scale to ever larger scenes, models, datasets, *etc*.

In this paper, we propose DriveGATr, a novel architecture for modeling agents that achieves SE(2)-equivariance without the computational cost of existing methods. Our approach builds on recent advances in geometric deep learning. In particular, we develop a novel extension of the Geometric Algebra Transformer (GATr) [2] to SE(2)-equivariance and new architectural primitives that improve its efficacy for agent modeling. First, we propose an efficient encoding of scene elements into the 2D projective geometric algebra $\mathbb{R}^*_{2,0,1}$, which allows for a native representation of 2D objects (*e.g.*, points, lines) and operators (*e.g.*, translations, rotations) as 8-dimensional multivectors. Next, we develop SE(2)-equivariant neural network primitives, such as linear / bilinear layers, nonlinearities, normalization and scaled dot product attention. In addition, we design a new primitive to transform equivariant geometric representations into invariant features, since an agent's actions are ultimately

[†] Work done while at Waabi.

invariant. These primitives form the basis of DriveGATr's transformer-based architecture.

Our approach has two key advantages. First, compared to non-equivariant models, DriveGATr is equipped with a geometric inductive bias that improves its expressivity, sample efficiency, and robustness to nuissance transformations. Second, unlike prior equivariant methods, DriveGATr achieves this without explicit pairwise relative positional encodings. Instead, it models geometric relationships between agents and scene elements using standard scaled dot product attention between multivectors, which have been heavily optimized [6]. This gives us flexibility to improve the model's performance by growing each agent's context (*e.g.*, to all agents or the entire lane graph) or scale to larger scenes, models, and datasets efficiently. Our experiments on the Waymo Open Motion Dataset [8] confirm these attributes, showing that DriveGATr achieves comparable results to the state-of-the-art in traffic simulation while establishing a superior Pareto front in performance *vs*. computation cost compared to the baselines.

To summarize, we introduce DriveGATr, a tailoring of GATr to acheive efficient $\mathrm{SE}(2)$-equivariance with internal geometric algebra representations. We provide theoretical analysis showing it is provably equivariant by construction. We develop novel components targeted for agent modeling in self-driving, and empirically validate the effectiveness through extensive large scale experiments.

2. Related work

2.1. SE(2) equivariance in traffic modeling

A natural symmetry in traffic modeling is equivariance to 2D roto-translations of the scene; *i.e.*, SE(2) equivariance. This symmetry gives rise to a powerful inductive bias for improving our traffic models' expressivity, efficiency, and generalization [15]. Indeed, equivariant models tend to outperform their non-equivariant baselines at any compute or data budget [3]. While the advantage of this inductive bias diminishes at scale, the scale required far exceeds that of any public self-driving datasets today, where equivariant models still top the leaderboards [17, 18]. Moreover, at a fixed compute budget, the optimal equivariant models are smaller, providing an additional inference time advantage.

To achieve SE(2) equivariance, existing models explicitly compute relative poses between each pair of actors and map elements in the scene [4, 17, 18, 20]. In particular, state-of-the-art models like SMART [17, 18] use a transformer-based architecture that extends the attention dot product with Relative Positional Embeddings (RPE),

$$\alpha_{ij} = \frac{\langle Q_i, K_j + \mathrm{RPE}(\mathrm{pos}_{i\to j})\rangle}{\sqrt{d_k}} \tag{1}$$

where $\mathrm{RPE}(\mathrm{pos}_{i\to j})$ featurizes the relative pose from i to j. Explicitly computing pairwise RPEs scales quadratically with the number of scene elements, limiting the suitability of these models to scale to larger scenes, models, and datasets. Moreover, because they do not use standard scaled dot product attention, these models cannot use highly optimized attention kernels like FlashAttention [6].

To address this limitation, recent work have explored ways to avoid explicit RPEs. For example, DRoPE [19] extends 1D rotary positional embeddings (RoPE) [16] to 2D, modifying the attention dot product so that query and key embeddings are first transformed by blockwise rotation matrices $\mathbf{R}_i$ and $\mathbf{R}_j$ encoding poses of i and j

$$\alpha_{ij} = \frac{\langle \mathbf{R}_i Q_i, \mathbf{R}_j K_j\rangle}{\sqrt{d_k}} \tag{2}$$

While DRoPE bypasses the scalability issue, it lacks expressivity as it does not encode explicit geometric information about the scene elements. It is also translation-equivariant but not rotation-equivariant. Alternatively, VN-Transformer [1] encodes poses as Vector-Neurons [7] and processes them with a transformer with SO(3) equivariant layers. However, for numerical stability, VN-Transformer requires modifications that sacrifice true equivariance.

