Latency-Aware Short-Term Video Action Anticipation and its Application in Trajectory Prediction

Harshayu Girase
Karttikeya Mangalam, Ed.
Jitendra Malik, Ed.

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2023-13
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-13.html

January 17, 2023

Acknowledgement

I am grateful for the many people who supported me during my academic journey. I want to express my sincere gratitude to my advisor, Professor Jitendra Malik, for the opportunity to research such fascinating problems. Thank you to Professor Trevor Darrell for feedback and reviewing this work. Thank you to my fantastic research mentor Karttikeya Mangalam who I have worked closely with for the past 3 years during both my undergrad and master's. I genuinely learned a lot during our collaborations and it has been a great pleasure. Thank you to all my friends that supported me throughout my journey.

Finally, thank you to my grandparents, parents, and sister for their love and support throughout my journey at Berkeley.

Latency-Aware Short-Term Video Action Anticipation and its Application in Trajectory Prediction

by

Harshayu Girase

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master's of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Jitendra Malik, Chair
Professor Trevor Darrell

Fall 2022

Abstract

Latency-Aware Short-Term Video Action Anticipation and its Application in Trajectory Prediction

by

Harshayu Girase

Master's of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Jitendra Malik, Chair

Following the successes of deep networks for image-based tasks, there has been an emphasis on developing video models to achieve similar feats. Many common tasks include video recognition, detection, and segmentation. However, only a few works study the task of video anticipation, which requires not only video understanding but also modeling of future behavior. In this work, we propose a novel self-supervised method to anticipate future actions in the short-term given a video input. As anticipation is a real-time problem, we highlight the importance of considering latency when developing and evaluating such models. In the first part of this article, we describe the task of short-term anticipation, dive into the current methodology, and propose a new metric and model for better evaluation of this task.

Furthermore, we also explore a specific application of short-term action anticipation: trajectory prediction. To demonstrate how the efficacy of short-term anticipation models can actually be utilized in practice, we create our own trajectory prediction dataset for both humans and vehicles. This dataset contains not only labeled trajectories for each agent but also detailed action labels. We propose a model that performs joint trajectory and future short-term action prediction. We demonstrate how the task of action anticipation can assist and improve the primary trajectory prediction task.

Acknowledgments

I am grateful for the many people who supported me during my academic journey. I want to express my sincere gratitude to my advisor, Professor Jitendra Malik, for the opportunity to research such fascinating problems. Thank you to Professor Trevor Darrell for feedback and reviewing this work. Thank you to my fantastic research mentor Karttikeya Mangalam who I have worked closely with for the past 3 years during both my undergrad and master's. I genuinely learned a lot during our collaborations and it has been a great pleasure. Thank you to all the Professors I have had the opportunity to take classes with during my undergrad and master's. Thank you to all my friends that supported me throughout my journey. Finally, thank you to my parents and sister for their love and support throughout my journey at Berkeley.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Latency matters. It is a crucial system design consideration for countless applications that operate in real-time from hardware design [86], network engineering [84], and satellite communications [39] to capital trading [41], human vision [79] and even COVID transmission patterns [74]. However, it has not been a center stage design consideration in modern computer vision systems of the past decade [58, 20]. Modern vision system design has largely focused on the correctness of the system rather than the latency of the predictions. Interestingly, recent neural network architectures have adopted FLOPs as a *proxy* for latency as a *second* axis for model design. While being a sufficient fidelity metric for offline after-the-fact applications like automatic content recognition, latency often comes second to correctness, even for online real-time systems such as forecasting models.

Forecasting empowers reactive planning [26]. An autonomous system present in rich human environments inevitably needs to understand human actions around it for smooth task planning and execution. Autonomous agent planning critically depends on anticipating the future of the scene in various forms such as trajectory prediction [30, 77, 78, 31], action forecasting [32, 28, 112] or future scene segmentation [14] and anticipating the future is a activity humans subconsciously do for day-to-day tasks [80]. And while vision based forecasting systems are often meant for embodied real-time deployment on autonomous agents like self-driving cars and robots, they are evaluated in an *offline* setting where inference latency is neglected (Figure 1.1).

In this work, we propose a *real-time* evaluation setting (Figure 1.1) that closely mimics the real-world deployment for a forecasting system. Suppose that in a real-time system, the design specifications require the forecasting system outputs t_f seconds in advance of the event to be able to plan and use the forecasts effectively. In current offline settings, the forecasting system begins the inference t_f seconds in advance of the event ('Present' in Figure 1.1) and the model latency is ignored (or assumed to be 0) such that the predictions are available instantly. However, in our proposed *real-time* setting, the model is required to start inference in advance of 'Present' so that the outputs are available with a horizon of t_f

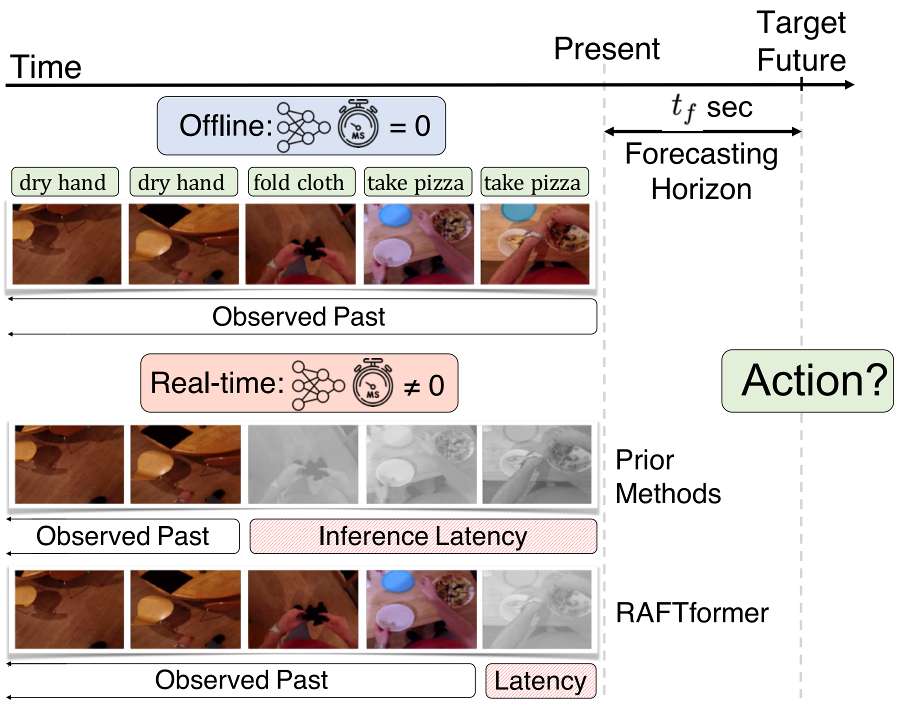


Figure 1.1: **Action Forecasting** is the task of predicting actions that will happen after a pre-determined time span, say t_f seconds, into the future. Prior works consider an **offline** evaluation setting that ignores the model inference latency. We propose a latency-aware **real-time** evaluation setting where the model is required to *finish* forecasting t_f seconds before the target time. We present RAFTformer, a fast action anticipation transformer that outperforms prior works both in offline & real-time setting while forecasting actions in real-time ($\geq$ 25 FPS).

seconds, meeting the design specification.

