Query-Centric Trajectory Prediction

Zikang Zhou\textsuperscript{1,2} Jianping Wang\textsuperscript{1,2} Yung-Hui Li\textsuperscript{3} Yu-Kai Huang\textsuperscript{4}
\textsuperscript{1}City University of Hong Kong \textsuperscript{2}City University of Hong Kong Shenzhen Research Institute
\textsuperscript{3}Hon Hai Research Institute \textsuperscript{4}Carnegie Mellon University
zikanzhou2-c@my.cityu.edu.hk

Abstract

Predicting the future trajectories of surrounding agents is essential for autonomous vehicles to operate safely. This paper presents QCNet, a modeling framework toward pushing the boundaries of trajectory prediction. First, we identify that the agent-centric modeling scheme used by existing approaches requires re-normalizing and re-encoding the input whenever the observation window slides forward, leading to redundant computations during online prediction. To overcome this limitation and achieve faster inference, we introduce a query-centric paradigm for scene encoding, which enables the reuse of past computations by learning representations independent of the global spacetime coordinate system. Sharing the invariant scene features among all target agents further allows the parallelism of multi-agent trajectory decoding. Second, even given rich encodings of the scene, existing decoding strategies struggle to capture the multimodality inherent in agents’ future behavior, especially when the prediction horizon is long. To tackle this challenge, we first employ anchor-free queries to generate trajectory proposals in a recurrent fashion, which allows the model to utilize different scene contexts when decoding waypoints at different horizons. A refinement module then takes the trajectory proposals as anchors and leverages anchor-based queries to refine the trajectories further. By supplying adaptive and high-quality anchors to the refinement module, our query-based decoder can better deal with the multimodality in the output of trajectory prediction. Our approach ranks 1\textsuperscript{st} on Argoverse 1 and Argoverse 2 motion forecasting benchmarks, outperforming all methods on all main metrics by a large margin. Meanwhile, our model can achieve streaming scene encoding and parallel multi-agent decoding thanks to the query-centric design ethos.

1. Introduction

Making safe decisions for autonomous vehicles requires accurate predictions of surrounding agents’ future trajectories. In recent years, learning-based methods have been widely used for trajectory prediction [14, 31, 37, 38, 46, 56]. Despite the considerable efforts made to enhance models’ forecasting ability, there is still a long way to go before fully addressing the problem of trajectory prediction. Why is this task so challenging, and what inability lies in existing approaches? We attempt to answer these questions from the following two perspectives:

(i) While the flourishing forecasting models have achieved impressive performance on trajectory prediction benchmarks [7, 13, 49], today’s most advanced architectures specialized for this task [37, 38, 46, 56] fail to process the heterogeneous traffic scenes efficiently. In an autonomous driving system, data frames arrive at the prediction module sequentially as a stream of sparse scene context, including the high-definition vector map and the surrounding agents’ kinematic states. A model must learn expressive representations of these scene elements to achieve accurate forecasts. With the continuing development of modeling techniques for sparse context encoding [14, 31, 50], the research community has witnessed rapid progress toward more powerful trajectory predictors. Notably, factorized attention-based Transformers [37, 38, 56] have recently raised prediction accuracy to an unprecedented level. However, they require learning attention-based representations for each spatial-
temporal scene element and suffer from prohibitively high costs when processing dense traffic scenes. As every minimal delay may lead to catastrophic accidents in autonomous driving, the unmet need for real-time predictions has limited the applicability of state-of-the-art approaches.

(ii) The immense uncertainty in the output of trajectory prediction, which grows explosively as the prediction horizon lengthens, has troubled the research community constantly. For example, a vehicle at an intersection may turn or go straight depending on the driver’s long-term goal. To avoid missing any potential behavior, a model must learn to capture the underlying multimodal distribution rather than simply predicting the most frequent mode. This learning task is challenging since only one possibility is logged in each training sample. To ease the learning difficulty, a body of works utilizes handcrafted anchors as guidance for multimodal prediction [6, 12, 39, 53, 55]. Their effectiveness, however, is subject to the quality of the anchors. Typically, these methods fail to work well when few anchors can precisely cover the ground truth. This problem is exacerbated in long-term prediction, where the search space for anchors is much larger. Some other works [10, 31, 38, 46, 56] circumvent this issue by directly predicting multiple trajectories, albeit at the risk of mode collapse and training instability [33, 41]. Due to the lack of spatial priors, these methods also fail to produce accurate long-term forecasts.