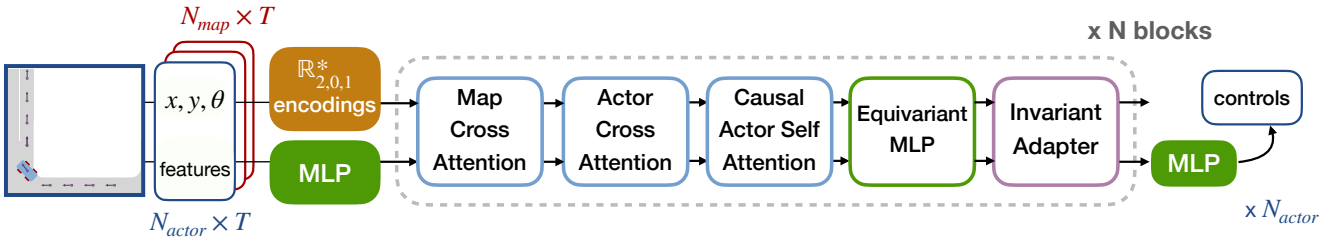
2.2. Geometric Algebra Transformer

Recently, a novel method from geometric deep learning, called the Geometric Algebra Transformer (GATr) [2], proposes an E(3)-equivariant transformer-based architecture which represents geometric objects as elements of a projective geometric algebra. This projective geometric algebra provides a single, unified 16-dimensional algebraic structure to represent all types of geometric data, including points, lines, planes, rotations, and translations. GATr consists of E(3)-equivariant network primitives, including equivariant attention, MLP, and normalization layers, which operate directly on these geometric algebra elements. Because it is built on the standard transformer framework, GATr is scalable, versatile, and efficient.

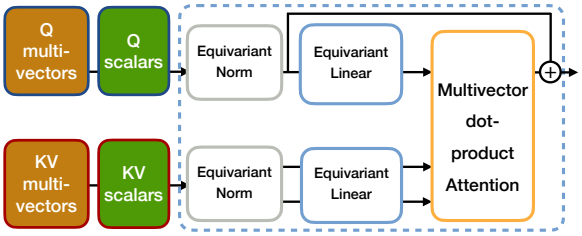
However, applying GATr directly to traffic agent modeling not only results in redundancy, but also inaccuracy. Notably, the original GATr architecture is $\mathrm{E}(3)$-equivariant, whereas a driving scene is only SE(2)-equivariant due to gravity and right-hand traffic rules. In this work, we derive and implement efficient 2D geometric encodings and equivariant layers which operate on an algebraic structure using 8 dimensions instead of 16.

3. DriveGATr

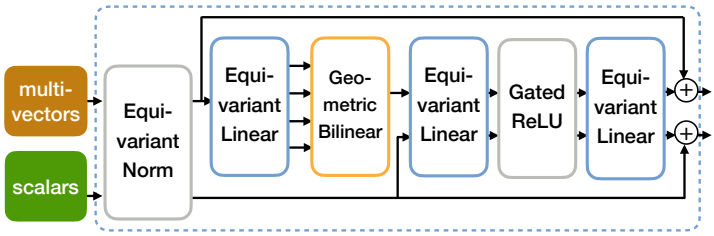
This work is motivated by the goal of designing an efficient and expressive approach to equivariant traffic modelling. As we will see later, a key idea in our proposed method will be to extend the attention dot product with more invariant



(a) Overview of the architecture.



(b) Multivector attention block.



(c) Equivariant MLP block.

Figure 1. The DriveGATr architecture. Figure 1a provides an overview. The poses and features of N_{actor} agents and N_{map} nodes in each scene are encoded as multivectors in $\mathbb{R}^*_{2,0,1}$ and scalars. These tensors are processed by N transformer blocks, each consisting of agent and map cross attention, temporal causal self-attention, equivariant MLPs and invariant adapters. Each of these modules contain skip connections, closely mimicking a standard transformer. Figure 1b describes the attention block. For all attention blocks, the query inputs are agent of interests. The key and value inputs are agents and map nodes for the cross attentions, attending across elements per-timestep. For self-attention, attention spans the temporal axis per-agent. Figure 1c describes the equivariant MLP block. Table 1 provides details on $\mathbb{R}^*_{2,0,1}$ encodings. Section 3.4 provides details on each of the equivariant primitive layers. Section 3.5 describes the invariant adapter block.

dot products. For example, if h_{Q_i} and h_{K_j} are 2D heading vectors, one could compute SE(2)-invariant attention logits via

$$\alpha_{ij} = \frac{\langle Q_i, K_j \rangle + \langle h_{Q_i}, h_{K_j} \rangle}{\sqrt{d_k + 2}} \tag{3}$$

so that query tokens attend more strongly to key tokens with similar heading. However, this naive example has several limitations: (i) h_{Q_i} and h_{K_j} are not learnable features, so this approach is not expressive; (ii) this only reasons about headings, and not positions; and (iii) in this example, the token features do not contain any explicit pose information. Using geometric algebra encodings will allow us to address all of these limitations.

