Wayformer: Motion Forecasting via Simple & Efficient Attention Networks

Nigamaa Nayakanti* nigamaa@waymo.com
Rami Al-Rfou* rmyeid@waymo.com
Aurick Zhou aurickz@waymo.com
Kratarth Goel kratarth@waymo.com
Khaled S. Refaat krefaat@waymo.com
Benjamin Sapp bensapp@waymo.com

Abstract: Motion forecasting for autonomous driving is a challenging task because complex driving scenarios result in a heterogeneous mix of static and dynamic inputs. It is an open problem how best to represent and fuse information about road geometry, lane connectivity, time-varying traffic light state, and history of a dynamic set of agents and their interactions into an effective encoding. To model this diverse set of input features, many approaches proposed to design an equally complex system with a diverse set of modality specific modules. This results in systems that are difficult to scale, extend, or tune in rigorous ways to trade off quality and efficiency.

In this paper, we present Wayformer, a family of attention based architectures for motion forecasting that are simple and homogeneous. Wayformer offers a compact model description consisting of an attention based scene encoder and a decoder. In the scene encoder we study the choice of early, late and hierarchical fusion of input modalities. For each fusion type we explore strategies to trade off efficiency and quality via factorized attention or latent query attention. We show that early fusion, despite its simplicity of construction, is not only modality agnostic but also achieves state-of-the-art results on both Waymo Open Motion Dataset (WOMD) and Argoverse leaderboards, demonstrating the effectiveness of our design philosophy.

Keywords: Motion Forecasting, Trajectory Prediction, Autonomous Driving, Transformer, Robotics, Learning

1 Introduction

In this work, we focus on the general task of future behavior prediction of agents (pedestrians, vehicles, cyclists) in real-world driving environments. This is an essential task for safe and comfortable human-robot interactions, enabling high-impact robotics applications like autonomous driving.

The modeling needed for such scene understanding is challenging for many reasons. For one, the *output* is highly unstructured and multimodal—*e.g.*, a person driving a vehicle could carry out one of many underlying intents unknown to an observer, and representing a distribution over diverse and disjoint possible futures is required. A second challenge is that the *input* consists of a heterogeneous mix of modalities, including agents' past physical state, static road information (*e.g.* location of lanes and their connectivity), and time-varying traffic light information.

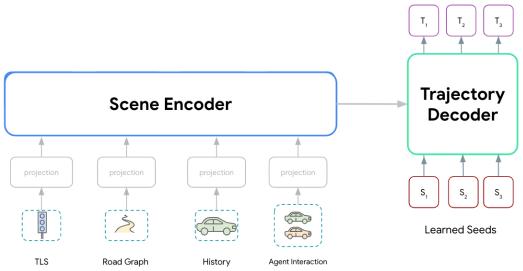


Figure 1: The Wayformer architecture as a pair of encoder/decoder Transformer networks. This model takes multimodal scene data as input and produces multimodal distribution of trajectories.

Many previous efforts address how to model the multimodal output [1, 2, 3, 4, 5, 6], and develop

*Equal contribution.

hand-engineered architectures to fuse different input types, each requiring their own preprocessing (*e.g.*, image rasterization [7, 2, 8]). Here, we focus on the multimodality of the *input space*, and develop a simple yet effective modality-agnostic framework that avoids complex and heterogeneous architectures, and leads to a simpler architecture parameterization. This compact description of a family of architectures results in a simpler design space and allows us to more directly and effectively control for trade-offs in model quality and latency by tuning model computation and capacity.

To keep complexity under control without sacrificing quality or efficiency, we need to find general modeling primitives, which can handle multimodal features that exist in temporal and spatial dimensions concurrently. Recently, several approaches proposed Transformer networks as the networks of choice for motion forecasting problems [9, 10, 11, 12, 13]. While these approaches offer simplified model architectures, they still require domain expertise and excessive modality specific tuning. [14] proposed a stack of cross attention layers sequentially processing one modality at a time. The order in which to process each modality is left to the designer and enumerating all possibilities is combinatorially prohibitive. [3] proposed using separate encoders for each modality, where the type of network and its capacity is open for tuning on a per-modality basis. Then modalities' embeddings are flattened and one single vector is fed to the predictor. While these approaches allow for many degrees of freedom, they increase the search space significantly. Without efficient network architecture search or significant human input and hand engineering, the chosen models will likely be sub-optimal given that a limited amount of the modeling options have been explored.

Our experiments suggest the domain of motion forecasting conforms to Occam's Razor. We show state of the art results with the simplest design choices and making minimal domain specific assumptions, which is in stark contrast to previous work. When tested in simulation and on real AVs, these Wayformer models showed good understanding of the scene.

Our contributions can be summarized as follows:

- We design a family of models with two basic primitives: a *self-attention encoder*, where we fuse one or more modalities across temporal and spatial dimensions, and a *cross-attention decoder*, where we attend to driving scene elements to produce a diverse set of trajectories.
- We study three variations of the scene encoder that differ in how and when different input modalities are fused.
- To keep our proposed models within practical real time constraints of motion forecasting, we study two common techniques to speed up self-attention: *factorized attention* and *latent query attention*.
- We achieve state-of-the-art results on both WOMD and Argoverse challenges.

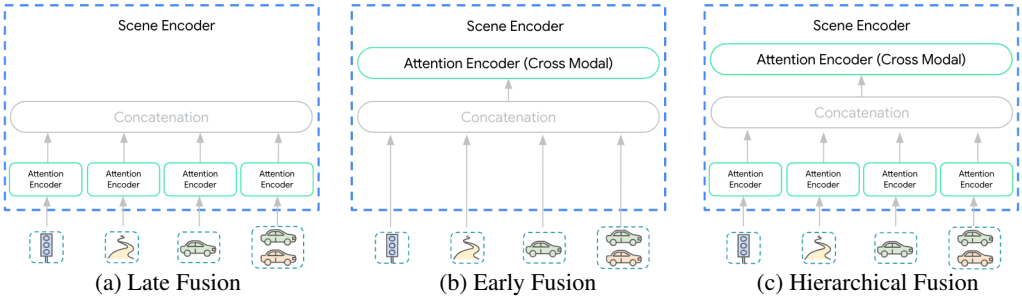
2 Multimodal Scene Understanding

Driving scenarios consist of multimodal data, such as road information, traffic light state, agent history, and agent interactions. In this section we detail the representation of these modalities in our setup. For readability, we define the following symbols: A denotes the number of modeled ego-agents, T denotes the number of past and current timesteps being considered in the history, with a feature size D_m. For a modality m, we might have a 4^{th} dimension (S_m) representing a "set of contextual objects" (i.e. representations of other road users) for each modeled agent.

Agent History contains a sequence of past agent states along with the current state $[A, T, 1, D_h]$. For each timestep $t \in T$, we consider features that define the state of the agent e.g. x, y, velocity, acceleration, bounding box and so on. We include a context dimension $S_h = 1$ for homogeneity.

Agent Interactions The interaction tensor $[A, T, S_i, D_i]$ represents the relationship between agents. For each modeled agent $a \in A$, a fixed number of the closest context agents $c_i \in S_i$ around the modeled agent are considered. These context agents represent the agents which influence the behavior of our modeled agent. The features in D_i represent the physical state of each context agents (as in D_h above), but transformed into the frame of reference of our ego-agent.

Roadgraph The roadgraph $[A, 1, S_r, D_r]$ contains road features around the agent. Following [2], we represent roadgraph segments as polylines, approximating the road shape with collections of line segments specified by their endpoints and annotated with type information. We use S_r roadgraph segments closest to the modeled agent. Note that there is no time dimension for the road features, but we include a time dimension of 1 for homogeneity with the other modalities.



(a) Late Fusion (b) Early Fusion (c) Hierarchical Fusion

Figure 2: Wayformer scene encoder fusing multimodal inputs at different stages. Late fusion dedicates an attention encoder per modality while early fusion process all inputs within one cross modal encoder. Finally, hierarchical fusion combines both the approaches.

Traffic Light State For each agent $a \in A$, traffic light information $[A, T, S_{tls}, D_{tls}]$ contains the states of the traffic signals that are closest to that agent. Each traffic signal point $tls \in S_{tls}$ has features D_{tls} describing the position and confidence of the signal.

3 Wayformer

We design the family of Wayformer models to consist of two main components: a Scene Encoder and a Decoder. The scene encoder is mainly composed of one or more attention encoders that summarize the driving scene. The decoder is a stack of one or more standard transformer cross-attention blocks, in which learned initial queries are fed in, and then cross-attended with the scene encoding to produce trajectories. Figure 1 shows the Wayformer model processing multimodal inputs to produce scene encoding. This scene encoding serves as the context for the decoder to generate k possible trajectories covering the multimodality of the output space.

Frame of Reference As our model is trained to produce futures for a single agent, we transform the scene into an ego-centric frame of reference by centering and rotating the scene's spatial features around the ego-agent's position and heading at the current time step.

Projection Layers Different input modalities may not share the same number of features, so we project them to a common dimension D before concatenating all modalities along the temporal and spatial dimensions $[S, T]$. We found the simple transformation $\text{Projection}(x_i) = \text{relu}(\mathbf{W}x_i + b)$, where $x_i \in \mathbb{R}^{D_m}$, $b \in \mathbb{R}^D$, and $\mathbf{W} \in \mathbb{R}^{D \times D_m}$, to be sufficient. Concretely, given an input of shape $[A, T, S_m, D_m]$ we project its last dimension producing a tensor of size $[A, T, S_m, D]$.