The analysis above drives us to propose a trajectory prediction framework, termed as QCNet, to overcome the limitations of previous solutions. First, we note that it is possible to achieve faster online inference while also benefiting from the power of factorized attention, but the agent-centric encoding scheme [25, 27, 46, 56] circumvents this issue by directly predicting multiple trajectories, albeit at the risk of mode collapse and training instability [33, 41]. Due to the lack of spatial priors, these methods also fail to produce accurate long-term forecasts.

The invariant scene features can be shared among all target agents in the scene to enable the parallelism of multi-agent decoding. Query-centric paradigms for scene encoding (see Fig. 1). The crux of our design ethos lies in processing all scene elements in their local spacetime reference frames and learning representations independent of the global coordinates. This strategy enables us to cache and reuse the previously computed encodings, spreading the computation across all observation windows and thereby reducing inference latency. The invariant scene features can also be shared among all target agents in the scene to enable the parallelism of multi-agent decoding. Second, to better utilize the scene encodings for multimodal and long-term prediction, we use anchor-free queries to retrieve the scene context recurrently and let them decode a short segment of future waypoints at each recurrence. This recurrent mechanism eases the modeling burden on the queries by allowing them to focus on different scene contexts when predicting waypoints at different horizons. The high-quality trajectories predicted by the recurrent decoder serve as dynamic anchors in the subsequent refinement module, where we use anchor-based queries to refine the trajectory proposals based on the scene context. As a result, our query-based decoding pipeline incorporates the flexibility of anchor-free methods into anchor-based solutions, taking the best of both worlds to facilitate multimodal and long-term prediction.

Our proposed query-centric encoding paradigm is the first that can exploit the sequential nature of trajectory prediction to achieve fast online inference. Besides, our query-based decoder exhibits superior performance for multimodal and long-term prediction. Experiments show that our approach achieves state-of-the-art results, ranking 1st on two large-scale motion forecasting benchmarks [7, 49].

2. Related Work

Scene context fusion encodes rich information for trajectory prediction. Early work rasterizes world states as multi-channel images and employs classic convolutional neural networks for learning [5, 6, 10, 21]. Due to the lossy rendering, limited receptive field, and prohibitively high cost of raster-based methods, the research community has turned to a vector-based encoding scheme [14, 31, 50]. With the use of permutation-invariant set operators such as pooling [3, 12, 14, 20, 46], graph convolution [11, 31, 36, 53], and attention mechanism [24, 26, 30, 32, 34, 52], vector-based methods can efficiently aggregate sparse information in traffic scenes. Several powerful trajectory prediction models have recently adopted Transformers [47] with factorized attention as their encoders [18, 37, 38, 56]. Although these models improve efficiency by learning agent-centric representations hierarchically [56] or encoding the whole scene in a shared coordinate system [38], their scalability is still limited by the computational complexity of factorized attention. In comparison, our encoder inherits the representational power of factorized attention while achieving more efficient scene context fusion by using a query-centric encoding paradigm, which goes beyond agent-centric modeling and enables streaming trajectory prediction.

Multimodal future distribution is a widely adopted output form of trajectory prediction, given that world states are partially observable and agents’ intentions are highly uncertain. While generative models naturally fit multimodal prediction [20, 28, 40, 45], sampling from latent variables introduces test-time stochasticity, which is undesirable for safety-critical applications such as autonomous driving. Another line of research tackles multimodality by decoding a discrete set of trajectories from the encoded scene context [6, 10, 31, 55]. Since only one mode is ob-
served in training data, predicting multiple diverse futures is challenging. Anchor-based methods achieve this with the guidance of anchors, which facilitate multimodal prediction by leveraging predefined maneuvers [12], candidate trajectories [6, 39], or map-adaptive goals [53, 55]. However, the quality of these anchors significantly impacts prediction performance. By contrast, anchor-free methods output multiple hypotheses freely at the risk of mode collapse and training instability [10, 31, 38, 46]. Our decoding pipeline takes advantage of both anchor-based and anchor-free solutions, with an anchor-free module generating adaptive anchors in a data-driven manner and an anchor-based module refining these anchors based on the scene context.