The geometric algebra $\mathbb{R}^*_{2,0,1}$ gives us a unified and efficient way to represent 2D geometries, including points, lines, translations, and rotations. We will show how to encode 2D poses as elements of this algebra, called *multivectors*; in particular, the multivector representing the pose (x, y, θ) will have components corresponding to $x, y, \cos\theta$, and $\sin\theta$. To make our geometric features learnable, we will derive SE(2)-equivariant layers $\mathcal{L} : \mathbb{R}^*_{2,0,1} \to \mathbb{R}^*_{2,0,1}$, including linear layers, normalization, and nonlinearities. The geometric algebra is also equipped with an invariant inner product $\langle \cdot, \cdot \rangle : {\mathbb{R}^*_{2,0,1}}^2 \to \mathbb{R}$, which allows us to compute SE(2)-invariant attention logits via

$$\alpha_{ij} = \frac{\langle Q_i, K_j \rangle + \langle Q_i^{MV}, K_j^{MV} \rangle}{\sqrt{d_k + d^{MV}}} \tag{4}$$

where Q_i^{MV} and K_j^{MV} are multivector geometric features and d^{MV} is the dimension of the invariant inner product. Notably, Equation 4 can be computed as standard dot-product attention via efficient kernels by concatenating Q, K with (certain components of) Q^{MV}, K^{KV}.

In the next sections, we first provide an overview of DriveGATr. We then provide a brief preliminary on generic geometric algebra and describe the details of DriveGATr, which is an efficient 2D extension of GATr [2] with modifications for agent modeling. We refer the readers to the supplementary material for additional preliminaries.

3.1. DriveGATr Architecture

Given a history of agent states $\mathcal{A}_t = \{a_1^{1:t}, \ldots, a_N^{1:t}\}$ and map nodes $\mathcal{M} = \{m_1, \ldots, m_M\}$, DriveGATr predicts an action for each agent: $\mu = \text{DriveGATr}(\mathcal{A}, \mathcal{M})$ at the current timestep. A dynamics model f advances the agent states: $a_i^{t+1} = f(a_i^t, \mu_i)$. We repeat this process to unroll the traffic scene in T timesteps. By construction, the trajectories unrolled by the dynamics model are equivariant relative to the scene. Figure 1 provides an overview of the architecture.

For every token in $\mathcal{A}$ and $\mathcal{M}$, we encode its 2D global[1] position (x, y) with heading θ into a single multivector, using the bivector components to encode the point (x, y), and the vector components to encode the line passing through (x, y) with direction angle θ.

We also encode invariant features in auxiliary scalars using a MLP. For agent states, we encode their speed and bounding box dimensions in the auxiliary scalars. For map segments, we encode their length, width, curvature, speed limit, and boundary types.

Our transformer consists of a sequence of factorized attention blocks, where each block consists of (i) multi-head cross attention between the agent states and map tokens, per timestep; (ii) multi-head self attention between the agent states, per timestep; (iii) multi-head causal self-attention between agent states, per agent; (iv) an equivariant MLP; and (v) an invariant adapter described in Section 3.5.

We use a final MLP on the auxiliary scalars to decode invariant action logits for each agent. We find it beneficial to attend agent states to the entire map in our cross-attention layer, rather than the nearest few map tokens.

3.2. Geometric algebra

Geometric product. The geometric product xy of $x, y \in \mathbb{R}^n$ is a way of multiplying vectors together. The geometric product is defined to be bilinear and associative, and is characterized by the fundamental equation $v^2 = \langle v, v \rangle$ where $\langle \cdot, \cdot \rangle$ is the standard inner product. This induces an algebra, called a *Euclidean geometric algebra*, whose elements are called *multivectors*. From the fundamental equation one can deduce that the geometric product is anticommutative on orthogonal basis vectors, i.e. $e_i e_j = -e_j e_i$ for $i \neq j$. Consequently, for $n = 2$ the Euclidean geometric algebra consists of multivectors of the form $x = x' + x_1 e_1 + x_2 e_2 + x_{12} e_{12}$, where x', x_1, x_2, x_{12} are real coefficients and $e_{12} := e_1 e_2$.

This structure is useful because it provides a unified framework for representing and manipulating geometric objects. For example, in 2D, we can rotate a vector $v = (x, y)$ by an angle θ using the geometric product:

$$R_\theta v = (\cos\theta - (\sin\theta) e_{12})(x e_1 + y e_2) \tag{5}$$

$$= (x\cos\theta - y\sin\theta) e_1 + (x\sin\theta + y\cos\theta) e_2 \tag{6}$$

where we have used that $e_1^2 = e_2^2 = 1$ and $e_2 e_1 = -e_1 e_2$.

Projective geometric algebra. Euclidean geometric algebra is only able to represent linear transformations. A translation is not a linear operation, but we can solve this by adding an extra, special dimension, similar to how homogeneous coordinates are used in computer graphics.

[1]Theoretically, the choice of coordinate frame can be arbitrary due to SE(2) equivariance, but for numerical stability we use the ego's initial pose.

A projective geometric algebra is obtained by first adjoining an orthogonal basis vector e_0 to $\mathbb{R}^n$ with norm zero, that is, $e_0^2 = 0$. For $n = 2$, the resulting algebra, denoted by $\mathbb{R}^*_{2,0,1}$, is 8-dimensional, spanned by the scalar 1, three vectors e_0, e_1, e_2, three *bivectors* e_{01}, e_{20}, e_{12}, and one *pseudoscalar* e_{012}.

Geometric algebra primitives. We define additional geometric algebra primitives that will be useful later.