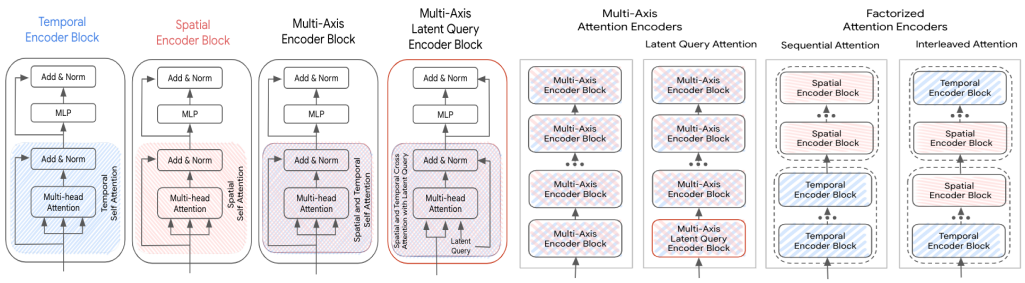
Positional Embeddings Self-attention is naturally permutation equivariant, therefore, we may think of them as set-encoders rather than sequence encoders. However, for modalities where the data does follow a specific ordering, for example agent state across different time steps, it is beneficial to break permutation equivariance and utilize the sequence information. This is commonly done through positional embeddings. For simplicity, we add learned positional embeddings for all modalities. As not all modalities are ordered, the learned positional embeddings are initially set to zero, letting the model learn if it is necessary to utilize the ordering within a modality.

3.1 Fusion

Once projections and positional embeddings are applied to different modalities, the scene encoder combines the information from all modalities to generate a representation of the environment. Concretely, we aim to learn a scene representation $\boldsymbol{Z} = Encoder(\{m_0, m_1, ..., m_k\})$, where $m_i \in \mathbb{R}^{A \times (T \times S_m) \times D}$, $\boldsymbol{Z} \in \mathbb{R}^{A \times L \times D}$, and L is a hyperparameter.

However, the diversity of input sources makes this integration a non-trivial task. Modalities might not be represented at the same abstraction level or scale: {pixels vs objects}. Therefore, some modalities might require more computation than the others. Splitting compute and parameter count among modalities is application specific and non-trivial to hand-engineer. We attempt to simplify the process by proposing three levels of fusion: {Late, Early, Hierarchical}.

Late Fusion This is the most common approach used by motion forecasting models, where each modality has its own dedicated encoder (See Figure 2). We set the width of these encoders to be



(a) Encoder Blocks (b) Encoders

Figure 3: A summary of encoder architectures considered for Wayformer. (a) provides an overview of different encoder blocks and (b) explains how these blocks are arranged to construct the encoder.

equal to avoid introducing extra projection layers to their outputs. Moreover, we share the same depth across all encoders to narrow down the exploration space to a manageable scope. Transfer of information across modalities is allowed only in the cross-attention layers of the trajectory decoder.

Early Fusion Instead of dedicating a self-attention encoder to each modality, early fusion reduces modality specific parameters to only the projection layers (See Figure 2). In this paradigm, the scene encoder consists of a single self-attention encoder ("Cross-Modal Encoder"), giving the network maximum flexibility in assigning importance across modalities with minimal inductive bias.

Hierarchical Fusion As a compromise between the two previous extremes, capacity is split between modality-specific self-attention encoders and the cross-modal encoder in a hierarchical fashion. As done in late fusion, width and depth is common across attention encoders and the cross modal encoder. This effectively splits the depth of the scene encoder between modality specific encoders and the cross modal encoder (Figure 2).

3.2 Attention

Transformer networks do not scale well for large multidimensional sequences due to two factors: (a) Self-attention is quadratic in the input sequence length. (b) Position-wise Feed-forward networks are expensive sub-networks. In the following sections, we discuss different speedups to the transformer networks that will help us scale more effectively.

Multi-Axis Attention This refers to the default transformer setting which applies self-attention across both spatial and temporal dimensions simultaneously (See Figure 3b), which we expect to be the most expensive computationally. Computational complexity of early, late and hierarchical fusions with multi-axis attention is $\mathcal{O}(S_m^2 \times T^2)$.

Factorized Attention Computational complexity of the self-attention is a quadratic in input sequence length. This becomes more pronounced in multi-dimensional sequences, since each extra dimension increases the size of the input by a multiplicative factor. For example, some input modalities have both temporal and spatial dimensions, so the compute cost scales as $\mathcal{O}(S_m^2 \times T^2)$. To alleviate this, we consider factorized attention [15, 16] along the two dimensions. This exploits the multidimensional structure of input sequences by applying self-attention over each dimension individually, which reduces the cost of self-attention sub-network from $\mathcal{O}(S_m^2 \times T^2)$ to $\mathcal{O}(S_m^2) + \mathcal{O}(T^2)$. Note that the linear term still tends to dominate if $\sum_m \mathcal{S}_m \times T << 12 \times D$ [17].

While factorized attention has the potential to reduce computation compared to multi-axis attention, it introduces complexity in deciding the order in which self-attention is applied to each dimension. In our work, we compare two paradigms of factorized attention (see Figure 3b):

- **Sequential Attention:** an N layer encoder consists of $N/2$ temporal encoder blocks followed by another $N/2$ spatial encoder blocks.
- **Interleaved Attention:** an N layer encoder consists of temporal and spatial encoder blocks alternating $N/2$ times.

Latent Query Attention Another approach to address the computational costs of large input sequences is to use latent queries [18, 19] in the first encoder block, where input $x \in \mathbb{R}^{A \times L_{\text{in}} \times D}$ is mapped to latent space $z \in \mathbb{R}^{A \times L_{\text{out}} \times D}$. These latents $z \in \mathbb{R}^{A \times L_{\text{out}} \times D}$ are processed further by a series of encoder blocks that take in and return arrays in this latent space (see Figure 3a). This gives

us full freedom to set the latent space resolution, reducing the computational costs of the both self-attention component and the position-wise feedforward network of each block. We set the reduction value ($R = L_{\text{out}}/L_{\text{in}}$) to be a percentage of the input sequence length. Reduction factor R is kept constant across all the attention encoders in late and hierarchical fusions.

3.3 Trajectory Decoding

As our focus is on how to integrate information from different modalities in the encoder, we simply follow the training and output format of [2, 3], where the Wayformer predictor outputs a mixture of Gaussians to represent the possible trajectories an agent may take. To generate predictions, we use a Transformer decoder which is fed a set of k learned initial queries $(\boldsymbol{S}_i \in \mathbb{R}^h)_{i=1}^k$ and cross attends them with the scene embeddings from the encoder in order to generate embeddings for each component in the output mixture of Gaussians.

Given the embedding Y_i for a particular component of the mixture, we estimate the mixture likelihood with a linear projection layer that produces the unnormalized log-likelihood for the component. To generate the trajectory, we project Y_i using another linear layer to output 4 time series: $T_i = \{\mu_x^t, \mu_y^t, \log \sigma_x^t, \log \sigma_y^t\}_{t=1}^T$ corresponding to the means and log-standard deviations of the predicted Gaussian at each timestep.

During training, we follow [2, 3] in decomposing the loss into separate classification and regression losses. Given k predicted Gaussians $(T_i)_{i=1}^k$, let $\hat{i}$ denote the index of the Gaussian with mean closest to the ground truth trajectory G. We train the mixture likelihoods on the log likelihood of selecting the index $\hat{i}$, and the Gaussian $T_{\hat{i}}$ to maximize the log-probability of the ground truth trajectory.

$$\max \underbrace{\log \Pr(\hat{i} \mid Y)}_{\text{classification loss}} + \underbrace{\log \Pr(G|T_{\hat{i}})}_{\text{regression loss}}. \tag{1}$$

3.4 Trajectory Aggregation

If the predictor outputs a GMM with many modes, it can be difficult to reason about a mixture with so many components, and the benchmark metrics often restrict the number of trajectories being considered. During evaluation, we thus apply trajectory aggregation following [3] in order to reduce the number of modes being considered while still preserving the diversity in the original output mixture. We refer the reader to Appendix C and [3] for details of the aggregation scheme.

4 Experimental Setup

4.1 Datasets

Waymo Open Motion Dataset (WOMD) consists of 1.1M examples time-windowed from 103K 20s scenarios derived from real-world driving in urban and suburban environments. Each example consists of 1 second of history state and 8 seconds of future, which we resample at 5Hz. The object-agent state contains attributes such as position, agent dimensions, velocity and acceleration vectors, orientation, angular velocity, and turn signal state. The long (8s) time horizon in this dataset tests the model's ability to capture a large field of view and scale to a large output space of trajectories.

Argoverse Dataset consists of 333K scenarios containing trajectory histories, context agents, and lane centerline inputs for motion prediction. The trajectories are sampled at 10Hz, with 2 seconds of history and a 3-second future prediction horizon.

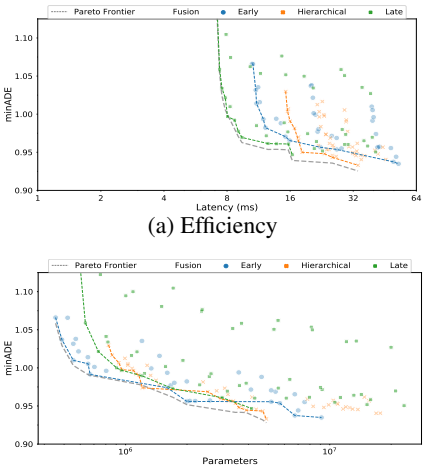
4.2 Training Details and Hyperparameters

We compare models using competition specific metrics associated with these datasets (see Appendix E). For all metrics, we consider only the top $k = 6$ most likely modes output by our model (after trajectory aggregation) and use only the mean of each mode.