We observe that in the real-time setting, the prior works fare quite poorly because of their slow model inference latency (Table 4.2). A large latency implies that the model has to start inference further in the past and has to rely on older video data to make forecasts with the benefit of more expressiveness (Figure 1.2). A smaller latency means the model can enjoy more recent video data but has limited capacity. Simply said, models that are only evaluated in the offline setting may fare poorly in the real-time deployment setting due to their latency agnostic design (Figure 1.2).

We present, RAFTformer, a real-time action forecasting transformer that uses a two-stage transformer encoder-based network for lightning fast forecasts in inference. RAFTformer uses a shuffled casual masking scheme based feature prediction loss for learning strong temporal cues that transfer to feature prediction. Further, RAFTformer uses specialized

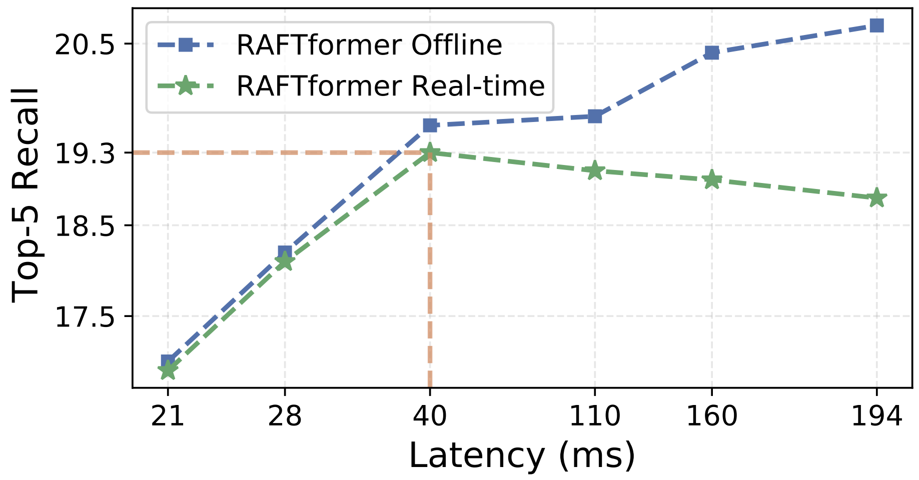


Figure 1.2: **Evaluation Performance Latency.** Bigger models perform better in latency agnostic offline settings. In the real-time evaluation setting, we observe that, beyond a limit, bigger models with higher latency cause a drop in forecasting performance. In practical deployment, there exists a trade-off between latency and high-fidelity forecasts. See 4.3 for details.

anticipation tokens for learning to predict action at multiple temporal horizons that improve model reasoning capabilities of short-term action forecasting as well. Finally, the model is explicitly designed for real-time embodied deployments that allows inference up to an order of magnitude faster than prior state-of-the-art methods. In summary, our contributions are three-fold:

First, we propose Real-time Action Forecasting Transformer (RAFTformer), a real-time action forecasting transformer with latency at least 9× smaller than prior state-of-the-art action forecasting methods. RAFTformer uses specialized anticipation tokens and a novel shuffled casual masking based self-supervision loss that allows it to outperform prior work while maintaining low latency with a reduction of 94% in GPU training time and 90% in number of trainable parameters compares to prior works. To the best of our knowledge, our work is the first to achieve action anticipation in real-time (25 fps).

Second, we propose a latency-aware real-time evaluation setting (Figure 1.1) that better mimics practical deployment settings for embodied forecasting systems. Real-time evaluation demonstrates a clear trade-off between inference latency and model forecasting fidelity, paving the path for development of latency-aware forecasting models in future.

Third, Through extensive experiments, we show that RAFTformer outperforms prior state-of-the-art methods by 4.9 points on the EGTEA Gaze+ dataset, by 1.4 points on the EPIC-Kitchens-100 dataset according to the Top-5 Recall metric and by a relative margin of 5.3% on the top-1 accuracy metric on EPIC-Kitchens-55 dataset.

Chapter 2

Background and Prior Work

2.1 Action Anticipation

Over the last few years, action anticipation has seen significant advances following the promising results in video recognition [48, 24, 123, 4, 50, 82, 73, 33, 112, 21, 6, 66] and video segmentation [108, 65, 53, 35, 113]. While earlier works [1, 75, 94] use CNN based methods for video action anticipation, many follow-up works transitioned into using recurrent sequence based networks [28, 85, 101, 89, 30]. Using multiple spatial and temporal scales was another key idea explored in several anticipation works [24, 122, 109, 100, 100, 112]. Masking based self-supervision has proven to be a new frontier for both image [7, 42, 23, 83, 45] and video [40, 105] representation learning. This concept has also been explored in the context of video anticipation [32] which differs from prior works that explore temporal consistency [25, 47, 52, 111, 115], inter-frame predictability [38, 37, 46], and cross-modal correspondence [5, 54, 104]. In our work we propose a novel generalized self-supervision scheme which is a crucial part of our models ability to generalize and outperform SOTA.

2.2 Transformers for Video Anticipation

With the rise of transformers for sequence tasks, recent works have explored various image and video based transformers [32, 34, 112, 121, 34, 110] for anticipating actions. [32] proposes a ViT-based [20] spatial transformer backbone with a transformer decoder head to anticipate next action with a 1s horizon. [32] predicts actions in the same latent space as their feature decodings causing conflation between feature reconstruction and future prediction. We propose specialized anticipation tokens to address this problem. Furthermore, [32] uses a recurrent decoder and frame-level operation, which is inefficient for both training compute and inference latency. MeMViT [112] proposes another fully transformer model which extends MViT [21] for better long-range modeling using a novel caching mechanism. However, their model is trained to only predict next action and thus does not explicitly learn to model the future or evolving scene dynamics. Furthermore, their model is less efficient than RAFTformer in terms of both training and inference time due to its larger model and input size and complete end-to-end training. RAFTformer uses a fixed pre-trained video network and only trains the RAFTformer network. Recently [34] propose a transformer based model for long-term action anticipation. They focus on a sequential prediction rather than next-action prediction and does not focus on inference latency.

2.3 Real-Time Systems

Latency matters, especially for real-time applications. Autonomous agents need to reason about nearby agents and take real-time decisions. This vision has led to great progress in the development of real-time systems in the related fields of semantic segmentation [22], video object segmentation [108], object detection [91, 72, 68, 13], multi-object tracking [51]. While some progress has also been made in activity understanding [118, 103], it is limited to recognition and detection. Keeping this in mind, some recent works in action anticipation have also started focusing on efficiency and memory footprint and report training time, inference time and trainable parameters [112]. Although a step in the right direction, these works have high inference times in comparison to their desired anticipation horizon. Our proposed method is significantly faster during both training and inference, has fewer trainable parameters and smaller memory footprint, and still outperforms state of the art methods.