The *wedge product* is another way of multiplying vectors together, defined by $x \wedge y = (xy - yx)/2$. It is bilinear and associative, and has the property that $v \wedge v = 0$ for any vector v. Also, for distinct basis vectors e_i, e_j, from $e_j e_i = -e_i e_j$ one can derive that $e_i \wedge e_j = e_i e_j$ concides with the geometric product. The wedge product also extends to general multivectors.

The *multivector dual* $x \mapsto x^*$ is a linear operator, defined for a basis multivector x as the multivector such that $x \wedge x^*$ equals the pseudoscalar.

The *join* operator of two multivectors is defined as $\text{Join}(x, y) = (x^* \wedge y^*)^*$.

The *k-blade projection* $\langle x \rangle_k$ is defined to select specific components of a multivector $x \in \mathbb{R}^*_{2,0,1}$

$$\langle x \rangle_0 := x' \tag{7}$$

$$\langle x \rangle_1 := x_0 e_0 + x_1 e_1 + x_2 e_2 \tag{8}$$

$$\langle x \rangle_2 := x_{01} e_{01} + x_{20} e_{20} + x_{12} e_{12} \tag{9}$$

$$\langle x \rangle_3 := x_{012} e_{012} \tag{10}$$

We define the *invariant inner product* $\langle \cdot, \cdot \rangle$ between two multivectors in $\mathbb{R}^*_{2,0,1}$ as

$$\langle x, y \rangle := x'y' + x_1 y_1 + x_2 y_2 + x_{12} y_{12} \in \mathbb{R} \tag{11}$$

which ignores components containing e_0.

3.3. Encodings

We encode 2D objects and transformations into $\mathbb{R}^*_{2,0,1}$ using Table 1 so that geometric features can be computed using the operations defined above. Following Brehmer et al. [2], we encode 2D geometries and transforms such that the result of applying a transform to a geometry is given by a sandwich product. For example, if t is a multivector representing a translation by (a, b) and p is a multivector representing the point (x, y), then tpt^{-1} is a multivector representing the point $(x + a, y + b)$. Given this property, we can state that a mapping $\mathcal{L} : \mathbb{R}^*_{2,0,1} \to \mathbb{R}^*_{2,0,1}$ is SE(2) equivariant if for any roto-translation $u \in \mathbb{R}^*_{2,0,1}$ and input $x \in \mathbb{R}^*_{2,0,1}$, we have

$$\mathcal{L}(uxu^{-1}) = u\mathcal{L}(x)u^{-1} \tag{12}$$

The encodings in Table 1 additionally satisfy the properties that (i) the intersection point of two lines $\ell_1, \ell_2 \in \mathbb{R}^*_{2,0,1}$

Geometry / transform	Representation	Transform inverse
Line $ax + by + c = 0$	$ae_1 + be_2 + ce_0$	-
Point (x, y)	$xe_{20} + ye_{01} + e_{12}$	-
Translation by (a, b)	$1 - \frac{a}{2}e_{01} + \frac{b}{2}e_{20}$	$1 + \frac{a}{2}e_{01} - \frac{b}{2}e_{20}$
Counterclockwise rotation of angle θ	$\cos\frac{\theta}{2} - \sin\frac{\theta}{2}e_{12}$	$\cos\frac{\theta}{2} + \sin\frac{\theta}{2}e_{12}$

Table 1. 2D encodings of geometries and transforms.

is given by their wedge product $p := \ell_1 \wedge \ell_2$; and (ii) the line joining two points $p_1, p_2 \in \mathbb{R}^*_{2,0,1}$ is given by a join operator $\ell := \text{Join}(p_1, p_2)$. For a complete proof of the properties discussed above, please refer to the supplemental material.

3.4. Equivariant layers

In this section we define the primitive equivariant layers of our model. For the full proofs that each of our layers satisfy Equation 12, please refer to the supplemental material.

Linear layers. We define $SE(2)$-equivariant linear maps $\phi : \mathbb{R}^*_{2,0,1} \to \mathbb{R}^*_{2,0,1}$ by the form

$$\phi(x) = \sum_{k=0}^{3} w_k \langle x \rangle_k + \sum_{k=0}^{2} v_k e_0 \langle x \rangle_k + \sum_{k=0}^{2} u_k e_{012} \langle x \rangle_k \quad (13)$$

where w_k, v_k, u_k are parameters and $\langle \cdot \rangle_k$ denotes blade projection. Since ϕ is linear and the map $x \mapsto uxu^{-1}$ is linear for any fixed operator u, to show that ϕ is equivariant it suffices to verify that $\phi(uxu^{-1}) = u\phi(x)u^{-1}$ for any *basis* multivector x and rotation or translation u; please refer to the supplement for details.

We then define affine layers between multivector arrays $\Phi : {\mathbb{R}^*_{2,0,1}}^c \to {\mathbb{R}^*_{2,0,1}}^{c'}$ via

$$\Phi(x)_i = \sum_{j=1}^{c} \phi^{(i,j)}(x_j) + b_i \qquad \forall i = 1, \ldots, c' \quad (14)$$

where $\phi^{(i,j)}$ are equivariant linear layers and b_i are learnable scalar bias terms.