For all experiments, we train models using the AdamW optimizer [20] with an initial learning rate of 2e-4 and linearly decaying to 0 over 1M steps. We train models using 16 TPU v3 cores each, with a batch size of 16 per core, resulting in a total batch size of 256 examples per step.

To vary the capacity of the models, we consider hidden sizes among $\{64, 128, 256\}$ and depths among $\{1, 2, 4\}$ layers. We fix the intermediate size in the feedforward network of the Transformer block to be either 2 or 4 times the hidden size.

For our architecture study in Sections (5.1-5.3), each predictor outputs a mixture of Gaussians with $m = 6$ components, with no trajectory aggregation. For our benchmark results in Section 5.4, each predictor outputs a mixture of Gaussians with $m = 64$ components, and we prune the mixture components using the trajectory aggregation scheme described in Section 3.4. For experiments with latent queries, we experiment with reducing the original input resolution to $0.25, 0.5, 0.75$ and 0.9 times the original sequence length. We include a full description of hyperparameters in Appendix B.



(a) Efficiency

(b) Capacity

Figure 4: MinADE of different fusion models with multi-axis attention.

5 Results

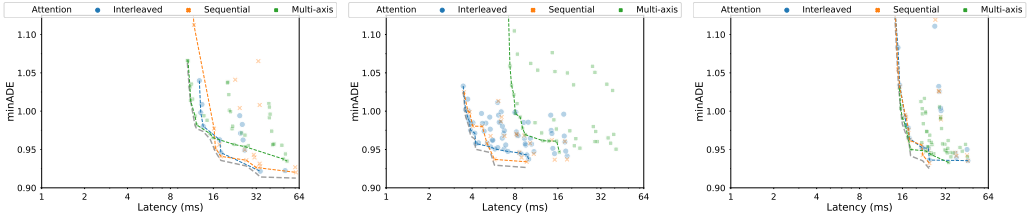
In this Section, we present experiments that demonstrate the trade-offs of combining different fusion strategies with vanilla self-attention (multi-axis) and more optimized methods such as factorized attention and learned queries. In our ablation studies (Section 5.1-5.3), we trained models with varying capacities (0.3M-20M parameters) for 1M steps on WOMD. We report their inference latency on a current generation GPU, capacity, and minADE as a proxy of quality.

5.1 Multi-Axis Attention

In these experiments, we train Wayformer models on early, hierarchical and late fusion (Section 3.1) in combination with multi-axis attention. In Figure (4a), we show that for models with low latency ($x \leq 16$ ms), late fusion represents an optimal choice. These models are computationally cheap since there is no interaction between modalities during the scene encoding step. Adding the cross modal encoder for hierarchical models unlocks further quality gains for models in the range ($16\text{ms} < x < 32\text{ms}$). Finally, we can see that early fusion can match hierarchical fusion at higher computational cost ($x > 32\text{ms}$). We then study the model quality as a function of capacity, as measured by the number of trainable parameters (Figure 4b). Small models perform best with early fusion, but as model capacity increases, sensitivity to the choice of fusion decreases dramatically.

5.2 Factorized Attention

To reduce the computational budget of our models, we train models with factorized attention instead of jointly attending to spatial and temporal dimensions together. When combining different modalities together for the cross modal encoder, we first tile the roadgraph modality to a common temporal dimension as the other modalities, then concatenate modalities along the spatial dimension. After the scene encoder, we pool the encodings over the time dimension before feeding to the predictor.



(a) Early Fusion. (b) Late Fusion. (c) Hierarchical Fusion.

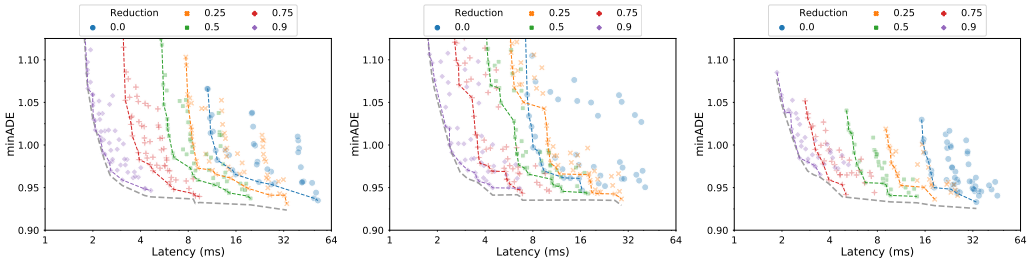
Figure 5: Factorized attention improves quality, but only speeds up late fusion models.

We study two types of factorized attention: sequential, interleaved (Figure 5). First, we observe that both sequential and interleaved factorized attention perform similarly across all types of fusion. Second, we are surprised to see quality gains from applying factorized attention to the early and late fusion cases (Figures 5a, 5b). Finally, we only observe latency improvements for late fusion models (Figure 5b), since tiling the road graph to the common temporal dimension in cross-modal encoder used in early and hierarchical fusion significantly increases the count of tokens.

5.3 Latent Queries

In this study, we train models with multi-axis latent query encoders with varying levels of input sequence length reduction in the first layer as shown in Figure 5. The number of the latent queries

is calculated to be a percentage of the input size of the Transformer network with 0.0% indicating the baseline models (multi-axis attention with no latent queries as presented in Figure 4).



(a) Early Fusion. (b) Late Fusion. (c) Hierarchical Fusion.

Figure 6: Latent queries reduce models' latency without significant degradation to the quality.

Figure 6 shows the results of applying latent queries, which speeds up all fusion models by 2x-16x times with minimal to no quality regression. Early and hierarchical fusion still produce the best quality results, showing the importance of the cross modal interaction stage.

5.4 Benchmark Results

We validate our learnings by comparing Wayformer models to competitive models on popular benchmarks of motion forecasting. We choose early fusion models since they match the quality of the hierarchical models without increased complexity of implementation. Moreover, as models' capacity increases they are less sensitive to the choice of fusion (See Figure 4b). We use latent queries since they speed up models without noticeable quality regression and, in some models, we combine them with factorized attention (see Appendix A) since that improves the quality further. We further apply ensembling, a standard practice for producing SOTA results for leaderboard submissions. Full hyperparameters for Wayformer models reported on benchmarks are reported in Appendix D.

When ensembling for WOMD, the model has a single shared encoder but uses $N = 3$ separate Transformer decoders. To merge predictions over the ensemble, we simply combine all mixture components from each predictor to get a total of $N \times 64$ modes, and renormalize the mixture probabilities. We then apply our trajectory aggregation scheme (section 3.4) to the combined mixture distribution to reduce the number of output modes to the desired count $k = 6$.

In Table 1, we present results on the Waymo Open Motion Dataset and Argoverse Dataset. We use the standard metrics used for the each dataset for their respective evaluation (see Appendix E). For the Waymo Open Motion Dataset, both Wayformer early fusion models outperform other models across all metrics; early fusion of input modalities results in better overall metrics independent of the attention structure (multi-axis or factorized attention).

For Argoverse leaderboard, we train 15 replicas each with its own encoder and $N = 10$ transformer decoders. To merge predictions over N decoders we follow the aggregation scheme in section 3.4 to result in $k = 6$ modes for each model. We then ensemble 15 such replicas following the same aggregation scheme (section 3.4) to reduce $N \times 6$ modes to $k = 6$.

		Waymo Open Motion Dataset					**Argoverse Dataset**			
Models		minFDE (↓)	minADE (↓)	MR (↓)	Overlap (↓)	mAP* (↑)	Brier-minFDE* (↓)	minFDE (↓)	MR (↓)	minADE (↓)
SceneTransformer [11]		1.212	0.612	0.156	0.147	0.279	1.8868	1.2321	0.1255	0.8026
DenseTNT [21]		1.551	1.039	0.157	0.178	0.328	1.9759	1.2858	0.1285	0.8817
MultiPath [2]		2.040	0.880	0.345	0.166	0.409	-	-	-	-
MultiPath++ [3]		1.158	0.556	0.134	0.131	0.409	1.7932	1.2144	0.1324	0.7897
LaneConv		-	-	-	-	-	2.0539	1.3622	0.1600	0.8703
LaneRCNN [22]		-	-	-	-	-	2.1470	1.4526	0.1232	0.9038
mmTransformer [14]		-	-	-	-	-	2.0328	1.3383	0.1540	0.8346
TNT [23]		-	-	-	-	-	2.1401	1.4457	0.1300	0.9400
DCMS [24]		-	-	-	-	-	1.7564	**1.1350**	**0.1094**	**0.7659**
Wayformer Early Fusion	**Attention**									
	LQ + Multi-Axis	1.128	**0.545**	**0.123**	**0.127**	**0.419**	**1.7408**	1.1615	0.1186	0.7675
	LQ + Factorized	**1.126**	**0.545**	**0.123**	**0.127**	0.412	1.7451	1.1625	0.1192	0.7672

Table 1: Wayformer models and select SOTA baselines on Waymo Open Motion Dataset 2021 and Argoverse 2021. * denotes the metric used for leaderboard ranking. LQ denotes latent query.