3. Approach

3.1. Input and Output Formulation

Consider a scenario with \( A \) agents surrounding the autonomous vehicle. During online running, the perception module supplies a stream of agent states to the prediction module at a fixed interval, where each agent state is associated with its spatial-temporal position and geometric attributes. For example, the \( i \)-th agent’s state at time step \( t \) comprises the spatial position \( p_i^t = (p_{i,x}^t, p_{i,y}^t) \), the angular position \( \theta_i^t \) (i.e., the yaw angle), the temporal position \( t \) (i.e., the time step), and the velocity \( v_i^t \). We also add the motion vector \( p_i^t - p_i^{t-1} \) to the geometric attributes similar to some baselines [31, 56]. Besides, the prediction module has access to \( M \) polygons on the high-definition map (e.g., lanes and crosswalks), where each map polygon is annotated with sampled points and semantic attributes (e.g., the user type of a lane). Given the map information and the agent states within an observation window of \( T \) time steps, the prediction module is tasked with forecasting \( K \) future trajectories for each target agent over a horizon of \( T' \) time steps and assigning a probability score for each forecast.

3.2. Query-Centric Scene Context Encoding

The first step of trajectory prediction is to encode the scene input. Recent research has found factorized attention incredibly effective for scene encoding [37, 38, 56]. These approaches let a query element attend to key/value elements along one axis at a time, which results in temporal attention, agent-map attention, and social attention (i.e., agent-agent attention) with the complexity of \( \mathcal{O}(AT^2) \), \( \mathcal{O}(ATM) \), and \( \mathcal{O}(A^2T) \), respectively. Unlike typical encoding strategies that first apply a temporal network to squeeze the time dimension and then perform agent-map and agent-agent fusions at the current time step only, factorized attention conducts fusions at every past time step within the observation window. As a result, factorized attention can capture more information, such as how the relations between agents and map elements evolve over the observation horizon. However, its scalability is limited by the cubic complexity of each fusion operation. In extreme circumstances involving hundreds of agents and map elements, such models may fail to emit predictions promptly. We ask: *is it possible to reduce the inference latency during online prediction while enjoying the representational power of factorized attention?*

Before diving into our solution, recall that trajectory prediction is a streaming processing task: when a new data frame arrives, we put it in the queue and drop the oldest one. Thus, the latest observation window has \( T - 1 \) time steps overlapping with its predecessor. This fact motivates us to raise another question: *can we reuse the overlapped time steps’ encodings computed previously after the observation window slides forward?* Unfortunately, this idea is infeasible owing to the normalization requirement for trajectory prediction: existing methods employ an agent-centric encoding paradigm for spatially roto-translation invariance [25, 27, 46, 56], where each agent is encoded in the local coordinate frame determined by its current time step’s position and yaw angle. Each time the observation window slides forward, the “current time step” also shifts accordingly, and the geometric attributes of all scene elements need to be re-normalized based on the positions of the latest agent states. Due to the variation in input, we are forced to re-encode all time steps’ elements even though the observation windows largely overlap.

Based on the analysis above, we identify that the evolving spacetime coordinate systems hinder the reuse of previously computed encodings. To address this issue, we introduce a query-centric encoding paradigm for learning representations independent of scene elements’ global coordinates. Specifically, we establish a local spacetime coordinate system for each scene element that a query vector derives from, processing query elements’ features in their local reference frames. Then, we inject the relative spatial-temporal positions into the key and value elements when performing attention-based scene context fusion. We elaborate on the encoding process in the following paragraphs.

**Local Spacetime Coordinate System.** Figure 1 shows an example of scene elements’ local coordinate systems. For the \( i \)-th agent’s state at time step \( t \), the local coordinate frame is determined by the reference spatial-temporal position \( (p_i^t, t) \) and the reference direction \( \theta_i^t \), where \( p_i^t \) and \( \theta_i^t \) are the agent state’s spatial and angular positions, respectively. For lanes and crosswalks, we choose the position and orientation at the entry point of the centerline as the reference. In this way, we build local coordinate systems canonically for all the scene elements considered, resulting in one dedicated local frame per map polygon and \( T \) reference frames per agent within any observation window.

**Scene Element Embedding.** For each spatial-temporal scene element, such as an agent state or a lane, we compute the polar coordinates of all geometric attributes (e.g.,
the velocity and motion vector of an agent state, the positions of all sampled points on a lane) relative to the spatial point and direction referenced by the element’s local frame. Then, we transform each polar coordinate into Fourier features [22, 35, 44] to facilitate learning high-frequency signals. For each agent state and each sampled point on the map, the Fourier features are concatenated with the signals. For each agent state and each sampled point on the lane, we transform each polar coordinate into Fourier features of all sampled points on a lane) relative to the spatial frame. For an element with absolute spatial-temporal position \( (p_j^s, \theta_j^s, t) \) and another with \( (p_j'^s, \theta_j'^s, s) \), we use a 4D descriptor to summarize their relative position, whose components are the relative distance \( \| p_j^s - p_j'^s \|_2 \), the relative direction \( \tan(2(p_j^s - p_j'^s, p_j^s, x - p_j^s, x) - \theta_j^s, \theta_j'^s) \), and the time gap \( s - t \). Since we can easily reconstruct one element’s absolute position from another with the help of the descriptor, we have preserved all spatial-temporal position information of the scene element pair. Then, we transform the 4D descriptor into Fourier features and pass them through an MLP to produce the relative positional embedding \( r_{j\rightarrow j'} \). If any of the two scene elements are static (e.g., static map polygons), we can omit the superscript and denote the embedding as \( r_{j\rightarrow j'} \).