Chapter 3

Real-Time Action Forecasting Transformer

In this section, we first present the problem formulation (3.1), followed by the architectural details of the video backbone for feature extraction (3.2) and the RAFTformer model architecture (3.3) including anticipation tokens (3.4), shuffled causal masking (3.4) and permutation encodings (3.4), followed by an outline of the loss functions used for training the model (3.6).

3.1 Problem Formulation

Given an observed video starting from time $T = 0$ and of arbitrary length t, $V_{0,t} = [F_0, ..., F_t]$ where F_i denotes the frame at time i, the task is to predict the future action t_f seconds in the future, , action A_{t+t_f} at time $T = t + t_f$.

3.2 Pre-trained Video Backbone

In contrast to state-of-the-art models [112, 32] that train end-to-end, we demonstrate that a 2-stage training process is both efficient and produces high-fidelity forecasts. First, we split the full duration of the past video $V_{0,t}$ into sub-clips $V = [C_0, C_1, ..., C_N]$ with a sliding window approach. Each clip, C_i is independently processed with the short-term video backbone like MViT [21] to extract clip-level features. A video backbone produces much richer spatio-temporal clip level features than an image backbone that lose temporal context like ViT [20], which has used in some prior works [32]. Further, using a video backbone also allows lower latency since the produced clip embeddings already contain fused features for several images at once.

Our two-stage design also allows for a hierarchical temporal processing of the long-form past video. The short-term recognition backbone model operates on short, high resolution clips.

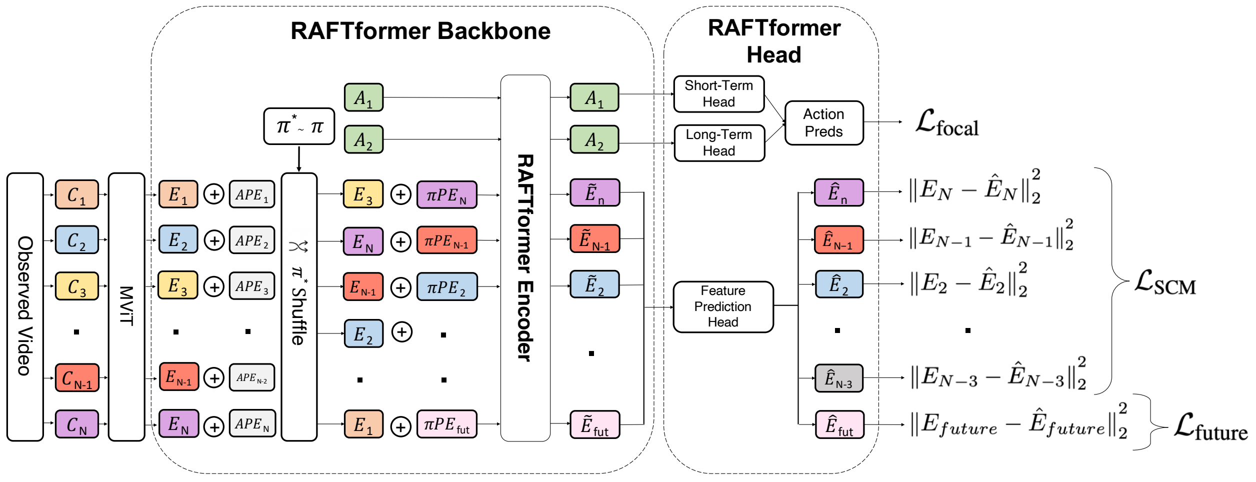

RAFTformer Backbone
RAFTformer Head
Observed Video
MViT
π* Shuffle
RAFTformer Encoder
Short-Term Head
Long-Term Head
Action Preds
Feature Prediction Head
$\mathcal{L}_{\text{focal}}$
$\mathcal{L}_{\text{SCM}}$
$\mathcal{L}_{\text{future}}$
$\pi^* \sim \pi$
$\|E_N - \hat{E}_N\|_2^2$
$\|E_{N-1} - \hat{E}_{N-1}\|_2^2$
$\|E_2 - \hat{E}_2\|_2^2$
$\|E_{N-3} - \hat{E}_{N-3}\|_2^2$
$\|E_{future} - \hat{E}_{future}\|_2^2$

Figure 3.1: **RAFTformer** is a real-time action anticipation transformer architecture comprised of two stages. First stage is a pre-trained short-term backbone that produces individual clip embeddings independently of other clips (3.2). In the second stage (3.3), absolute position encodings are added and the resulting sequence is shuffled with a sampled permutation $\pi^* \sim \pi$. The permutation encodings (3.4) are then added to the shuffled sequence and after concatenation with anticipation tokens, the sequence is processed with the RAFTformer encoder. The output tokens are decoded via short & long-term action anticipation heads (3.5), as well as the feature prediction head and trained with self-supervised loss ($\mathcal{L}_{\text{SCM}}$), future feature prediction loss ($\mathcal{L}_{\text{future}}$) and action forecasting loss ($\mathcal{L}_{\text{focal}}$) (3.6).

These extracted features are then used as input to the head network to process longer-range lower resolution features which captures key information from each clip. This is a crucial design consideration to lower the inference latency in RAFTformer while still capturing longer-range temporal dependencies compared to [32, 28].

3.3 RAFTformer Model Network

We propose the RAFTformer encoder be a transformer encoder model. The transformer encoder has the advantage of being able to effectively learn across clip dependencies by attending over independently extracted clip features without facing memory bottlenecks such as faced in LSTM encoders [44] that have been used in some prior works [28, 100]. However, to make the encoder effective for action forecasting on pre-trained features, we propose several changes to the training process to allow two-staged training to work as well as end-to-end training for action forecasting.

3.4 Anticipation Tokens

The extracted clip embeddings $[C_0 \cdots C_{N-1}]$ form the first part of input to the action anticipation head. For, the second part we propose to train learnable 'anticipation tokens' that can aggregate global context and later can be decoded into the future predictions. This design choice stands in contrast to prior works like [32] where the the output of the anticipation token is implicitly designed to be in the same latent space as the output of their image-feature tokens. In contrast to prior works, we find that temporal aggregation such as mean pooling for forecasting, leads to subpar performance (see Table 4.3). Hence, we use the output tokens corresponding to $[C_0 \cdots C_N]$ solely for self-supervised feature loss rather than action anticipation.

Instead, for action anticipation we propose to train learnable 'anticipation tokens' to learn useful global context from the clip tokens. The output of the anticipation token is no longer restricted to be in the same latent space as the output of clip-feature tokens like in [32] and can better capture information needed to anticipate the next action. Further, we propose to use multiple anticipation tokens to generate additional supervision, with each token attending to different past video length and producing forecasts for different time horizons in the future.

Self-Supervision via Shuffled Feature Prediction

Prior works have explored using the self-supervision task of predicting frame or clip features and using a MSE loss[32]. We propose to use a generalized form of masking based self-supervision [42] based on predicting shuffled future features. Predicting future clip features has a two-fold benefit: (1) It encourages causally learning the observed sequence incentivizing the model to grasp the underlying scene dynamics and (2) Prediction in the latent feature space allows reasoning semantically about the future without wasting modeling capacity with low pixel level scene details.