6 Related Work

Motion prediction architectures : Increasing interest in self-driving applications and the availability of benchmarks [25, 26, 27] has allowed motion prediction models to flourish. Successful

modeling techniques fuse multi-modal inputs that represent different static, dynamic, social and temporal aspects of the scene. One class of models draws heavily from the computer vision literature, rendering inputs as a multichannel rasterized top-down image [4, 2, 28, 29, 7, 23]. In this approach, relationships between scene elements are rendered in the top down orthographic plane and modeled via spatio-temporal convolutional networks. However, the localized structure of convolutions is well suited to processing image inputs, but is not effective at capturing the long range spatio-temporal relationships. A popular alternative is to use an entity-centric approach, where agent state history is typically encoded via sequence modeling techniques like RNNs [10, 30, 31, 32] or temporal convolutions [33]. Road elements are approximated with basic primitives (e.g. piecewise linear segments) which encode pose and semantic information. Modeling relationships between entities is often presented as an information aggregation process, and models employ pooling [23, 34, 31, 35, 10, 28], soft-attention [10, 23] or graph neural networks [36, 33, 30]. Like our proposed method, several recent models use Transformers [37], which are a popular state-of-the-art choice for sequence modeling in NLP [38, 39], and have shown promise in core computer vision tasks such as detection [40, 41, 42], tracking [43] and classification [41, 44].

Iterative cross-attention A recent approach to encode multi-modal data is to sequentially process one modality at a time [14, 19, 9]. [14] ingests the scene in the order {agent history, nearby agents, map}; they argue that it is computationally expensive to perform self-attention over multiple modalities at once. [9] pre-encodes the agent history and contextual agents through self-attention and cross-attends to the map with agent encodings as queries. The order of self-attention and cross-attention relies heavily on the designer's intuition and has, to our knowledge, not been ablated before.

Factorized Attention Flattening high dimensional data leads to long sequences which make self-attention computationally prohibitive. [16] proposed limiting each attention operation to a single axes to alleviate the computational costs and applied this technique to autoregressive generative modeling for images. Similarly, [15] factorize the spatial and temporal dimensions of the video input when constructing their self-attention based classifier. This axis based attention, which gets applied in interleaved fashion across layers, has been adopted in Transformer-based motion forecasting models [9] and graph neural network approaches [12]. The order of applying attention over {temporal, social/spatial} dimensions has been studied with two different common patterns: (a) Temporal first [31, 35, 45] (b) Social/Spatial first [46, 47]. In Section 3.2, we study a 'sequential' mode and contrast it with interleaved mode where interleave dimensions of attention similar to [9].

Multimodal Encoding [13] argued that attending to temporal and spatial dimensions independently leads to loss of information. Moreover, allowing all inputs to self-attend to each other early on the encoding process reduces complexity and the need to handcraft architectures to address the scaling of computation for transformers with the increase in the input sequence length [48]. However, self-attention is known to be computationally expensive for large inputs [49], and recently there has been huge interest in approaches improving its scalability. For a complete discussion of previous works, we refer the reader to the comprehensive survey [50]. One compelling approach is to use learned latent queries to decouples the number of query vectors of a Transformer encoder from the original input sequence length [18]. This allows us to set the resolution of the Transformer output to arbitrary scales independent of the input, and flexibly tune model computational costs. This approach is appealing since it does not assume any structure in the input and has proven effective in fusing multimodal inputs [48]. We take inspiration from such frameworks and present a study of their benefits when applied to the task of motion forecasting in the self-driving domain.

7 Limitations

Scope of the current study is subject to the following limitations: (1) Ego-centric modeling is subject to repeated computations on dense scenes. This can be alleviated by encoding the scene only once in a global frame of reference. (2) Our system input is a sparse abstract state description of the world, which fails to capture some important nuances in highly interactive scenes, e.g., visual cues from pedestrians or fine-granularity contour or wheel angle information for vehicles. Learning perception and prediction end-to-end could unlock improvements. (3) We model the distribution over possible futures independently per agent, and temporally conditionally independent for each agent given intent. These simplifying assumptions allow for efficient computation but fail to fully describe combinatorially many futures. Multi-agent, temporally causal models could show further benefits in interactive situations.

Acknowledgments

We thank Balakrishnan Varadarajan for help on ensembling strategies; Dragomir Anguelov and Eugene Ie for their helpful feedback on the paper.

References

[1] N. Rhinehart, K. Kitani, and P. Vernaza. R2P2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *ECCV*, 2018.

[2] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *CoRL*, 2019.

[3] B. Varadarajan, A. S. Hefny, A. Srivastava, K. S. Refaat, N. Nayakanti, A. Cornman, K. Chen, B. Douillard, C. P. Lam, D. Anguelov, and B. Sapp. Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction. In *ICRA*, 2021.

[4] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2090–2096. IEEE, 2019.

[5] C. Tang and R. R. Salakhutdinov. Multiple futures prediction. In *nips*, 2019.

[6] J. Liang, L. Jiang, K. Murphy, T. Yu, and A. Hauptmann. The garden of forking paths: Towards multi-future trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10508–10518, 2020.

[7] S. Casas, W. Luo, and R. Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *Conf. on Robot Learning*, 2018.

[8] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun. End-to-end interpretable neural motion planner. In *CVPR*, 2019.

[9] R. Girgis, F. Golemo, F. Codevilla, J. A. D'Souza, S. E. Kahou, F. Heide, and C. J. Pal. Latent variable nested set transformers & autobots. *CoRR*, abs/2104.00563, 2021. URL https://arxiv.org/abs/2104.00563.

[10] J. P. Mercat, T. Gilles, N. E. Zoghby, G. Sandou, D. Beauvois, and G. P. Gil. Multi-head attention for multi-modal joint vehicle motion forecasting. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9638–9644, 2020.

[11] J. Ngiam, B. Caine, V. Vasudevan, Z. Zhang, H. L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal, D. Weiss, B. Sapp, Z. Chen, and J. Shlens. Scene transformer: A unified multi-task model for behavior prediction and planning. *CoRR*, abs/2106.08417, 2021. URL https://arxiv.org/abs/2106.08417.

[12] C. Yu, X. Ma, J. Ren, H. Zhao, and S. Yi. Spatio-temporal graph transformer networks for pedestrian trajectory prediction. In *ECCV*, 2020.

[13] Y. Yuan, X. Weng, Y. Ou, and K. Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. *ArXiv*, abs/2103.14023, 2021.

[14] Y. Liu, J. Zhang, L. Fang, Q. Jiang, and B. Zhou. Multimodal motion prediction with stacked transformers. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7573–7582, 2021.

[15] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6836–6846, October 2021.

[16] J. Ho, N. Kalchbrenner, D. Weissenborn, and T. Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

[17] J. Kaplan, S. McCandlish, T. J. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *ArXiv*, abs/2001.08361, 2020.

[18] J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, and Y. W. Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*, 2019.

[19] A. Jaegle, F. Gimeno, A. Brock, A. Zisserman, O. Vinyals, and J. Carreira. Perceiver: General perception with iterative attention. In *ICML*, 2021.

[20] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[21] J. Gu, C. Sun, and H. Zhao. Densetnt: End-to-end trajectory prediction from dense goal sets. *CoRR*, abs/2108.09640, 2021. URL https://arxiv.org/abs/2108.09640.

[22] W. Zeng, M. Liang, R. Liao, and R. Urtasun. Lanercnn: Distributed representations for graph-centric motion forecasting. *CoRR*, abs/2101.06653, 2021. URL https://arxiv.org/abs/2101.06653.

[23] H. Zhao, J. Gao, T. Lan, C. Sun, B. Sapp, B. Varadarajan, Y. Shen, Y. Shen, Y. Chai, C. Schmid, et al. Tnt: Target-driven trajectory prediction. *arXiv preprint arXiv:2008.08294*, 2020.

[24] M. Ye, J. Xu, X. Xu, T. Cao, and Q. Chen. Dcms: Motion forecasting with dual consistency and multi-pseudo-target supervision, 2022.

[25] M. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays. Argoverse: 3d tracking and forecasting with rich maps. *CoRR*, abs/1911.02620, 2019. URL http://arxiv.org/abs/1911.02620.

[26] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska. One thousand and one hours: Self-driving motion prediction dataset. *CoRR*, abs/2006.14480, 2020. URL https://arxiv.org/abs/2006.14480.

[27] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. R. Qi, Y. Zhou, Z. Yang, A. Chouard, P. Sun, J. Ngiam, V. Vasudevan, A. McCauley, J. Shlens, and D. Anguelov. Large scale interactive motion forecasting for autonomous driving : The waymo open motion dataset. *CoRR*, abs/2104.10133, 2021. URL https://arxiv.org/abs/2104.10133.

[28] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. S. Torr, and M. K. Chandraker. DESIRE: distant future prediction in dynamic scenes with interacting agents. *CoRR*, abs/1704.04394, 2017. URL http://arxiv.org/abs/1704.04394.

[29] J. Hong, B. Sapp, and J. Philbin. Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions. *CoRR*, abs/1906.08945, 2019. URL http://arxiv.org/abs/1906.08945.

[30] S. Khandelwal, W. Qi, J. Singh, A. Hartnett, and D. Ramanan. What-if motion prediction for autonomous driving. *ArXiv*, 2020.

[31] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, 2016.

[32] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. In *ECCV*, 2019.

[33] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun. Learning lane graph representations for motion forecasting. *arXiv preprint arXiv:2007.13732*, 2020.

[34] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid. VectorNet: Encoding hd maps and agent dynamics from vectorized representation. In *CVPR*, 2020.

[35] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[36] S. Casas, C. Gulino, R. Liao, and R. Urtasun. Spagnn: Spatially-aware graph neural networks for relational behavior forecasting from sensor data. In *IEEE Intl. Conf. on Robotics and Automation*. IEEE, 2020.

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[38] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[39] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. URL https://arxiv.org/abs/2005.14165.