Geometric bilinears. To increase expressivity, we include the geometric product which is equivariant as $(uxu^{-1})(uyu^{-1}) = u(xy)u^{-1}$ for any x, y and operator u, and we include the join operator which is also equivariant Brehmer et al. [2]. These are combined to form the equivariant geometric layer $\text{Geometric}(w, x, y, z) = \text{Concatenate}(wx, \text{Join}(y, z))$.

Nonlinearities and normalization. We define the equivariant scalar-gated activation $\text{GatedRELU}(x) = \text{RELU}(\langle x \rangle_0)x$. We define equivariant normalization over channels by $\text{LayerNorm}(x) = x/\sqrt{\mathbb{E}\langle x, x \rangle + \varepsilon}$ where $\langle \cdot, \cdot \rangle$ is the invariant inner product.

Scaled dot-product attention. For multivector query and key tokens $q, k \in {\mathbb{R}^*_{2,0,1}}^C$, the corresponding invariant attention logit may be computed as

$$\frac{\sum_{c=1}^{C} \langle q_c, k_c \rangle}{\sqrt{4C}} \quad (15)$$

where $\langle \cdot, \cdot \rangle$ is the invariant inner product. We find it crucial to follow Brehmer et al. [2] and extend the query and key multivectors with "distance awareness" features

$$\phi(q) = \frac{q_{12}}{q_{12}^2 + \varepsilon} \begin{pmatrix} q_{12}^2 \\ q_{01}^2 + q_{20}^2 \\ q_{01} q_{12} \\ q_{20} q_{12} \end{pmatrix} \quad (16)$$

$$\psi(k) = \frac{k_{12}}{k_{12}^2 + \varepsilon} \begin{pmatrix} -k_{01}^2 - k_{20}^2 \\ -k_{12}^2 \\ 2k_{01}k_{12} \\ 2k_{20}k_{12} \end{pmatrix} \quad (17)$$

which have the property that $\phi(q) \cdot \psi(k)$ is invariant, and moreover if the bivector components of q and k represent points with $q_{12} = k_{12} = 1$ then $\phi(q) \cdot \psi(k) = -\frac{1}{(1+\varepsilon)^2}[(k_{01} - q_{01})^2 + (k_{20} - q_{20})^2]$ which is proportional to the negative squared distance between the two points.

Combining the multivector features, extended features, and auxiliary scalar features $q^s, k^s \in \mathbb{R}^{C'}$, the invariant attention logits are computed as

$$\frac{\sum_{c=1}^{C} \langle q_c, k_c \rangle + \sum_{c=1}^{C} \phi(q_c) \cdot \psi(k_c) + \sum_{c=1}^{C'} q_c^s k_c^s}{\sqrt{4C + 4C + C'}}. \quad (18)$$

As in a standard transformer, the attention logits are softmaxed and used to take a linear combination of the value tokens. Notably, Equation 18 can be computed as one dot-product by concatenating the three terms for both query and key, allowing drop-in usage of efficient attention kernels.

3.5. Invariant adapter

The output of our model is an invariant action which can be decoded from the auxiliary scalar features. However, the multivector features contain important geometric information that should be used as well. To this end, we introduce

a novel transform which converts each agent's equivariant geometric features into invariant features by transforming the multivectors into a local coordinate frame.

Specifically, given equivariant multivector features $v \in {\mathbb{R}^*_{2,0,1}}^{N \times d}$ and auxiliary scalar features $s \in \mathbb{R}^{N \times d'}$, for each agent $n = 1, \ldots, N$ we compute

$$s_n \leftarrow \text{MLP}(\text{flatten\_components}(u_n v_n u_n^{-1})) + s_n \quad (19)$$

where $u_n \in \mathbb{R}^*_{2,0,1}$ is the roto-translation transforming global poses into the coordinate frame of agent n. Similar to all other layers, Equation 19 is batched across scenes and agent.

For each n, $u_n v_n u_n^{-1}$ is invariant because it converts global equivariant features v_n into agent-centric features. Therefore, this transform preserves the invariance of the auxiliary scalars.

4. Experiments

We evaluate DriveGATr in traffic simulation — a representative task for modeling traffic agents. Our experiments demonstrate two key properties: (1) comparable performance to the state-of-the-art; and (2) a superior Pareto front in performance *vs*. computational costs compared to prior work.

4.1. Comparison to state-of-the-art

In traffic simulation, equivariant models top the leaderboard but sacrifice computational efficiency due to their explicit relative positional encodings. We propose DriveGATr as an alternative that achieves SE(2)-equivariance without the high computational costs of existing methods. Our first experiment seeks to answer the question: Does DriveGATr match the state-of-the-art?

Dataset & metrics. We evaluate our method on the Waymo Open Motion Dataset (WOMD) [8] using the protocol proposed by the Waymo Open Sim Agents Challenge (WOSAC) [11]. Given 1s of context in a traffic scenario, the task is to generate 32 rollouts of 8s closed-loop simulations at 10 Hz, each containing the trajectories of every agent in the scenario. The rollouts are evaluated using distribution matching metrics along three dimensions— kinematic, interactive, and map-based metrics—which are then summarized into a single Realism Meta-Metric (RMM) score. We also report the lowest displacement error, averaged over time, across rollouts (minADE).