Different from loss functions proposed in prior works [32], we propose an improved auto-regressive scheme for self-supervised learning by future feature prediction. Rather than sequentially predicting missing clip features in order using a causal attention mask, we propose to use a model-level augmentation of the attention masking scheme where some of the attention weights are not used from the original causal mask.

Random Masking. Construction of sparse augmented attention mask is non-trivial. Simply generating a random mask such as in MAE [42] **fails** to isolate information within the intended partitioning due to multi-hop message passing caused by repeated application of the same mask. For example, consider Figure 3.2. We illustrate the information available to each token as it passes through multiple layers with a fixed random masking scheme. Before the first layer, each token has access to only its own features hence, the

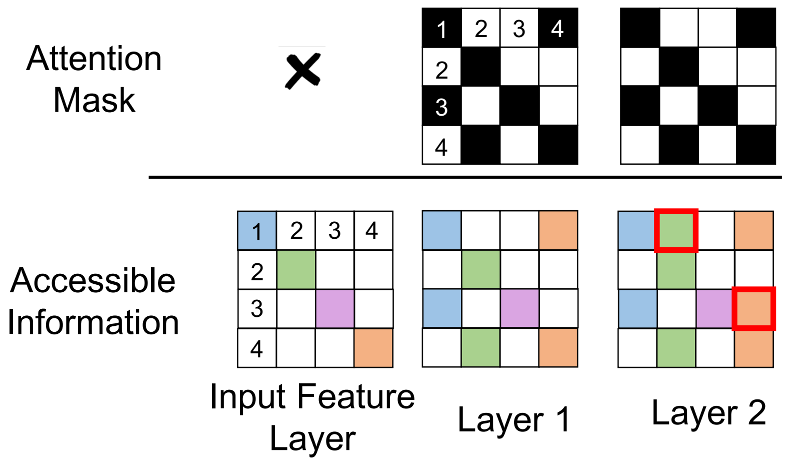


Figure 3.2: **Random Masking Self-Supervision** Illustration of how naive random masking **fails** for self-supervised feature prediction task. T_i represents the embeddings for clip i, as it transforms through the transformer layers. We can see that in Layer 1 the tokens can only access information according to the provided mask. However, if the same mask is used in Layer 2, there is information leakage across tokens which is undesirable.

accessible information matrix is diagonal. After Layer 1, each token now has access to tokens that were not masked (white) in the attention mask at Layer 1. However after Layer 2, because of multi-hop message-passing, some tokens (1 and 3 in Figure 3.2) have access to information from other tokens that are actually masked in the token's attention matrix (2 and 3 respectively, shown in red grid border).

Hence, improper masking can cause temporal information leakage, causing self-supervision to fail. For example, self-supervising token 1 with token 2 feature prediction would fail in Figure 3.2, despite token 2 being masked in the first row of the attention matrix. To prevent such an objective collapse, a careful mask generation scheme is needed, so that there is no unintended multi-hop information leakage. Instead, we propose a simple yet effective solution.

Shuffled Causal Masking. We know that vanilla causal masks ensure that information available to each token is invariant under repeated applications of the attention mask, , multi-hop message passing. Since multi-hop message passing is token permutation invariant, any permutation of the tokens from the causal mask will preserve the invariance of accessible information under multiple hops. Thus, this allows a general framework for structured randomized mask construction without temporal leakage. We notice that a row-wise permutation of the attention mask is equivalent to the same permutation applied

Auto-regressive $[1, 2, 3, 4]$	Shuffle $\pi_1 : [4, 1, 3, 2]$	Another Shuffle $\pi_2 : [3, 4, 2, 1]$
$1 \mapsto 2$	$4 \mapsto 1$	$3 \mapsto 4$
$2 \mapsto 3$	$1 \mapsto 3$	$4 \mapsto 2$
$3 \mapsto 4$	$3 \mapsto 2$	$2 \mapsto 1$

Table 3.1: **Encoding the input permutation π.** Shuffling the input sequence arbitrarily changes the successor of each token.

to the sequence itself. Hence, the same effect as a properly constructed random masks can simply be achieved by shuffling the input token sequence itself. This allows generalizing vanilla auto-regressive prediction order to arbitrary sequence permutations without any multi-hop information leakage through the layers. Thus, rather predicting clip features sequentially from 0 to $N - 1$ like [32], our proposed scheme allows for *exponentially* more variations.

Referring to Figure 3.1, we first add absolute position embeddings (APE) to each of the clip features before shuffling them. This allows transformer to leverage the information about the *actual* temporal order of each clip in the video. Without the absolute position embeddings, the transformer in input permutation equivariant which is not a desirable property for action forecasting. However, while APE is sufficient for positional information in vanilla auto-regressive ordering, it is not enough for prediction under our proposed Shuffled Causal Masking (SCM) scheme. In vanilla autoregressive ordering, for any specific token, the next token in the sequence corresponds to a fixed position and hence the self-supervised training process can subtly learn this bias via next token feature supervision. In contrast, under SCM the temporal position of the next token varies with the shuffling permutation and hence the self-supervised training cannot learn perform effective feature prediction without the information about the shuffling permutation.

Permutation Position Encoding

Naïve Encodings. For an L length sequence, there exist $L!$ possible admissible permutations. Encoding each permutation (π) by itself is clearly intractable. First, we observe that enocding π^* as a set of a $O(L)$ embeddings which are shared among different π is more efficient. A follow-up solution would be to instead encode each predecessor $\mapsto$ successor relationship as an encoding and have $L - 1$ of such embeddings together encode π^*. This reduces to total number of required embeddings to $L(L - 1)$ from $L!$. Each π^* now shares every one of its L embeddings with other π, but considered as a set, the L embeddings uniquely encode π^*. However, even L^2 becomes intractable for large L.

Permutation Position Encodings (πPE) provides an elegant solution for encoding π^*

requiring only L total encodings to be learnt. πPE encodes the predecessor $\mapsto$ successor relationships but further simplifies by noting that adding the encoding to the token itself makes the predecessor information redundant. The token itself is the predecessor and has the positional information available from APE. Hence, we simply encode the successor positional information and that in combination with APE uniquely encodes π^*. Hence, we design πPE to be the encoding of the *original* temporal position of the successor in the permuted sequence. So for the π_1 permutation in Table 3.1, we would add πPE[1] to token 4, πPE[3] to token 1 and πPE[2] to token 3.

3.5 Overall Mechanism

The past video is split into clips, and the clip embeddings $[C_0 \cdots C_{N-1}]$ are extracted using a video backbone (Figure 3.1). Anticipation tokens are concatenated to the extracted sequence (3.4), followed by addition of absolute position encoding to the concatenated sequence. Now, the clip embedding are shuffled according to a randomly chosen permutation π^* to obtain $[C_{\pi^*[0]} \cdots C_{\pi^*[N-1]}]$. The sampled permutation π^* is then encoded by adding the successor's (3.4) embedding to each token. In Figure 3.1, $\pi^*[1] = 3$ and $\pi^*[2] = N$, hence $\pi\text{PE}[1] = N$ which is used to self-supervise the feature at the second position in the shuffle sequence, , token 3 (3.4). The shuffled input sequence, in addition to the anticipation tokens, is now propagated through the transformer encoder using a causal attention masking scheme and the output is decoded using the RAFTformer head networks.