**Self-Attention for Map Encoding.** We employ self-attention to model the relationships among map elements, after which the updated map encodings will enrich the agent features and assist trajectory decoding. For the \( i \)-th map polygon, we derive a query vector from its embedding \( \mathbf{m}_i \) and let it attend to the neighboring lanes and crosswalks \( \{ \mathbf{m}_j \}_j \in \mathcal{N}_i \), where \( \mathcal{N}_i \) denotes the neighbor set of the polygon. To incorporate spatial awareness for map encoding, we generate the \( j \)-th key/value vector from the concatenation of \( \mathbf{m}_j \) and the relative positional embedding, i.e., \( [\mathbf{m}_j; r_{j\rightarrow j'}] \). Since each triple of \( (\mathbf{m}_i, \mathbf{m}_j, r_{j\rightarrow j'}) \) input to the attention layer is independent of the global spacetime coordinate system, the output map encodings \( \{ \mathbf{m}_i' \}_{i=1}^M \) are also invariant under transformations of the global reference frame. Thus, they can be shared across all agents and all time steps and can even be pre-computed offline, thereby avoiding redundant computation suffered by agent-centric modeling.

**Factorized Attention for Agent Encoding.** To help the agent embeddings capture more information, we also consider factorized attention across agent time steps, among agents, and between agents and maps. Take the \( i \)-th agent at time step \( t \) as an example. Given the query vector derived from the agent state’s embedding \( \mathbf{a}_i^t \), we employ temporal attention by computing the key and value vectors based on \( \{ [\mathbf{a}_i^t; r_{i\rightarrow j}^t] \}_{j=1}^N, \) which are the \( i \)-th agent’s embeddings from time step \( t - \tau \) to time step \( t - 1 \) and the corresponding relative positional embeddings. Likewise, the key and value vectors for agent-map and social attention are derived from \( \{ [\mathbf{m}_j; r_{j\rightarrow i}] \}_{j} \) and \( \{ [\mathbf{a}_i^t; r_{i\rightarrow j}^t] \}_{j} \), respectively, where the neighbor set \( \mathcal{N}_i \) is determined by a distance threshold of 50 meters. As a result of updating the initially invariant queries with invariant keys and values, the outputs of these layers are also invariant. We stack the temporal, the agent-map, and the social attention sequentially as one fusion block and repeat such blocks \( L_{enc} \) times.

Thanks to the query-centric modeling, all the agent and map encodings are unique and fixed no matter from which spacetime coordinate system we view them (i.e., rotation-invariance for the space dimension and translation-invariance for the time dimension), enabling the model to reuse past computations and operate streamingly. During online prediction, we can cache the encodings computed in previous observation windows and incrementally update the scene representation. As shown in Fig. 2, our model only performs factorized attention for the \( A \) incoming agent states when a new data frame arrives, resulting in temporal attention with \( O(\mathcal{A}T^2) \) complexity, agent-map attention with \( O(\mathcal{A}M) \) complexity, and social attention with \( O(\mathcal{A}^2) \) complexity. All of these operations are an order less expensive than their non-streaming counterpart. Finally, we update the cached tensors using the newly computed encodings.
3.3. Query-Based Trajectory Decoding

The second step of trajectory prediction is to utilize the scene encodings output by the encoder to decode $K$ future trajectories for each target agent, which is non-trivial since the encoder returns only one set of feature embeddings. Inspired by the progress in object detection, some recent works [18, 32, 37, 46] employ DETR-like decoders [4] to deal with such a one-to-many problem, where multiple learnable queries cross-attend the scene encodings and decode trajectories. However, these models suffer from training instability and mode collapse like other anchor-free approaches. Moreover, they do not perform well in long-term prediction, where the forecasting task is much more challenging due to the explosive uncertainty in the distant future. Our query-based decoder overcomes these limitations by utilizing a recurrent, anchor-free proposal module to generate adaptive trajectory anchors, followed by an anchor-based module that further refines the initial proposals. An overview of our decoding pipeline is shown in Fig. 3. In the following, we will illustrate the components of the decoder in detail.