Baselines. We compare against the state-of-the-art on WOSAC: **BehaviorGPT** [20], **SMART** [17], and **CAT-K** [18]. BehaviorGPT and SMART use explicit relative positional encodings (RPEs). CAT-K further trains SMART with closed-loop fine-tuning.

We also compare against four baselines representative of how state-of-the-art methods achieve SE(2)-equivariance. These baselines share the same base architecture and learning algorithm as DriveGATr, differing only in how they introduce equivariance. **Transformer** is a scene-centric model that uses roto-translation data augmentation to approximate equivariance and is based on Trajeglish [13]. **Transformer + DRoPE** [19] uses rotary positional encodings (RoPE) to achieve translation (but not rotation) equivariance. We faithfully reproduce DRoPE using publicly available information. **Transformer + RPE** [4] uses explicit RPEs to achieve equivariance. Like other methods that use RPEs, we limit each agent's context to a neighbourhood (eight agents and four map tokens) so that it fits in GPU memory during training.

Implementation details. We use a clustered discretized action space. Starting with a large number of state-to-state transitions, we use a k-disk procedure [13] to select a vocabulary of 2048 actions for each agent class (vehicle, cyclist, pedestrian). We then use this vocabulary to discretize the trajectories in our training dataset, and use a cross entropy loss to train our model to predict the next action for each agent. We use 16 multivector channels, and six factorized attention blocks. Our 3M variant uses auxilliary dimension 128 for a total of 2.7 million parameters, and our 30M variant uses auxilliary dimension 512 for 29.4 million parameters. Similar to prior methods [17, 18], we embed each query token with its prediction at the last timestep as well as its actor class. For all models and baselines, we train for 250,000 steps with learning rate 10^{-3} and cosine annealing. We conduct analysis on the 3M variant, except for the comparison in Table 2, for efficiency purposes.

Results. Table 2 reports results on the WOSAC 2024 validation set. DriveGATr achieves comparable realism scores to the state-of-the-art despite its relatively small model size. In fact, DriveGATr achieves a 2% improvement in RMM over BehaviorGPT—a state-of-the-art model of similar size. The 30M variant of DriveGATr achieves comparable RMM with SMART [17], the top method on the WOSAC leaderboard. For simplicity, we did not explore closed-loop fine-tuning, top-k sampling, and nucleus sampling in our experiments, but we expect that these techniques will improve performance if tuned properly.

DriveGATr marks a significant improvement over our baselines, despite sharing largely the same base architecture. From a modeling perspective, DriveGATr has two important advantages over the baselines: in contrast to Transformer and Transformer + DRoPE, DriveGATr is SE(2)-equivariant by construction; and in contrast to Transformer + RPE, DriveGATr can grow each agent's context to encompass all agents and map tokens in the scene whereas the baseline is limited to a small neighbourhood due to the memory footprint of its explicit RPEs.

Method	# Params	Finetuned	RMM ↑	Kinematic ↑	Interactive ↑	Map-based ↑	minADE ↓
BehaviorGPT	3M	-	0.7438	0.4254	0.7233	0.7976	1.3804
SMART-7M	7M	-	0.7678	0.4894	0.7306	0.8163	1.3532
SMART-7M + CAT-K	7M	✓	0.7709	0.4937	0.7333	0.8185	1.2953
Transformer	3M	-	0.7257	0.4759	0.7058	0.7654	1.6720
Transformer + RPE	3M	-	0.7251	0.4708	0.6953	0.7808	1.7486
Transformer + DRoPE	3M	-	0.7206	0.4629	0.7017	0.7604	1.9193
DriveGATr-3M (ours)	3M	-	0.7620	0.4859	0.7264	0.8103	1.4192
DriveGATr-30M (ours)	30M	-	0.7636	0.4890	0.7272	0.8120	1.3682

Table 2. **Results on WOSAC 2024 validation set.** RMM is the Realism Meta Metric used for ranking. DriveGATr achieves the best RMM among its baselines. Compared to publicly available state-of-the-art methods, DriveGATr achieves comparable realism without top-k sampling and closed-loop fine-tuning.

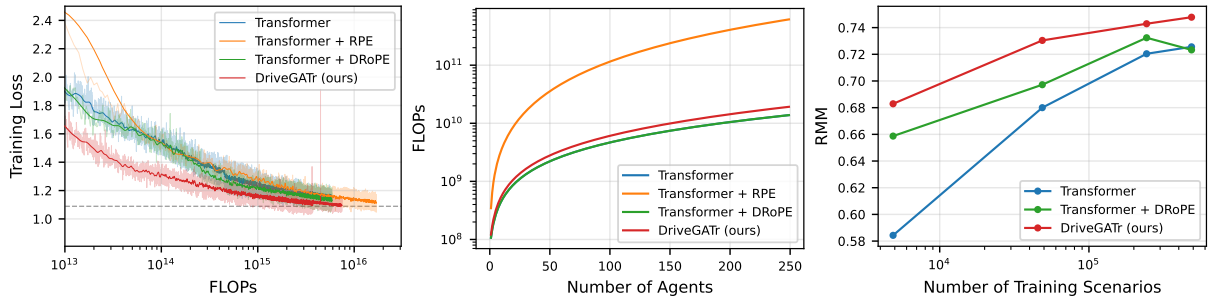


Figure 2. **Training curves.** We compare the envelope of minimal training loss per FLOP (**left**), compute efficiency as the number of agents scale (**middle**), and sample efficiency as the training dataset grows (**right**). Compared to the baselines, DriveGATr establishes a superior Pareto front in performance and computation cost thanks to its compute and sample efficiency.