[40] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, editors, *Computer Vision – ECCV 2020*, pages 213–229, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58452-8.

[41] I. Bello, B. Zoph, A. Vaswani, J. Shlens, and Q. V. Le. Attention augmented convolutional networks. *CoRR*, abs/1904.09925, 2019. URL http://arxiv.org/abs/1904.09925.

[42] A. Srinivas, T. Lin, N. Parmar, J. Shlens, P. Abbeel, and A. Vaswani. Bottleneck transformers for visual recognition. *CoRR*, abs/2101.11605, 2021. URL https://arxiv.org/abs/2101.11605.

[43] W. Hung, H. Kretzschmar, T. Lin, Y. Chai, R. Yu, M. Yang, and D. Anguelov. Soda: Multi-object tracking with soft data association. *CoRR*, abs/2008.07725, 2020. URL https://arxiv.org/abs/2008.07725.

[44] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens. Stand-alone self-attention in vision models. *CoRR*, abs/1906.05909, 2019. URL http://arxiv.org/abs/1906.05909.

[45] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. D. Reid, S. H. Rezatofighi, and S. Savarese. Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. In *NeurIPS*, 2019.

[46] Y. Huang, H. Bi, Z. Li, T. Mao, and Z. qi Wang. Stgat: Modeling spatial-temporal interactions for human trajectory prediction. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6271–6280, 2019.

[47] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone. Trajectron++: Multi-agent generative trajectory forecasting with heterogeneous data for control. *CoRR*, abs/2001.03093, 2020. URL http://arxiv.org/abs/2001.03093.

[48] A. Jaegle, S. Borgeaud, J.-B. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, A. Brock, E. Shelhamer, O. J. H'enaff, M. M. Botvinick, A. Zisserman, O. Vinyals, and J. Carreira. Perceiver io: A general architecture for structured inputs & outputs. *ArXiv*, abs/2107.14795, 2021.

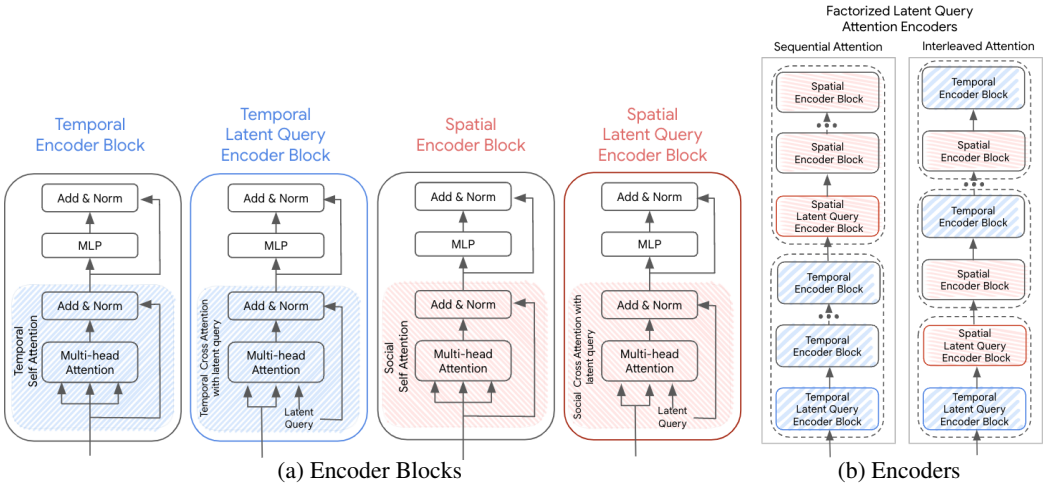
[49] Y. Tay, M. Dehghani, S. Abnar, Y. Shen, D. Bahri, P. Pham, J. Rao, L. Yang, S. Ruder, and D. Metzler. Long range arena: A benchmark for efficient transformers. *ArXiv*, abs/2011.04006, 2021.

[50] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler. Efficient transformers: A survey. *ArXiv*, abs/2009.06732, 2020.

Appendix

A Factorized Latent Query Attention

Figure 7a shows the implementation of Factorized latent query attention encoder blocks and Figure 7b shows how they are used in constructing the encoders. Specifically in factorized attention (sequential or interleaved), the first temporal encoder block and the first spatial encoder blocks in Figure 3 are replaced with temporal latent query encoder block and spatial latent query encoder block respectively.



(a) Encoder Blocks (b) Encoders

Figure 7: A summary of encoder architectures considered for Wayformer. (a) provides an overview of different encoder blocks and (b) explains how these blocks are arranged to construct the encoder.

B Hyperparameters

Hyperparameter	Values
Hidden size	{128, 256, 512}
Intermediate size	{2x, 4x} hidden size
Num encoder layers	[2, 16]
Num decoder layers	[2, 16]
Latent query ratio	{0.25, 0.5, 0.75 1.0}
Number GMM modes	64
Optimizer	AdamW
Initial learning rate	2e-4
Training steps	1000000
Learning rate decay	linear
Batch size	256

Table 2: Model and training hyperparameters across all ablation experiments done on WOMD.

Hyperparameter	WOMD	Argoverse
Max num history timesteps (including current timestep)	11	20
Max num roadgraph feats	512	1024
Max num context agents	64	64
Max num traffic lights	32	32

Table 3: Hyperparameters for generating WOMD and Arogverse input features. Fixed for all experiments

C Trajectory Aggregation Details

Given a distance threshold D, the trajectory aggregation scheme attempts to first select the fewest centroid modes such that all output modes are within a final distance D away from the nearest centroid. The aggregation algorithm iteratively selects centroid modes by greedily selecting the output mode that covers the maximum total likelihood out of the uncovered modes, and proceeds until all output modes have been covered.

After initializing these k centroid modes, the aggregation algorithm then proceeds into a refinement stage and runs another iterative procedure similar to k-means clustering starting from the initial centroid modes. In each iteration, each centroid mode becomes of the weighted average of all output modes assigned to it, and then output modes are reassigned to the new closest centroid mode.

D SOTA Wayformer Details

We describe the the hyperparameters used for WOMD and Argoverse benchmark results in Tables 4 and 5 respectively.

Hyperparameter	**Multi-axis Latent Query**	**Factorized Latent Query**
Hidden size	256	256
Intermediate size	1024	1024
Num encoder layers	2	4
Num decoder layers	8	4
Latent queries	192	4 time latents, 192 spatial latents
Number GMM modes	64	64
Ensemble size	3	3
Optimizer	AdamW	AdamW
Initial learning rate	2e-4	2e-4
Learning rate decay	linear	linear
Training steps	1200000	1000000
Batch size	256	256
Aggregation initial distance threshold	2.3	2.3
Aggregation refinement iterations	3	3
Aggregation max num trajectories	6	6

Table 4: Model and training hyperparameters for benchmark experiments on Waymo Open Motion 2021 Dataset

E Metrics

We compare models using competition specific metrics associated with these datasets. For all metrics, we consider only the top $k = 6$ most likely modes output by our model (after trajectory aggregation) and use only the mean of each mode.

Specifically, we report the following metrics taken from the evaluation procedure used in the standard evaluations based on the dataset being used.

$\mathbf{minDE^t_k}$ (Minimum Distance Error): Considers the top-k most likely trajectories output by the model, and computes the minimum distance to the ground truth trajectory at timestep t.

$\mathbf{MR^t}$ (Miss Rate): For each predicted trajectory, we compute whether it is sufficiently close to the predicted agent's ground truth trajectory at time t. Miss rate as the proportion of predicted agents for which none of the predicted trajectories are sufficiently close to the ground truth. We defer details of how a trajectory is determined to be sufficiently close to the WOMD metrics definition [27].

$\mathbf{minADE_k}$ (Minimum Average Distance Error): Similar to $minDE^t_k$, but the distance is calculated as an average over all timesteps.

$\mathbf{mAP}$: For each set of predicted trajectories, we have at most one positive - the one closest to the ground truth and which is within τ distance from the ground truth. The other predicted trajectories are reported as misses. From this, we can compute precision and recall at various thresholds. Following WOMD metrics definition [27] the agents future trajectories are partitioned into behavior

Hyperparameter	Multi-axis Latent Query	Factorized Latent Query
Encoder Hidden size	128	256
Encoder Intermediate size	512	1536
Decoder Hidden size	128	128
Decoder Intermediate size	512	768
Num encoder layers	4	4
Num decoder layers	6	6
Latent queries	1024	6 time latents, 192 spatial latents
Number GMM modes	6	6
Ensemble size	10	10
Optimizer	AdamW	AdamW
Initial learning rate	2e-4	2e-4
Learning rate decay	linear	linear
Training steps	1000000	1000000
Batch size	4	4
Aggregation initial distance threshold	2.9	2.9
Aggregation refinement iterations	5	5
Aggregation max num trajectories	6	6

Table 5: Model and training hyperparameters for benchmark experiments on Argoverse 2021 Dataset.

buckets, and an area under the precision-recall curve is computed using the possible true positive and false positives per agent, giving us Average Precision per behavior bucket. The total mAP value is a mean over the AP's for each behavior bucket.