**Mode2Scene and Mode2Mode Attention.** Both the proposal and refinement modules use a DETR-like architecture. Similar to the concept of object queries in DETR [4], each query takes charge of decoding one of the $K$ trajectory modes. In the Mode2Scene attention, we use cross-attention layers to update the mode queries with multiple contexts, including the history encodings of the target agent, the map encodings, and the neighboring agents’ encodings. Following the Mode2Scene attention, the $K$ mode queries “talk” to each other via the Mode2Mode self-attention to improve the diversity of multiple modes.

**Reference Frames of Mode Queries.** To predict the trajectories of multiple agents in parallel, we share the same set of scene encodings among all target agents in the scene. As these encodings are derived from their local spacetime coordinate systems, we need to project them into each target agent’s current viewpoint to achieve the same effect as agent-centric modeling. To this end, we hallucinate a coordinate frame for each mode query based on the corresponding target agent’s current position and yaw angle. When updating the query embeddings via Mode2Scene attention, the scene elements’ positions relative to the queries are incorporated into the keys and values, which is similar to what we have done for the encoder.

**Anchor-Free Trajectory Proposal.** We use learnable, anchor-free queries to propose initial trajectories. These proposals will later act as anchors in the refinement module. Compared with anchor-based methods that attempt to cover the ground truth with densely sampled handcrafted anchors [6, 19], our proposal module generates $K$ adaptive anchors in a data-driven manner. Thanks to the cross-attention layers, the mode queries can retrieve the scene context and quickly narrow the search space for anchors. The self-attention layer further allows the queries to collaborate with each other when generating trajectory proposals.

Over an extended prediction horizon, an agent can travel a long distance, and its surrounding environment may vary quickly. As a result, it is hard to summarize all information required for decoding a long sequence into a single query embedding. To ease the queries’ burden of context extraction and improve the anchors’ quality, we generalize the DETR-like decoder to a **recurrent** fashion. Using $T_{rec}$ recurrent steps, the context-aware mode queries only decode $T'/T_{rec}$ future waypoints via an MLP at the end of each recurrent step. At the subsequent recurrence, these queries become the input again and extract the scene context relevant to the next few waypionts’ prediction. For efficiency, $T_{rec}$ is far smaller than the prediction horizon $T'$. We also find that using much more recurrent steps is unnecessary.

**Anchor-Based Trajectory Refinement.** Anchor-free decoding can be a two-edged sword: despite its flexibility, the unstable training process may lead to mode collapse occasionally. On the other hand, the randomly initialized mode queries must adapt to all target agents in all scenes and lack the scenario-specific bias, which may result in non-

---

Figure 3. Overview of the decoding pipeline. An anchor-free module generates trajectory proposals **recurrently** based on the encoded scene context. These proposals act as the anchors in the refinement module, where an anchor-based decoder refines the anchor trajectories and assigns a probability score for each hypothesis.
compliant predictions, such as trajectories that violate the laws of motion or break the traffic rules conveyed by the high-definition map. We are thus motivated to employ an anchor-based module to refine the proposals further. Taking the output of the proposal module as anchors, we let the refinement module predict the offset to the proposed trajectories and estimate the likelihood of each hypothesis. This module also adopts a DETR-like architecture, but its mode queries are derived from the proposed trajectory anchors instead of randomly initialized. Specifically, a small GRU [8] is used to embed each trajectory anchor, and we take its final hidden state as the mode query. These anchor-based queries provide explicit spatial prior for the model, enabling the attention layers to localize the context of interest more easily.

3.4. Training Objectives

Following HiVT [56], we parameterize the i-th agent’s future trajectory as a mixture of Laplace distributions:

$$f(\{p_t\}_{t=1}^{T}) = \sum_{k=1}^{K} \pi_{i,k} \prod_{t=1}^{T} \text{Laplace}(p_t^i | \mu_{i,k}^t, b_{i,k}^t),$$

(1)

where \{\pi_{i,k}\}_{k=1}^{K} are the mixing coefficients, and the k-th mixture component’s Laplace density at time step t is parameterized by the location \(\mu_{i,k}^t\) and the scale \(b_{i,k}^t\). We then use a classification loss \(L_{\text{cls}}\) to optimize the mixing coefficients predicted by the refinement module. This loss minimizes the negative log-likelihood of Eq. (1), and we stop the gradients of the locations and scales to optimize the mixing coefficients only. On the other hand, we adopt the winner-take-all strategy [29] to optimize the locations and scales output by the proposal and refinement modules, which conducts backpropagation on the best-predicted proposal and its refinement only. For stabilization, the refinement module stops the gradients of the proposed trajectory anchors.