Head Networks We propose using three MLP heads on top of the transformer encoder network (Figure 3.1). Two MLP heads decode the anticipation tokens into the predicted future action distribution. The third MLP head upsamples the encoded tokens to the original representation space of the input tokens to allow self-supervision loss.

3.6 Loss Functions

We observe that the ground truth action distributions are often long-tailed, and propose to use the the focal loss [69], for supervising the future action prediction distributions.

$$\mathcal{L}_{\text{focal}} = \sum_{A_i \in A} \sum_{i=0}^{n} -(1 - p_{A_i})^{\gamma} \log(p_{A_i})$$

where p_i is the predicted probability for the correct class for the ith example, A_i represents predictions from a specific anticipation token and γ is the focusing parameter where $\gamma = 0$ is cross entropy loss. Increasing γ results in increased penalty for hard, misclassified examples. For self-supervision using shuffled causal masking to predict next-token embeddings, we use an expected ℓ_2 loss over both past tokens, with the expectation being over sampled permutations $\pi^* \sim \pi$.

$$\mathcal{L}_{\text{SCM}} = \mathop{\mathbb{E}}_{\pi^* \sim \pi} \sum_{j=0}^{N-1} \left\| E_{\pi^*[j]} - \hat{E}_{\pi^*[j]} \right\|_2^2$$

where $E_{\pi^*[j]}$ and $\hat{E}_{\pi^*[j]}$ denotes the original and the predicted clip embedding at position j after permuting with π^*. For future token prediction, we use a simple ℓ_2 loss

$$\mathcal{L}_{\text{future}} = \|E_{\text{future}} - \hat{E}_{\text{future}}\|_2^2$$

Finally, the overall loss is simply a weighed sum,

$$\mathcal{L} = \mathcal{L}_{\text{focal}} + \lambda_1 \mathcal{L}_{\text{SCM}} + \lambda_2 \mathcal{L}_{\text{future}}$$

Chapter 4

Experiments and Results

4.1 Datasets & Metrics

In this work, we explore the widely used EPIC-KITCHENS dataset [16], an unscripted dataset with nearly 20x more action classes and $10 - 100$x more observed sequences than other action datasets such as Breakfast dataset [59] and 50Salads [61] dataset. We benchmark the EPIC-KITCHENS dataset on both the EPIC-55 and EPIC-100 anticipation splits and use the same training/testing split as in prior works [32, 28, 112, 100]. The EPIC-55 dataset consists of 39,600 segments split up from raw, unscripted videos of humans performing 2,513 actions in the kitchen. The EPIC-100 dataset contains close to 90,000 segments with 3,806 actions. For EK55, we use the Top-1 accuracy metric which was the primary metric used by most works on this dataset. For EK100, we use the Top-5 Action Recall metric to compare performance as done in most prior works for this dataset.

We also report results on the EGTEA+ Gaze dataset [67] which contains 106 actions from over 10,000 segments. Note that we use the same training/testing split (5-fold cross-validation average) as reported in prior baselines/works [28]. We report with anticipation horizon, $t_f = 1$ so it is the same for all our datasets in the offline setting.

Since human decision-making is inherently multimodal, we propose an approach to predicting multiple reasonable future action forecasts. In addition to top-1 accuracy, we report top-5 recall following prior works [28, 32, 112]. We use the baseline data splits and report metrics on both the validation and test set. In addition, we report the number of trainable model parameters (M), total compute spent for training the model to convergence on a Tesla V100 GPU (in hours), and the inference latency (in milliseconds).

4.2 Evaluation Setting

Offline evaluation

In this setting, model inference latency is ignored, or in other words, assumed to be zero. All prior works [28, 32, 112] consider this setting. Referring to Figure 1.1, prior works assume access to all past video frames up till the present moment $T = t$. Using this information, the model then predicts that action with a t_f second horizon at $T = t + t_f$. However, the prediction would actually be produced at time $T = t + t_l$ where t_l is the model inference latency. This is not practically useful since often, we require time horizon t_f to meaningfully use the predicted future outcomes. Further, this does not account for absurdities where a large model might even have latency $t_l > t_f$, in which case the model is predicting an action that has already happened by the time it predicts it! A complex model can have great offline performance but might have too long of an inference time for it to be useful in practice.

Real-time evaluation.

To remedy this impractical situation, we consider the real-time evaluation setting. In this setting, the model is required to finish inference with at least t_f seconds horizon before the target time (Fig. 1.1). Hence, the model is allowed access to past video data only for $T < t - t_l$. This setting even allows for large models where $t_l > t_f$ since predictions would still be produced with t_f seconds before the target time. Unlike other recognition scenarios where latency is a mere annoyance, in the case of future forecasting, models are often used in real-time rather than offline. The offline setting subtly oversteps the prediction horizon t_f, by receiving forecasts with a margin much less than the postulated t_f horizon. By coupling model latency to the past video data observed, the real-time setting incentivizes the development of efficient forecasting methods that utilize the recent data better than relying on slow large models that cannot miss the recent frames, which arguably are the most crucial for near future prediction.

4.3 EPIC-Kitchens-100

Network Details

For feature extraction, we fix the pre-trained 16x4 (K400+IN1K) and 32x3 (K700) MViT-B backbones [21, 66] trained for recognition on the EK100 dataset. Each clip is embedded as a 768-dimensional feature vector. Unless mentioned otherwise, the action forecasting experiments use a $t_f = 1$ second horizon. RAFTformer encoder is a lightweight 4 layer, 4 head transformer encoder using post-normalization and ReLU activations. A linear projection from up-projects $768 \to 1024$ for the transformer. Only during training, we use an input shuffling probability of 0.3. The output of the encoder is down projected from $1024 \to 768$ channels. Both short-term and long-term action prediction heads are fast

& lightweight MLPs (1024 → 2048 → 3806). We set $\lambda_1 = \lambda_2 = 14$. For other details, please see supplementary.