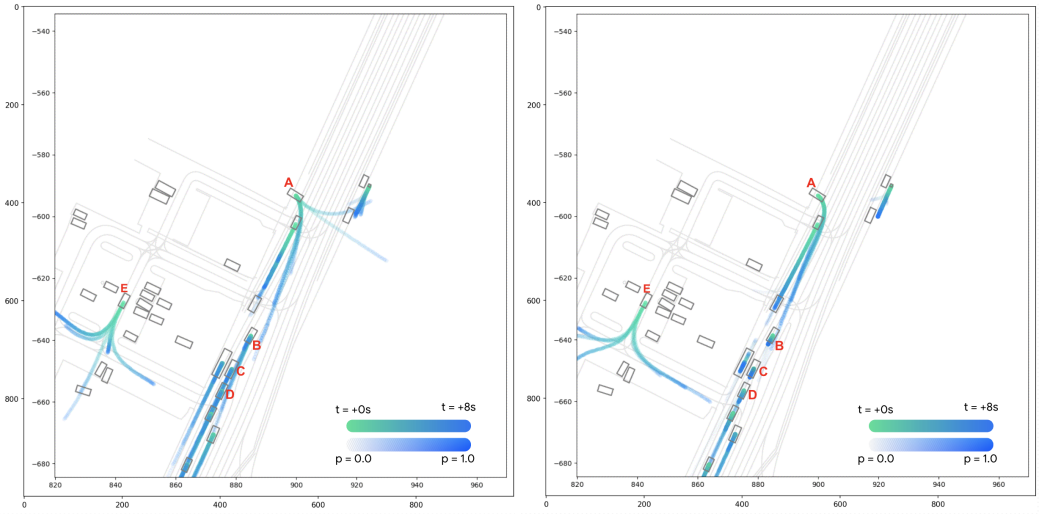
$\mathbf{Overlap^t}$: The fraction of timesteps of the most likely trajectory prediction for which the prediction overlaps with the corresponding timestep real future trajectory of another agent.

minFDE (Minimum Final Displacement Error): The L2 distance between the endpoint of the best forecasted trajectory and the ground truth.

$\mathbf{brier-minFDE}$: is defined as the sum of minFDE and the brier score $(1-p)^2$, where p is the probability of the best-predicted trajectory.

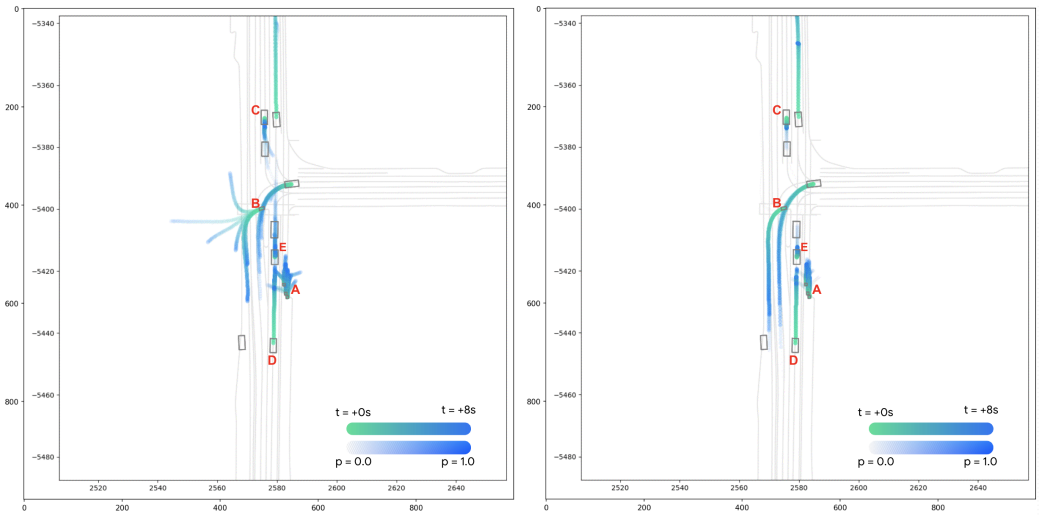
F Qualitative Wins

In this section, we present some examples of Wayformer (WF) predictions on WOMD scenes in comparison with MultiPath++ (MP++) model [3]. In all the following examples, (a) Hue indicates time horizon (0s - 8s), while transparency indicates probability. (b) Rectangles indicate vehicles, and squares indicate pedestrians or cyclists.



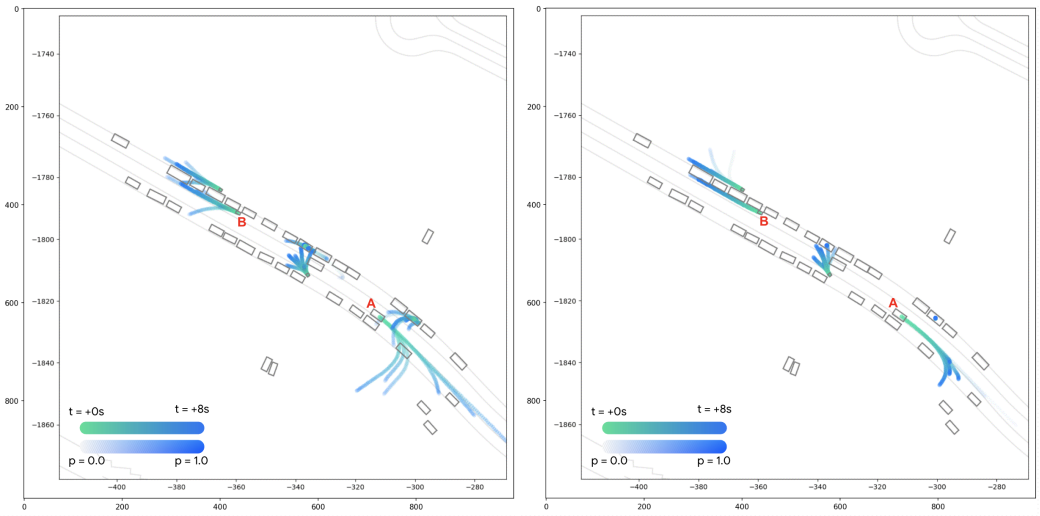
(a) MultiPath++ (MP++) (b) Wayformer (WF)

Figure 8: This scenario represents a multi-lane road with a parking lot on the left side. Here, we see that WF's performance on several vehicles is more safe and road following than that of MP++. For example: (a) Vehicle A is seen merging onto the road coming out of a parking lot. MP++'s predictions are completely off-road while WF's predictions follow rules of the road. (b) Vehicles B, C, and D's predictions overlap with each other for MP++ predicting collision with each other. But, WF correctly predicts that D yields for the vehicle before, C yields for D and B yields for C. (c) MP++'s predictions for vehicle E navigating the parking lot go through an already parked vehicles, while WF understands the interactions better and produces predictions which are not colliding.



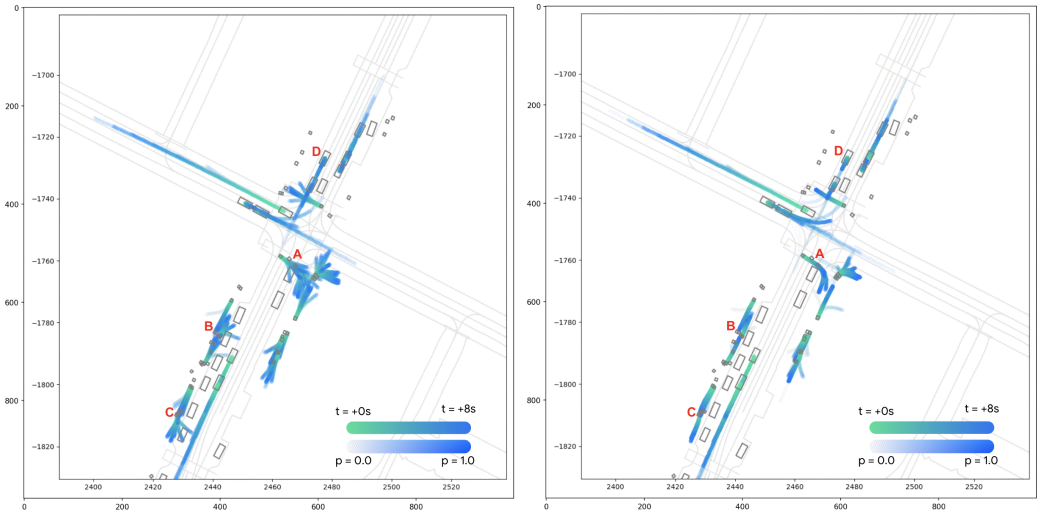
(a) MultiPath++ (MP++) (b) Wayformer (WF)

Figure 9: This scenario represents a T intersection. Here we see (a) a cyclist B, making a left turn. MP++'s predictions are off-road and going beyond the available road. But, WF's predictions follow rules of the road and present multiple speed profiles for the same action of taking a left turn. (b) We also see better predictions for a pedestrian (pedestrian A) where MP++ predicts that the pedestrian is going to walk onto the road with oncoming traffic. But, WF's predictions are constrained to the side walk. (c) In addition, we also notice that WF's predicts safe futures for vehicles C, D and E in comparison with MP++.