The final loss function combines the trajectory proposal loss \(L_{\text{propose}}\), the trajectory refinement loss \(L_{\text{refine}}\), and the classification loss \(L_{\text{cls}}\) for end-to-end training:

$$L = L_{\text{propose}} + L_{\text{refine}} + \lambda L_{\text{cls}},$$

(2)

where we use \(\lambda\) to balance regression and classification.

4. Experiments

4.1. Experimental Settings

Datasets. We use Argoverse 1 [7] and Argoverse 2 [49], two large-scale motion forecasting datasets, to test the efficacy of our approach. The Argoverse 1 dataset collects 323,557 sequences of data from Miami and Pittsburgh, while the Argoverse 2 dataset contains 250,000 scenarios spanning six cities. Both datasets have a sampled rate of 10 Hz. For the Argoverse 1 dataset, models need to predict agents’ 3-second future trajectories given the 2-second observations of history. The Argoverse 2 dataset, in comparison, is featured by improved data diversity, higher data quality, a larger observation window of 5 seconds, and a longer prediction horizon of 6 seconds. Using these two datasets, we intend to examine models’ forecasting capability on various data distributions and prediction horizons.

Metrics. Following the standard evaluation protocol, we adopt metrics including minimum Average Displacement Error (minADE\(_K\)), minimum Final Displacement Error (minFDE\(_K\)), Brier-minimum Final Displacement Error (b-minFDE\(_K\)), and Miss Rate (MR\(_K\)) for evaluation. The metric minADE\(_K\) calculates the \(\ell_2\) distance in meters between the ground-truth trajectory and the best of \(K\) predicted trajectories as an average of all future time steps. On the other hand, the metric minFDE\(_K\) only concerns the prediction error at the final time step to emphasize long-term performance. To further measure the performance of uncertainty estimation, the metric b-minFDE\(_K\) adds \((1 - \hat{\pi})^2\) to the final-step error, where \(\hat{\pi}\) denotes the best-predicted trajectory’s probability score that the model assigns. Moreover, the metric MR\(_K\) is used for counting the ratio of cases where minFDE\(_K\) exceeds 2 meters. As a common practice, \(K\) is selected as 1 and 6. If a model outputs more than \(K\) trajectories, only the predictions with the top-\(K\) probability scores are considered during evaluation.

4.2. Comparison with State of the Art

We compare our method with the strongest baselines on the Argoverse 1 and the Argoverse 2 motion forecasting benchmarks [7, 49]. We first conduct experiments on the Argoverse 2 dataset [49], which favors solutions that work well on long-term prediction, given that its prediction horizon is as long as 6 seconds. The results are shown in Tab. 1. Even without ensembling, QCNet has already outperformed all previous approaches on the Argoverse 2 test set in terms of minADE\(_6\), minFDE\(_6\), minADE\(_1\), and minFDE\(_1\). After using ensembling techniques similar to other entries, QCNet surpasses all methods on all metrics by a large margin. We also evaluate our model on the Argoverse 1 dataset [7] to better understand the generalizability of our approach. Although the performance on the Argoverse 1 benchmark has saturated for years [49], Tab. 2 shows that QCNet significantly advances state-of-the-art on most metrics. As of the time we submitted the paper, QCNet ranks 1st on the leaderboards of Argoverse 1 and Argoverse 2, outperforming all published and unpublished works on the two benchmarks. Please refer to the supplementary material for more results on Argoverse 2 [49] and Waymo Open Motion Dataset [13].

4.3. Ablation Study

Effects of Scene Context Fusion. We study the effects of scene context fusion in Tab. 3. The first question we answer is whether factorized attention is worth it. If no fac-
Table 1. Quantitative results on the Argoverse 2 motion forecasting leaderboard [1] ranked by b-minFDE$_{6}$. Baselines that are known to have used ensemble are marked with symbol “$\ast$”. For each metric, the best result is in bold and the second best result is underlined.