Offline Evaluation

Split	Method	Addl. Modality	Init	Epic Boxes	Top-5 Recall			Parameters	GPU	Inference
					Verb	Noun	Action	($\times 10^6$)	Hours	Latency (ms)
Val	TempAgg [100]	None	IN1K		24.2	29.8	13.0	-	-	-
	RULSTM [28]	None	IN1K		-	-	13.3	-	-	-
	RULSTM [28]	Obj+Flow	IN1K	✓	30.8	27.8	14.0	-	-	-
	TempAgg [100]	Obj+Flow+ROI	IN1K	✓	23.2	31.4	14.7	-	-	-
	AVT [32]	None	IN21K		30.2	31.7	14.9	378	-	420
	AVT+ [32]	Obj	IN21K	✓	28.2	32.0	15.9	-	-	-
	TSN-AVT+ [32]	Obj	IN21K	✓	31.8	25.5	14.8	-	-	-
	MeMVit [112]	None	K400		32.8	33.2	15.1	59	-	160
	MeMVit [112]	None	K700		32.2	37.0	17.7	212	368	350
	RAFTformer	None	K400 + IN1K		33.3	35.5	17.6	**26**	**23**	**40**
	RAFTformer	None	K700		33.7	37.1	18.0	**26**	27	110
	RAFTformer-2B	None	K700 + IN1K		**33.8**	**37.9**	**19.1**	52	50	160
Test	RULSTM [28]	Obj+Flow	IN1K	✓	25.3	26.7	11.2	-	-	-
	TBN [117]	Obj+Flow	IN1K	✓	21.5	26.8	11.0	-	-	-
	AVT+ [32]	Obj+Flow	IN21K	✓	25.6	28.8	12.6	-	-	-
	Abstract Goal [95]	Obj+Flow	IN1K	✓	31.4	30.1	14.3	-	-	-
	AFFT [121]	Obj+Flow	-	✓	20.7	31.8	14.9	-	-	-
	RAFTformer	None	K400 + IN1K		27.3	32.8	14.0	26	23	40
	RAFTformer	None	K700		27.4	34.0	14.7	26	27	110
	RAFTformer-2B	None	K700 + IN1K		**30.1**	**34.1**	**15.4**	52	50	160

Table 4.1: **Offline Evaluation Results** on the EK-100 dataset for $t_f = 1$ second horizon. Latency is measured over the entire model including the backbone & head networks on a single 16G Tesla V100 GPU. Methods that use other modalities (+RGB) are deemphasized.

In Table 4.1 we first report RAFTformer results against several prior works, including previous SOTA MeMViT [112], on the validation set. Results that use additional modalities like Object (Obj), Flow (Flow), Object Region of Interest (ROI) or Epic Kitchen boxes are de-emphasized. Using only RGB inputs, RAFTformer (K700) outperforms the AVT RGB model in action T5R by 3.1 points and the AVT RGB+Obj model by 2.1 points. RAFTformer slightly outperforms MeMViT action T5R while predicting actions **8.75**× faster with **8.2**× lesser trainable parameters and a **16**× faster training time. We also combine two separate RAFTformer models to train a two-backbone model (RAFTformer-2B) that achieves state-of-the-art by a large margin of 1.4% improvement in T5R performance. In the second section, we report our submitted results to the EK100 test server. We observe that RAFTformer generalizes well to the test set, improving upon the AVT+ multimodal ensemble model [32] by 2.8 T5R points.

Model	Init	Latency (t_l ms)	Inference Start Time Stamp	Inference End Time Stamp	Target Time Stamp	Top-5 Recall Verb	Noun	Action
AVT[32]	IN21K	$t_{\text{avt}} = 420$	T	$T + t_{\text{avt}}$	$T + 1$	30.2	31.7	14.9
RAFTformer	K400 + IN1k	$t_{\text{ours}} = 40$	$T + t_{\text{avt}} - t_{\text{ours}}$	$T + t_{\text{avt}}$	$T + 1$	34.1	38.2	**19.3** (+4.4)
MemViT [112]	K400	$t_{\text{vit}} = 160$	T	$T + t_{\text{vit}}$	$T + 1$	32.8	33.2	15.1
RAFTformer	K400 + IN1k	$t_{\text{ours}} = 40$	$T + t_{\text{vit}} - t_{\text{ours}}$	$T + t_{\text{vit}}$	$T + 1$	33.8	37.1	**18.1** (+3.0)
MemViT [112]	K700	$t_{\text{vit}} = 350$	T	$T + t_{\text{vit}}$	$T + 1$	32.2	37.0	17.7
RAFTformer	K400 + IN1k	$t_{\text{ours}} = 40$	$T + t_{\text{vit}} - t_{\text{ours}}$	$T + t_{\text{vit}}$	$T + 1$	33.7	37.9	**19.0** (+1.3)

Table 4.2: **Real-time Evaluation Results** for benchmarking action forecasting methods in a practical setting (4.2). Each comparison is performed between a pair of models where their start time ('Inference Start') times have been adjusted (Figure 1.1) by their latency so that they produce the forecasting output for the 'Target Time' *simultaneously* ('Inference End'). For prior works [32, 112], start & end times are kept same as their original offline settings (Table 4.1) to avoid any training recipe change. Faster models can pragmatically utilize recent frames while slower models must rely on higher fidelity prediction from older frames. Latency measured on a single 16G Tesla V100 GPU.

Real-time Evaluation

In table 4.2, we fix the target - start time difference to be what the prior models such as, AVT [32] or MeMViT [112] were designed for, to avoid any unintended change in their training recipes. Hence, we shift RAFTformer start time so that the output forecasts are produced simultaneously for the pair.

We observe that RAFTformer outperforms prior methods by an even *larger margin* in the real-time evaluation setting than in offline evaluation setting. RAFTformer effectively utilizes its compute to produce the maximum forecasting effect at least latency cost. This allows RAFTformer to exploit more recent information that slower models miss because of higher inference latencies ($t_{\text{ours}} < \{t_{\text{avt}}, t_{\text{vit}}\}$). This also shows that disproportional effect of inference latency on forecasting performance. While RAFTFormer K400 + IN1K has similar Top-5 Recall as MeMViT K700 in the offline setting, in the practical real-time evaluation (Table 4.2), RAFTformer outperforms MeMViT by 1.3% by leveraging its faster inference latency (40 350 ms).

Latency vs. Offline & Real-time Forecasting

In offline evaluation, latency is ignored and all models data access parity. As scaling laws [49] would predict, bigger models (with higher latency) have stronger forecasting performance (Blue curve in Fig. 1.2). In the real-time setting, a higher latency implies access to less recent data (Green curve in Fig. 1.2) and hence, an interesting trade-off materializes. We evaluate this by varying our model size. At first, as latency (and model size) increases, the performance improves. This is because the benefit of having a more expressive model outweighs the cost of access to older video data. However, as the latency increases further, the

Model	Setting				Top-5
	AT	FL	SCM	2×AT	Recall
Mean pooling					15.2
RAFTformer	✓				16.1 **(+0.9)**
	✓	✓			16.8 **(+0.7)**
	✓	✓	✓		17.4 **(+0.6)**
	✓	✓	✓	✓	**17.6** **(+0.2)**

Table 4.3: **EK100 Ablation Study.** FL is feature loss (no SCM), SCM is Shuffled Causal Masking (3.4), AT (2×) are single (double) anticipation tokens (3.4).

real-time performance starts decreasing. Now the harm of not having access to recent data outweighs the benefit of a more expressive model. For our setting, the optimal RAFTformer model has a latency of 40 ms with an offline and real-time performance of 19.6 and 19.3 Top-5 recall respectively. A similar trade-off would exist for any combination of dataset, model and deployment hardware regimes. Plots are for RAFTformer models of varying backbone parameter count for $t_f = 0.58$ second prediction on EK100.