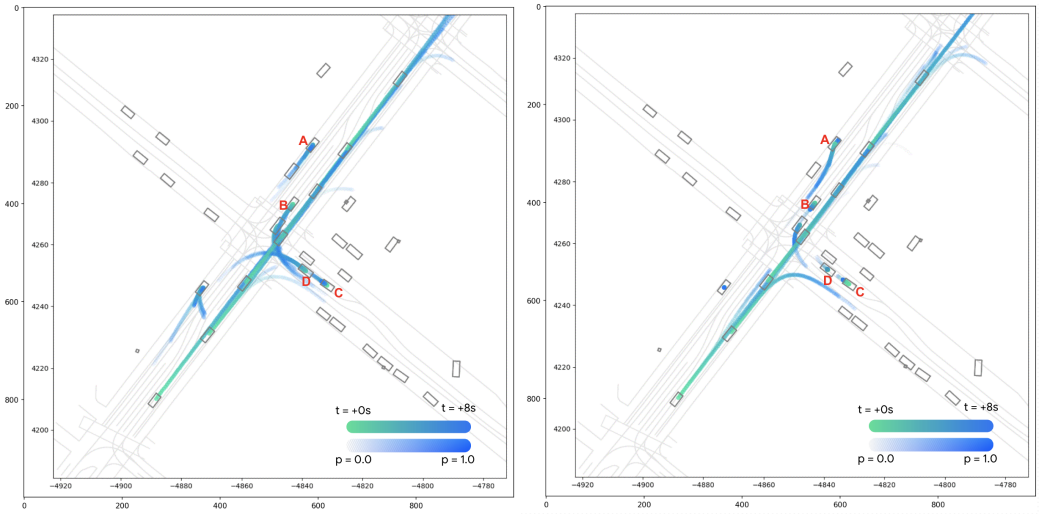
(a) MultiPath++ (MP++) (b) Wayformer (WF)

Figure 10: This scenario represents a vehicle (agent A) turning into a parking structure. MP++'s prediction discounts the presence of other parked vehicles and some predictions are made through the parked agents. WF models these interactions better and only predicts trajectories that do not collide with other parked entities.



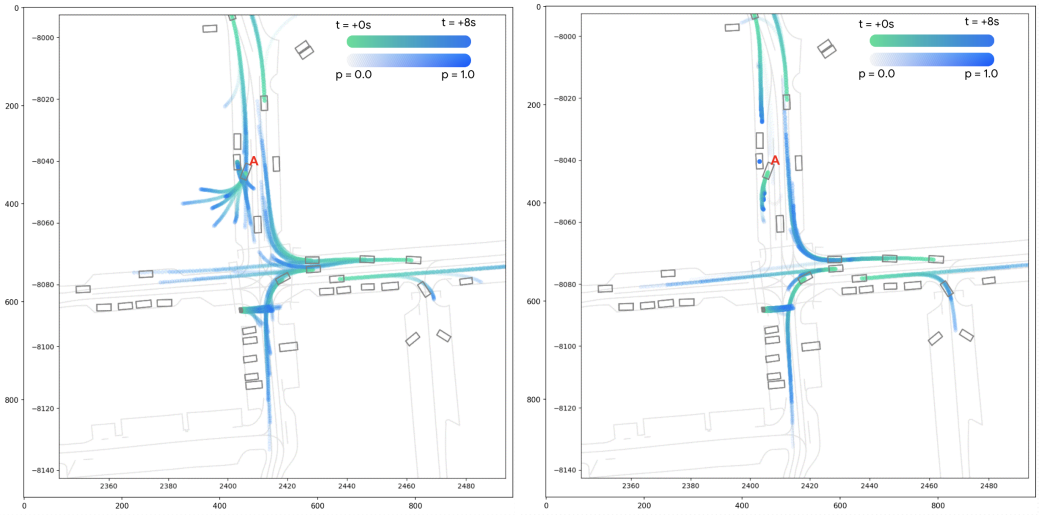
(a) MultiPath++ (MP++) (b) Wayformer (WF)

Figure 11: This scenario represents a busy 4-way intersection. First we discuss the WF improvements for pedestrian trajectory predictions. MP++ predicts pedestrian (A) as going into the oncoming vehicle demonstrating it fails to model this spatial interaction. WF demonstrates how the same pedestrian crosses in-front of this stopped vehicle and continues to walk on the corsswalk on the opposite side of the road. Pedestrian (agent B and C) on the lower left corner of the image show similar behavior. MP++ predicts them to bump into cars parked right next to them and walk onto the road surface towards oncoming traffic. WF on the other hand predicts nice and consistent along road trajectories for these pedestrians. We now observe the predicts for a vehicle (agent D) in this scene. MP++ predicts the trajectories of this vehicle to collide both with the static car in-front of it as well as the pedestrian passing in-front on that car. WF models all these spatial interactions well and predicts the trajectories for these car to wait behind the car in-front of it and not nudge into the pedestrian crossing in-front.



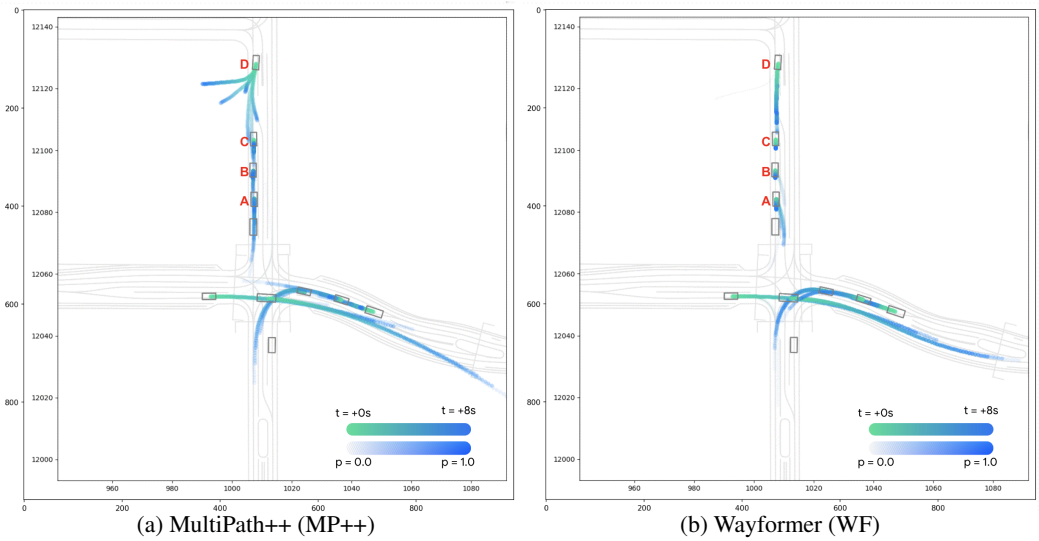
(a) MultiPath++ (MP++) (b) Wayformer (WF)

Figure 12: This scenario represents a comlpex 4-way intersection with lots of cars passing through. Similar to Fig- 11 we see MP++ predicting trajectories for vehicles (agent A, B, C an d D) in the scene to collide with cars in-front of them. WF demonstrates very sophisticated behavior. For agent A, it is able to estimate that the car parked in-front of agent A is a double-parked vehicle and there is space on the road next to it, so it predicts trajectories that nudge around it. For B, C and D it is able to carefully model the rules of the road and allow either oncoming (in case of agent B) or cross traffic (in case of agent C and D) to take precedence and predicts yielding trajectories for them.



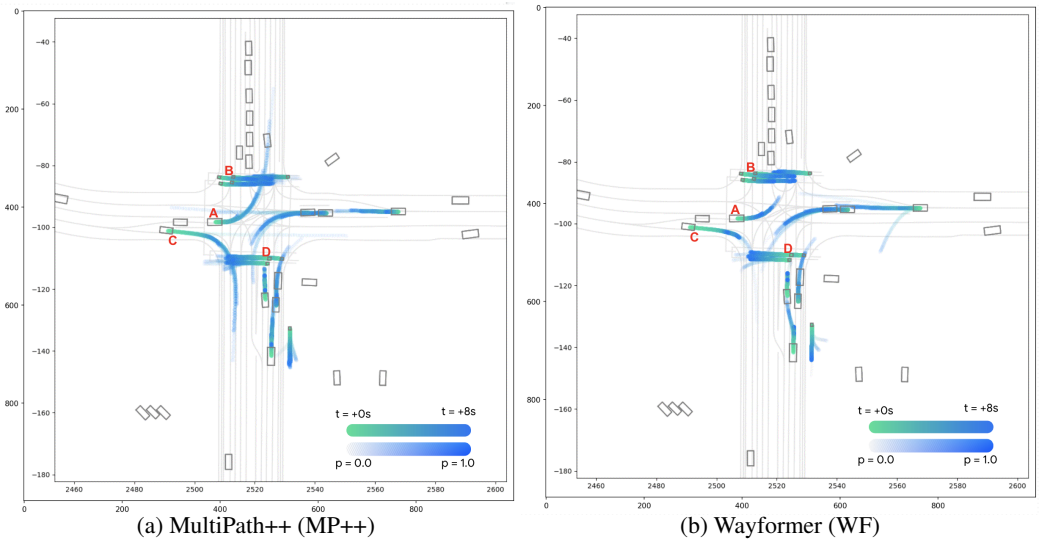
(a) MultiPath++ (MP++) (b) Wayformer (WF)

Figure 13: In this scenario we observe that MP++ is not able to model the future of the vehicle (agent A) entering the parking lane and outputs a multi-modal equally likely future for this agent. WF understands the roadgraph interaction much better and outputs trajectories that have high likelihood that agent A is entering the parking lane.