<table>
<thead>
<tr>
<th>Method</th>
<th>b-minFDE$_{6}$</th>
<th>minADE$_{6}$</th>
<th>minFDE$_{6}$</th>
<th>MR$_{6}$</th>
<th>minADE$_{1}$</th>
<th>minFDE$_{1}$</th>
<th>MR$_{1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>THOMAS [17]</td>
<td>2.16</td>
<td>0.88</td>
<td>1.51</td>
<td>0.20</td>
<td>1.95</td>
<td>4.71</td>
<td>0.64</td>
</tr>
<tr>
<td>GoRela [9]</td>
<td>2.01</td>
<td>0.76</td>
<td>1.48</td>
<td>0.22</td>
<td>1.82</td>
<td>4.62</td>
<td>0.66</td>
</tr>
<tr>
<td>MTR [42]</td>
<td>1.98</td>
<td>0.73</td>
<td>1.44</td>
<td>0.15</td>
<td>1.74</td>
<td>4.39</td>
<td>0.58</td>
</tr>
<tr>
<td>GANet [48]</td>
<td>1.96</td>
<td>0.72</td>
<td>1.34</td>
<td>0.17</td>
<td>1.77</td>
<td>4.48</td>
<td>0.59</td>
</tr>
<tr>
<td>QML$^*$ [43]</td>
<td>1.95</td>
<td>0.69</td>
<td>1.39</td>
<td>0.19</td>
<td>1.84</td>
<td>4.98</td>
<td>0.62</td>
</tr>
<tr>
<td>BANet$^*$ [54]</td>
<td>1.92</td>
<td>0.71</td>
<td>1.36</td>
<td>0.19</td>
<td>1.79</td>
<td>4.61</td>
<td>0.60</td>
</tr>
<tr>
<td>QCNet (w/o ensemble)</td>
<td>1.91</td>
<td>0.65</td>
<td>1.29</td>
<td>0.16</td>
<td>1.69</td>
<td>4.30</td>
<td>0.59</td>
</tr>
<tr>
<td>QCNet (w/ ensemble)</td>
<td>1.78</td>
<td>0.62</td>
<td>1.19</td>
<td>0.14</td>
<td>1.56</td>
<td>3.96</td>
<td>0.55</td>
</tr>
</tbody>
</table>

Table 2. Quantitative results on the Argoverse 1 motion forecasting leaderboard [2]. The leaderboard is sorted by b-minFDE$_{6}$.

<table>
<thead>
<tr>
<th>Method</th>
<th>b-minFDE$_{6}$</th>
<th>minADE$_{6}$</th>
<th>minFDE$_{6}$</th>
<th>MR$_{6}$</th>
<th>b-minFDE$_{1}$</th>
<th>minADE$_{1}$</th>
<th>minFDE$_{1}$</th>
<th>MR$_{1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LaneGCN [31]</td>
<td>2.06</td>
<td>0.87</td>
<td>1.36</td>
<td>0.16</td>
<td>1.84</td>
<td>0.77</td>
<td>1.17</td>
<td>0.13</td>
</tr>
<tr>
<td>mmTransformer [32]</td>
<td>2.03</td>
<td>0.84</td>
<td>1.34</td>
<td>0.15</td>
<td>1.79</td>
<td>0.79</td>
<td>1.21</td>
<td>0.13</td>
</tr>
<tr>
<td>DenseTNT [19]</td>
<td>1.98</td>
<td>0.88</td>
<td>1.23</td>
<td>0.13</td>
<td>1.89</td>
<td>0.80</td>
<td>1.23</td>
<td>0.13</td>
</tr>
<tr>
<td>TPCN [50]</td>
<td>1.93</td>
<td>0.82</td>
<td>1.24</td>
<td>0.13</td>
<td>1.86</td>
<td>0.89</td>
<td>1.29</td>
<td>0.08</td>
</tr>
<tr>
<td>SceneTransformer [38]</td>
<td>1.89</td>
<td>0.80</td>
<td>1.23</td>
<td>0.13</td>
<td>1.76</td>
<td>0.77</td>
<td>1.14</td>
<td>0.11</td>
</tr>
<tr>
<td>HOME+GOHOME [15, 16]</td>
<td>1.86</td>
<td>0.89</td>
<td>1.29</td>
<td>0.08</td>
<td>1.74</td>
<td>0.77</td>
<td>1.16</td>
<td>0.12</td>
</tr>
<tr>
<td>HiVT [56]</td>
<td>1.84</td>
<td>0.77</td>
<td>1.17</td>
<td>0.13</td>
<td>1.69</td>
<td>0.73</td>
<td>1.07</td>
<td>0.11</td>
</tr>
<tr>
<td>MultiPath++ [46]</td>
<td>1.79</td>
<td>0.79</td>
<td>1.21</td>
<td>0.13</td>
<td>1.76</td>
<td>0.80</td>
<td>1.21</td>
<td>0.11</td>
</tr>
<tr>
<td>GANet [48]</td>
<td>1.79</td>
<td>0.81</td>
<td>1.16</td>
<td>0.12</td>
<td>1.76</td>
<td>0.77</td>
<td>1.14</td>
<td>0.11</td>
</tr>
<tr>
<td>PAGA [11]</td>
<td>1.76</td>
<td>0.80</td>
<td>1.21</td>
<td>0.11</td>
<td>1.76</td>
<td>0.77</td>
<td>1.14</td>
<td>0.11</td>
</tr>
<tr>
<td>DCMS [51]</td>
<td>1.76</td>
<td>0.77</td>
<td>1.14</td>
<td>0.11</td>
<td>1.74</td>
<td>0.77</td>
<td>1.16</td>
<td>0.12</td>
</tr>
<tr>
<td>Wayformer [37]</td>
<td>1.74</td>
<td>0.77</td>
<td>1.16</td>
<td>0.12</td>
<td>1.69</td>
<td>0.73</td>
<td>1.07</td>
<td>0.11</td>
</tr>
<tr>
<td>Ours</td>
<td>1.69</td>
<td>0.73</td>
<td>1.07</td>
<td>0.11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Models’ performance and inference latency evaluated on the Argoverse 2 validation set. We use an A40 GPU to measure encoders’ online inference latency in the densest traffic scene involving 190 agents and 169 map polygons.