4.2. Scaling analysis

Our next set of experiments evaluate DriveGATr's scalability with respect to training compute and data.

Experiment setup. For each of our baselines, we train two models with varying training FLOP profiles on the full WOMD training dataset. In our experiments, we vary batch size since we found that varying model size, learning rates, *etc*. was inconsequential. Then, for each training run, we smooth its training loss curve with Gaussian smoothing. Finally, from these training loss curves, we extract the envelope of minimal training loss per FLOP. This training loss envelope gives us an estimate of how each baseline's performance improves as we scale the available training compute.

Results. Figure 2 (left) shows that DriveGATr's training loss envelope is significantly lower than those of the baselines. For any number of training FLOPs, DriveGATr achieves the lowest training loss, demonstrating its superior scalability with training compute. This comes down to DriveGATr's two key advantages: (1) higher sample efficiency due to SE(2)-equivariance; and (2) higher compute efficiency due to how it achieves SE(2)-equivariance.

To demonstrate DriveGATr's compute efficiency, we analyze how each model's number of FLOPs scale with number of agents during inference. From Figure 2 (middle), we see that DriveGATr has significantly lower computational overhead than Transformer + RPE. The latter requires computing explicit RPE, which introduces an additional overhead that is quadratic in the number of agents. DriveGATr avoids this overhead by modeling geometric relationships using standard attention between multivectors.

To demonstrate DriveGATr's sample efficiency, for each model family, we train on a range of dataset sizes: 1%, 10%, 50%, and 100%. Next, we plot each model's RMM on the validation split against the number of scenarios seen during training. Following Zhang et al. [18], we evaluate on 2% of the validation set due to the high cost of evaluation. In Figure 2 (right), we see an intuitive ordering—models with more inductive bias built in tend to have higher sample efficiency.

4.3. Additional analysis

Robustness to roto-translations. SE(2)-equivariance not only improves sample efficiency but also robustness to nuissance transformations. In Figure 3, we compare DriveGATr's robustness to roto-translations against Transformer and Transformer + DRoPE. In each figure, we overlay rollouts from when the model is given inputs in its origi-

Figure 3. **Robustness to roto-translations.** We compare Transformer (**left**), Transformer + DRoPE (**middle**), and DriveGATr (**right**)'s robustness to roto-translations. In each figure, we overlay rollouts from the original coordinate frame vs one rotated by 90°and translated by 100m forward. Blue trajectories visualize model predictions in the original input, and red visualize predictions in the transformed scene. DriveGATr produces consistent trajectories despite closed-loop execution, demonstrating its robustness to roto-translations.

IA	DA	RMM ↑	minADE ↓
✓		0.7408	1.6176
	✓	0.7483	1.6053
✓	✓	0.7478	1.5798

Map Attn.	RMM ↑	minADE ↓
$k = 4$	0.7478	1.5798
$k = 8$	0.7528	1.5293
All	0.7617	1.4174

Table 3. **Ablation studies on the WOSAC 2% validation split.** On the **top**, we ablate the importance of the invariant adapter (IA) and distance-awareness (DA). On the **bottom**, we ablate the number of map tokens each agent attends to in agent-to-map attention.

nal coordinate frame *vs*. one rotated by 90°and translated by 100m forward. Note that for each rollout, instead of sampling, the model at each timestep executes its highest scored action. As expected, Transformer is not robust to roto-translations—its rollouts change significantly when given the same inputs from different coordinate frames. Likewise, Transformer + DRoPE is not robust because it is not rotation equivariant. In contrast, DriveGATr is robust because it has a mathematical guarantee for SE(2)-equivariance.

Ablation studies. In Table 3, we ablate two of DriveGATr's key architectural choices on the 2% validation set, namely (i) whether we use the invariant adapter after each factorized attention block (including the last one) and (ii) whether we use distance-awareness in DriveGATr's attention layers. From the left side of Table 3, we observe that both architectural choices improve DriveGATr's realism. For the invariant adapter, we see improvements in minADE only whereas for distance-awareness, we see improvements across both metrics. This matches our intuitions that there is important geometric information in the multivector features that should be mixed into the auxiliary scalars, and that agents should attend to nearby agents and map tokens.

Latency. We provide memory footprint and latency comparisons in the supplementary material. We note that DriveGATr is faster and more memory efficient than the fully invariant baseline Transformer + RPE.

5. Conclusion

SE(2) equivariance is an inherent symmetry in agents modeling problems. It is a ubiqutous inductive bias in state-of-the-art agent traffic simulation models, improving their expressivity, sample efficiency, and robustness to nuissance transformations. However, existing approaches achieve this with high computational costs. We propose DriveGATr, a novel architecture for modeling traffic agents that guarantees SE(2) equivariance yet avoids the high computational costs of existing approaches. By leveraging efficient 2D geometric algebra encodings and the equivariant layer primitives, DriveGATr is expressive, efficient and provably equivariant.