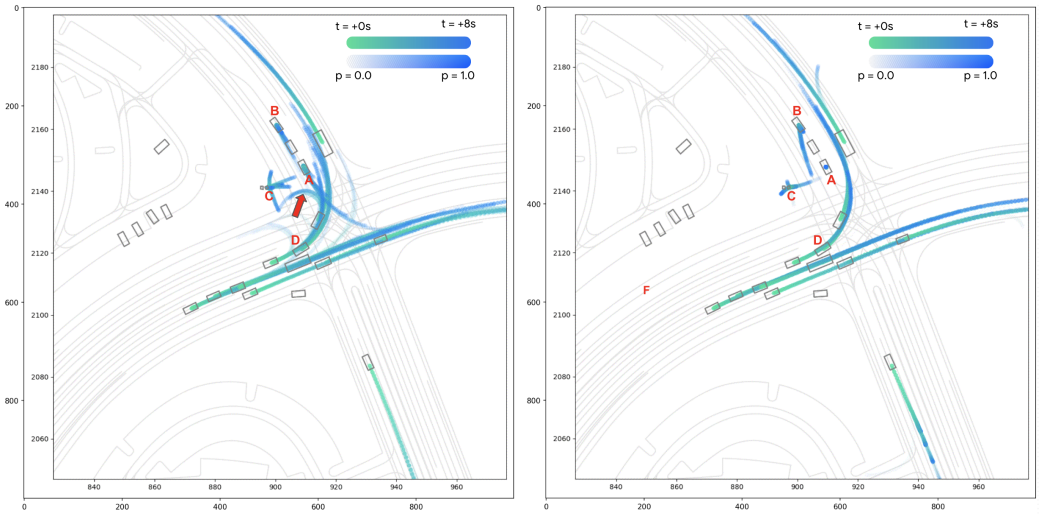
(a) MultiPath++ (MP++) (b) Wayformer (WF)

Figure 14: We see agents A, B, and C are waiting behind a stationary vehicle. WF predicts agent A will nudge around the stationary vehicle to make progress, while MP predicts the agents will proceed through the stationary vehicle. Additionally, MP predicts agent D could proceed off the road, while WF predicts it to follow the road behind agent C.



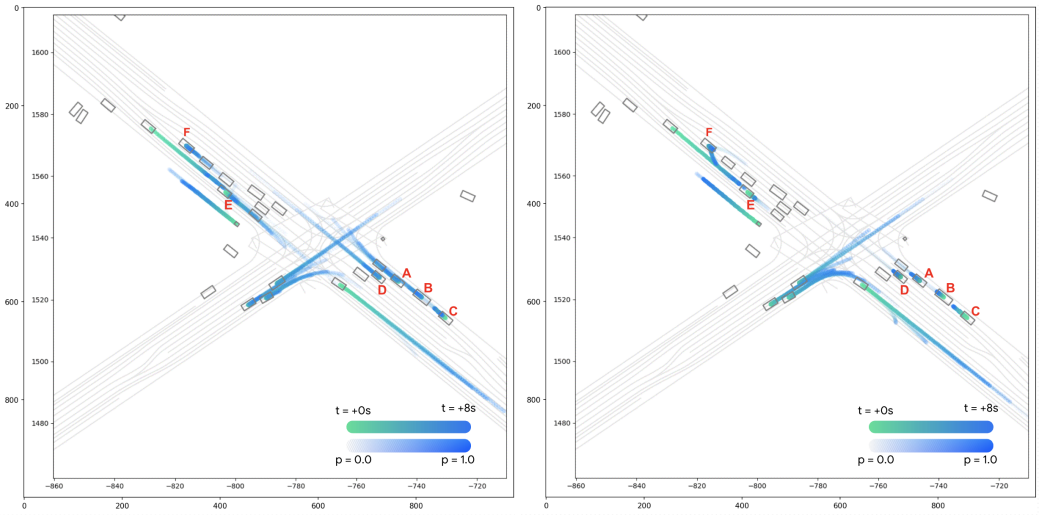
(a) MultiPath++ (MP++) (b) Wayformer (WF)

Figure 15: Multiple pedestrians, including agent B, are crossing the road and both MP++ and WF predict car A wants to make a left turn through that crosswalk. WF predicts car A will start to turn, then wait as the pedestrians cross, while MP++ predicts that car A will proceed through the crosswalk even as the pedestrians are crossing.



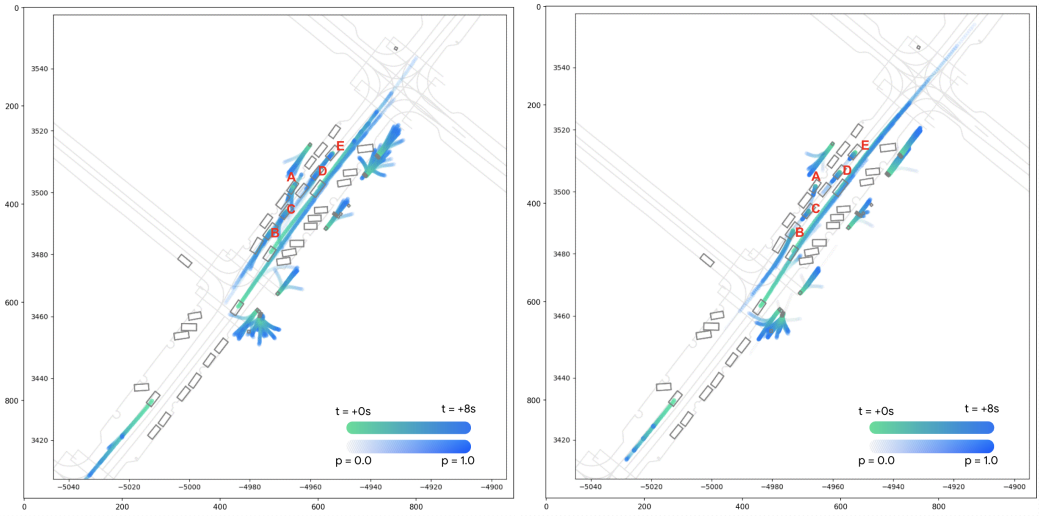
(a) MultiPath++ (MP++) (b) Wayformer (WF)

Figure 16: This shows a busy intersection, with both WF and MP++ predicting vehicles in the left-right road (i.e. agent D) are either proceeding straight or left turning. However, MP++ predicts agent A to try to make a left turn directly into the flow of traffic, including through other cars left turning, while WF predicts agent A will wait. Additionally, MP++ predicts agent B will try to proceed through the vehicle waiting in front of it, while WF instead predicts it either remaining stationary or nudging to the adjacent lane. Furthermore, WF also predicts agent D to potentially make a U-turn that goes through the corner of the sidewalk near agent C (highlighted by the red arrow).



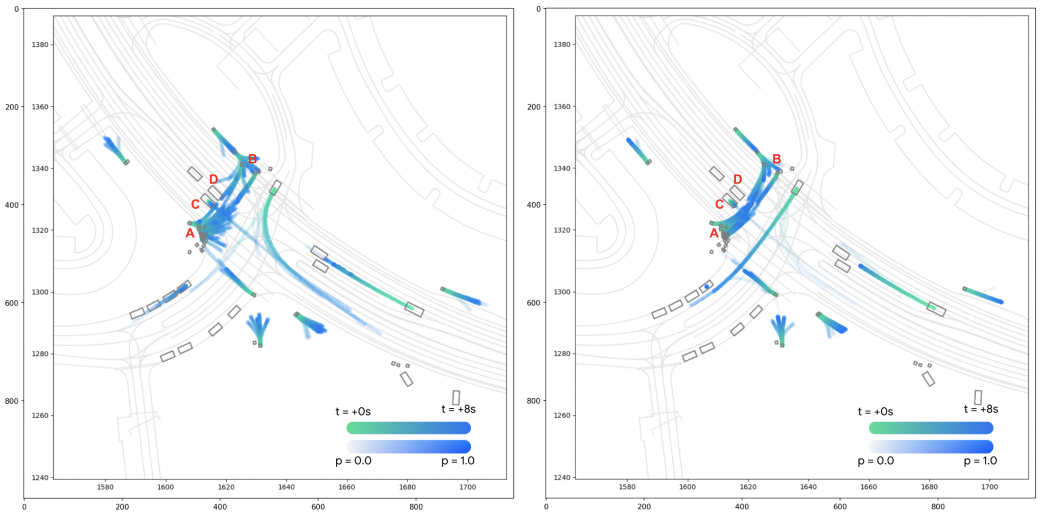
(a) MultiPath++ (MP++) (b) Wayformer (WF)

Figure 17: This scenario represents a 4-way intersection. (a) At the intersection vehicles A, B, C and D are all stopped at the intersection due to signal. WF takes this into account and predicts yielding behavior. Vehicle D yielding for the light, Vehicle C yielding for B, Vehicle B yielding for A and Vehicle A yielding for the vehicle in-front. But, MP++'s predictions for the same agents go through the intersection (Vehicle D) and for vehicles A, B and C, they pass through the vehicles in-front. (b) We see similar behaivor on the other side of the intersection, where vehicle E's WF predictions are yielding and MP++ predictions are passing through vehicles in the front.



(a) MultiPath++ (MP++) (b) Wayformer (WF)

Figure 18: This scenario represents a T intersection with narrow roads and parked cars. In this highly interactive scene, we observe that (a) a parked vehicle (vehicle A) is trying to merge into traffic. WF predicts nudging around already parked cars and merging onto the traffic, while MP++ predictions pass through the parked cars in front of A. (b) In addition, we also see that for vehicle B WF predicts that nudges around the vehicle in front while MP++ predictions go through the car in-front. (c) For vehicles C, D and E, WF predicts yielding behavior (C yielding for B, D yielding for car in the front and E yielding for D), while MP++ predictions go through the vehicles in front.



(a) MultiPath++ (MP++) (b) Wayformer (WF)

Figure 19: This scenario represents a very busy 4-way intersection with clusters of pedestrians (A, B). Both these clusters are pedestrians crossing the signal from either side of the road. We observe that MP++ prediction's are more distributed, some of them going through already stopped vehicles (vehicles C and D) at the intersection. But, WF understands the presence of other vehicles and produces predictions which do not cross through them. We also see that WF's predictions for vehicle C yield to pedestrians while MP++'s predictions do not.