<table>
<thead>
<tr>
<th>Model</th>
<th>Online Inference (ms)</th>
<th>b-minFDE$_{6}$</th>
<th>minADE$_{6}$</th>
<th>minFDE$_{6}$</th>
<th>MR$_{6}$</th>
<th>b-minFDE$_{1}$</th>
<th>minADE$_{1}$</th>
<th>minFDE$_{1}$</th>
<th>MR$_{1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>QCNet ($L_{enc} = 0$)</td>
<td>8±1</td>
<td>1±0</td>
<td>0.76</td>
<td>1.33</td>
<td>0.18</td>
<td>1±0</td>
<td>0.74</td>
<td>1.30</td>
<td>0.17</td>
</tr>
<tr>
<td>QCNet ($L_{enc} = 1$)</td>
<td>64±1</td>
<td>10±1</td>
<td>0.74</td>
<td>1.30</td>
<td>0.17</td>
<td>10±1</td>
<td>0.74</td>
<td>1.30</td>
<td>0.17</td>
</tr>
<tr>
<td>QCNet ($L_{enc} = 2$)</td>
<td>82±1</td>
<td>13±1</td>
<td>0.73</td>
<td>1.27</td>
<td>0.16</td>
<td>12±1</td>
<td>0.74</td>
<td>1.30</td>
<td>0.17</td>
</tr>
</tbody>
</table>

Table 4 demonstrates the effects of the recurrent mechanism and the refinement module on different datasets. On Argoverse 1, most agents merely exhibit trivial behavior, and the scene context usually does not have significant variation within the 3-second prediction horizon. For this reason, our design choices only bring marginal improvement when evaluated on this dataset. However, on the more challenging Argoverse 2 dataset where the prediction horizon is 6 seconds, increasing the number of recurrent steps from 1 (i.e., no recurrence) to 3 leads to much better long-term performance, and the refinement module offers a dramatic improvement in terms of both accuracy and multimodality. We also notice that using much more recurrent steps is redundant: when increasing the number from 3 to 6, the model performance on Argoverse 2 cannot be further improved.

4.4. Qualitative Results

We present some qualitative results on the Argoverse 2 validation set. Comparing Fig. 4a and Fig. 4b, we can see...
that the recurrent mechanism of the proposal module can reduce the prediction error in the long term. Figure 4c further demonstrates the effectiveness of the refinement module, which improves the diversity of multiple hypotheses and the smoothness of the predicted trajectories.

5. Conclusion

This paper introduces QCNet, a neural architecture that overcomes some important challenges in trajectory prediction. Powered by the design ethos of query-centric modeling, QCNet maintains the representational capability of factorized attention while enjoying much faster inference. It achieves multimodal and long-term prediction by employing a recurrent, anchor-free trajectory proposal module and an anchor-based refinement module. QCNet exhibits unprecedented performance on large-scale trajectory prediction datasets, demonstrating the effectiveness of its designs.

Acknowledgement

This work was partially supported by Hong Kong Research Grant Council under GRF 11200220, Science and Technology Innovation Committee Foundation of Shenzhen under Grant No. JCYJ20200109143223052.
References