Limitations and broader impact. Although SE(2) is a useful equivariance property for agent modeling, self-driving is ultimately a 3D problem. DriveGATr models equivariance in 2D but, like prior work [5], it can be easily extended to 2.5D by adding the height dimension to the auxiliary scalar features. Another possibility is combining our method of acheiving SE(2) equivariance in the xy-plane, with translational invariance in the z-axis. While we only evaluate DriveGATr in traffic simulation, we believe it is extensible to similar tasks like motion forecasting and ML-based planning. We also highlight that DriveGATr is capable of making not just invariant but equivariant predictions in the global coordinate frame, making it a suitable architectural choice for more challenging problems such as traffic scenario generation. We leave this direction for future work.

References

[1] Serge Assaad, Carlton Downey, Rami Al-Rfou, Nigamaa Nayakanti, and Benjamin Sapp. Vn-transformer: Rotation-equivariant attention for vector neurons. *TMLR*, 2023. 2

[2] Johann Brehmer, Pim De Haan, Sönke Behrends, and Taco S Cohen. Geometric algebra transformer. In *NeurIPS*, 2023. 1, 2, 3, 4, 5

[3] Johann Brehmer, Sönke Behrends, Pim De Haan, and Taco Cohen. Does equivariance matter at scale? *arXiv*, 2024. 2

[4] Alexander Cui, Sergio Casas, Kelvin Wong, Simon Suo, and Raquel Urtasun. GoRela: Go relative for viewpoint-invariant motion forecasting. In *ICRA*, 2023. 1, 2, 6

[5] Marco Cusumano-Towner, David Hafner, Alex Hertzberg, Brody Huval, Aleksei Petrenko, Eugene Vinitsky, Erik Wijmans, Taylor Killian, Stuart Bowers, Ozan Sener, et al. Robust autonomy emerges from self-play. *arXiv*, 2025. 1, 8

[6] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *NeurIPS*, 2022. 2

[7] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas J. Guibas. Vector neurons: A general framework for so(3)-equivariant networks. *arXiv*. 2

[8] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R. Qi, Yin Zhou, Zoey Yang, Aurelien Chouard, Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov. Large scale interactive motion forecasting for autonomous driving : The waymo open motion dataset. In *ICCV*, 2021. 2, 6

[9] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *CVPR*, 2020. 1

[10] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. In *ECCV*, 2020. 1

[11] Nico Montali, John Lambert, Paul Mougin, Alex Kuefler, Nick Rhinehart, Michelle Li, Cole Gulino, Tristan Emrich, Zoey Yang, Shimon Whiteson, Brandyn White, and Dragomir Anguelov. The waymo open sim agents challenge. *arXiv*, 2023. 6

[12] Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S Refaat, and Benjamin Sapp. Wayformer: Motion forecasting via simple & efficient attention networks. In *ICRA*, 2023. 1

[13] Jonah Philion, Xue Bin Peng, and Sanja Fidler. Trajeglish: Traffic modeling as next-token prediction. In *ICLR*, 2024. 6

[14] Ari Seff, Brian Cera, Dian Chen, Mason Ng, Aurick Zhou, Nigamaa Nayakanti, Khaled S Refaat, Rami Al-Rfou, and Benjamin Sapp. Motionlm: Multi-agent motion forecasting as language modeling. In *ICCV*, 2023. 1

[15] DiJia Andy Su, Bertrand Douillard, Rami Al-Rfou, Cheol Park, and Benjamin Sapp. Narrowing the coordinate-frame gap in behavior prediction models: Distillation for efficient and accurate scene-centric motion forecasting. In *2022 International Conference on Robotics and Automation, ICRA 2022, Philadelphia, PA, USA, May 23-27, 2022*, pages 653–659. IEEE, 2022. 2

[16] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 2024. 2

[17] Wei Wu, Xiaoxin Feng, Ziyan Gao, and Yuheng Kan. Smart: Scalable multi-agent real-time motion generation via next-token prediction. In *NeurIPS*, 2024. 2, 6

[18] Zhejun Zhang, Peter Karkus, Maximilian Igl, Wenhao Ding, Yuxiao Chen, Boris Ivanovic, and Marco Pavone. Closed-loop supervised fine-tuning of tokenized traffic models. In *CVPR*, 2025. 2, 6, 7

[19] Jianbo Zhao, Taiyu Ban, Zhihao Liu, Hangning Zhou, Xiyang Wang, Qibin Zhou, Hailong Qin, Mu Yang, Lei Liu, and Bin Li. Drope: Directional rotary position embedding for efficient agent interaction modeling. *arXiv*, 2025. 2, 6

[20] Zikang Zhou, HU Haibo, Xinhong Chen, Jianping Wang, Nan Guan, Kui Wu, Yung-Hui Li, Yu-Kai Huang, and Chun Jason Xue. Behaviorgpt: Smart agent simulation for autonomous driving with next-patch prediction. In *NeurIPS*, 2024. 1, 2, 6