Ablations

We ablate RAFTformer thoroughly in Table 4.3. In contrast to prior works [32, 112] that either pool information or use the same embedding for feature supervision and action prediction, we train explicit anticipation tokens (3.4) to perform forecasting, improving performance by 1.1 T5R. Anticipation tokens specialize in predicting next action instead of using features that multi-task between reconstructing clip features and future action prediction [32]. Self-supervision via feature loss (FL) further improves by 0.7 T5R. Further, our proposed SCM technique effectively regularizes feature loss with input permutations and improves 0.6 T5R. Finally, an additional anticipation token (2×AT) for supervision further improves 0.2 T5R. Additional longer-horizon anticipation task helps with learning relevant global video features that transfer over to the main anticipation task.

4.4 Additional Datasets

EPIC-Kitchens 55

We also compare RAFTformer to prior baselines on the EK55 dataset in Table 4.4 on top-1 accuracy. We evaluate against other models using the same initialization for fair comparison. We observe that RAFTformer outperforms prior benchmarks by about 0.7% using the exact same backbone features. This shows the superiority of our proposed shuffle causal masking (3.4) & anticipation token prediction (3.4) in a controlled setting.

Model	Init	Top-1 Acc
RULSTM [28]	TSN/IN1k	13.1
ActionBanks [100]	TSN/IN1k	12.3
AVT-h [32]	TSN/IN1k	13.1
RAFTformer	TSN/IN1k	**13.8**

Table 4.4: **Offline Evaluation** top-1 accuracy results on the Epic-Kitchens-55 dataset.

Model	Init	Modality	Top-5 Recall
DMR [107]	-	RGB	38.1
ATSN [16]	TSN/IN1k	RGB+Flow	31.6
MCE [27]	TSN/IN1k	RGB+Flow	43.8
TCN [8]	-	RGB	47.1
FN [17]	VGG-16	RGB	42.7
ED [29]	VGG-16/TS	RGB+Flow	54.6
RULSTM [28]	TSN/IN1k	RGB+Obj+Flow	58.6
RAFTformer	TSN/IN1k	RGB	**63.5**

Table 4.5: **EGTEA Gaze+.** Under same initialization, RAFTformer significantly outperforms all prior methods without using any additional modalities.

EGTEA Gaze+

We also evaluate on the EGTEA Gaze+ dataset in Table 4.5 to showcase RAFTformer performance on new settings outside of EPIC-Kitchens. Again, we use the setting of anticipating 1s into the future and use the T5R metric. While prior methods use many input modalities including as object, flow, and RGB, we significantly outperform the prior state-of-the-art in this setting [28] by 4.9%. Note that we use pre-extracted TSN RGB features as provided by [28] for direct comparison. Our very strong performance on another egocentric video dataset with complex human tasks further validates the promise of RAFTformer.

4.5 Additional Results

Object and Flow Feature Results

In Tables 4.6 and 4.7 we present results on EK55 dataset (Top-1 Accuracy), using flow and object features as input. Note that this is the standard offline setting. We show that even with these modalities (which is not used for our final model), we outperform prior models. Note that the flow and object features are the same used in [28]. All performance

Model	Top-5 Recall
RULSTM [28]	7.8
AVT [32]	8.7
RAFTformer	**9.5**

Table 4.6: Results of our model compared to others on flow features provided by [28]. Note that all models used the same input features, but we still outperform prior SOTA. Results are evaluated using the same train/val split [28, 32] which we refer to as the offline setting ($t_f = 1s$)

Model	Top-5 Recall
RULSTM [28]	7.2
AVT [32]	6.7
RAFTformer	**7.5**

Table 4.7: Results of our model compared to others on object features provided by [28]. Note that all models used the same input features, but we still outperform prior SOTA. Results are evaluated using the same train/val split [28, 32] which we refer to as the offline setting ($t_f = 1s$).

improvements seen in these tables are due to the RAFTformer Head architecture (Anticipation Tokens and Shuffled Causal Masking), since the backbone input features are the exact same (frame-wise TSN features [28]).

Qualitative Results

In figure 4.1, we visualize multiple examples where AVT has incorrect predictions, but RAFTformer is able to correctly predict the next action. In figure 4.2, we show cases where RAFTformer has incorrect predictions.

4.6 Implementation Details

MViT Backbone

We perform our own feature extraction using the MViT=B [66, 21] backbone. For the K400+IN1K model, we use the 16x4 model pre=trained for short-term action recognition on the Kinetics-400 dataset. The 16x4 model uses 16 frames sampled 4 frames apart at a 30fps. This translates to each clip being of length 2 seconds sampled at 8fps. We sample directly from the RGB frames provided in the EPIC-KITCHENS-100 dataset and do not use flow or object features that few other works such as [28, 32]. Our K700 model uses a different 32x3 pre-trained MViT on the K700 dataset. This model uses 32 frames sampled 3 frames apart at a 30fps; while the performance is better due to the heavier model architecture, the inference time is slower. This trade-off is explored in detail in the main paper.

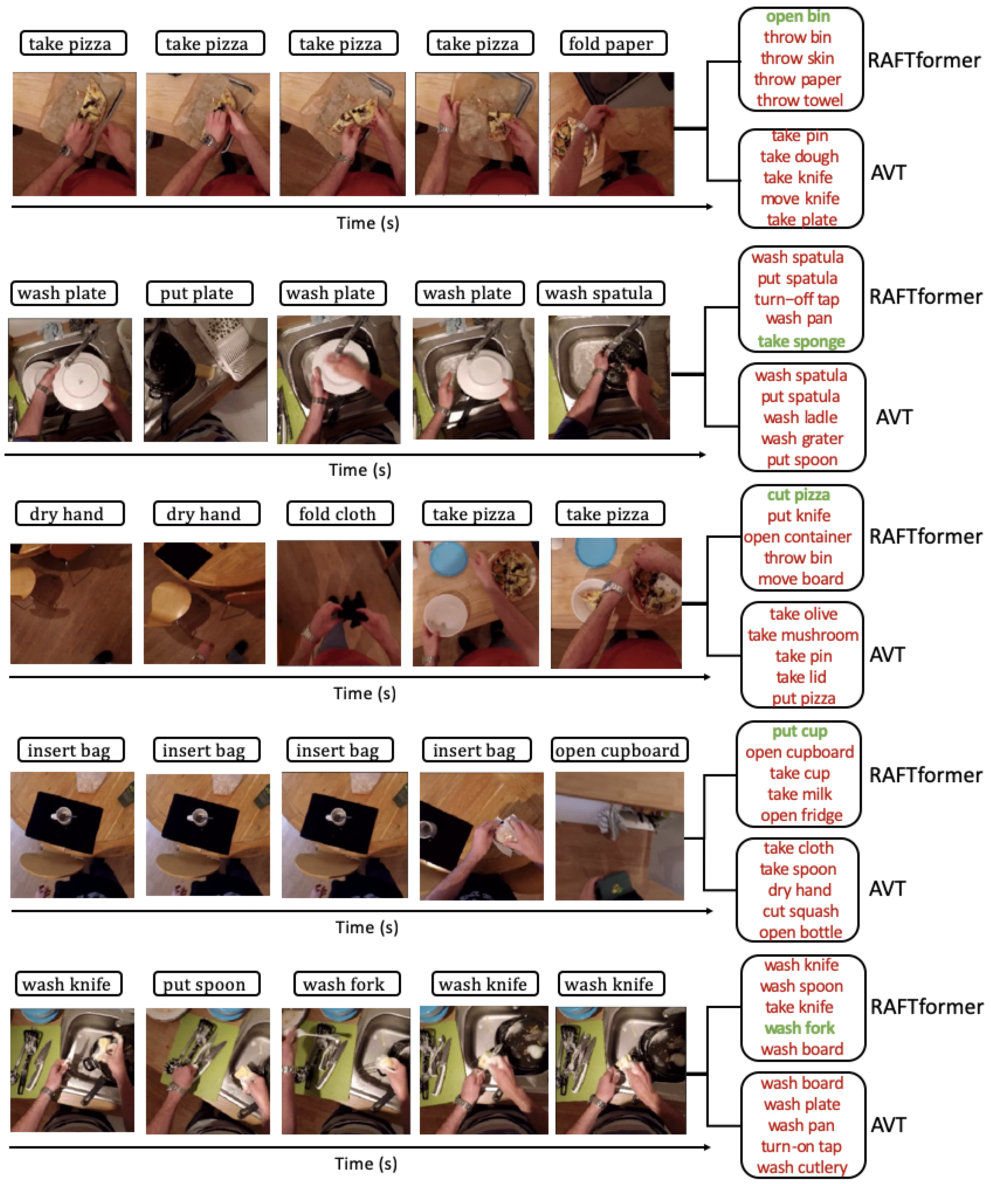


Figure 4.1: This figure shows qualitative examples in which RAFTformer predicts the next action correctly, but AVT [32] is incorrect. Note that we show Top-5 predictions. The frames and their corresponding labels show the past video and corresponding actions (from 16s ago to the present).

Pre-processing Details

We find that proper data pre-processing is critical to model performance. We use a center crop of 224x224 and normalize the images after cropping to $mean = [0.45, 0.45, 0.45]$ and

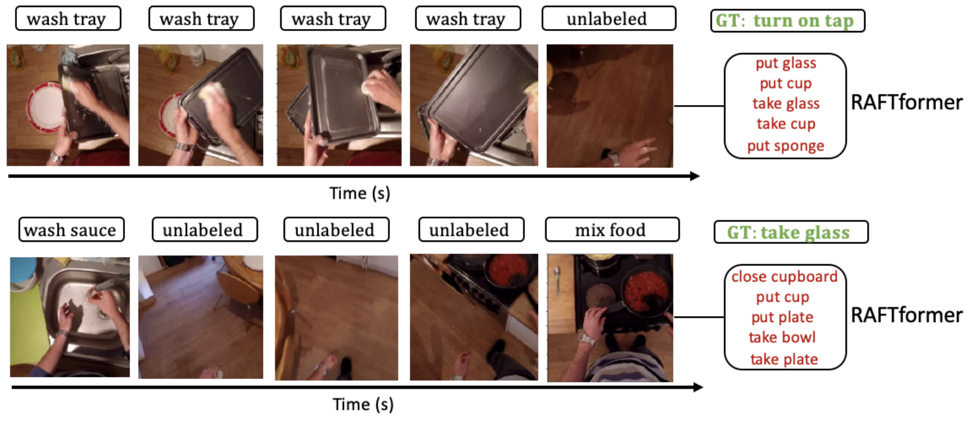


Figure 4.2: Two examples of failure cases of RAFTformer. We can see that there are some cases where even RAFTformer has a lot of trouble understanding context and what exactly is going on. In the first image, a tap is not clearly visible which is why RAFTformer may fail to predict actions that involve a tap.

$STD = [0.25, 0.25, 0.25]$ following the scheme in MViT [21]. Note that techniques such as image augmentations, random cropping, and multi-crop evaluation would boost performance during both inference and training. However, we do not perform augmentations on our data due to GPU limitations making our training recipe much faster in practice; training end-to-end with augmentations however would further increase our performance as well. For example, [32] uses 3-crop testing which significantly improves performance. Our proposed method does **not** require end-to-end training and is suitable for faster experimentation iterations and those with GPU constraints. We use an observed sequence length of 16-18 seconds which is longer than most previous works by 6+ seconds but half as long as MemViT's model [112]. This however helps our model to be lower latency than SOTA model MemViT.

Training Hyperparameters

We train our model with the AdamW optimizer with momentum 0.8 and weight decay of 0.001. We use a batch size of 512 and AdamW optimizer for training. We train for 75 epochs using a cosine scheduler with a warm-up of 30 epochs. Our base learning rate is 1e-4 and end learning rate is 8e-7. We use a dropout of 0.25 for the transformer and a dropout of 0.1 for both the inputs and MLP feature/action heads.

Chapter 5

Conclusion

So far, we have proposed Real-Time Action Forecasting Transformer (RAFTformer), a parsimonious two-stage fully transformer-based architecture consisting of a short-range video transformer backbone for feature extraction and a long-range head transformer encoder that temporally aggregates information on a longer horizon across multiple clips. We also introduce a real-time evaluation setting for action forecasting models that directly penalizes high latency and closely mimics the real-world deployment scenario for forecasting models. RAFTformer outperforms prior state-of-the-art methods by significant margins across several action forecasting benchmarks in the offline setting, and by even larger margin of upto 4.4 T5R, in the real-time setting. RAFTformer achieves 9$\times$ lower inference latency at the same forecasting fidelity, using 16$\times$ less training compute and 10$\times$ lesser trainable parameters than prior baselines. We hope this work pushes future works toward designing latency-aware action forecasting models.

5.1 Limitations

In this work we focus solely on the short-term anticipation setting. Our model as is would not work directly for rolling-out multiple steps into the future which could be interesting for longer-term planning [34]. Furthermore, reasoning about longer-term horizons raises more questions about different types of multimodality that require additional modeling. Another limitation is that our model is tested on egocentric videos of a single human acting in an environment. While this is indeed applicable for human-robot interaction and robot understanding, testing in even more complex interactive environments would be an interesting future direction.

5.2 Use Cases

You may be wondering: why is anticipating over such a short horizon of 1 second even useful? In highly reactive environments, autonomous agents need to quickly anticipate what may happen. This quick prediction is critical in deciding what action the autonomous agent should take during downstream planning. In the next section, we demonstrate how short-term action anticipation is useful for the task of trajectory prediction – a critical part of the autonomous vehicle stack.

Chapter 6

Short-Term Anticipation for Trajectory Prediction

Trajectory prediction is a critical task for autonomous agents such as self-driving cars. The autonomous stack is typically broken down into:

- **Perception.** Observation of the world (camera, LiDAR, etc.)
- **Prediction.** How various agents in the world may act in the near future
- **Planning.** What actions the autonomous agent should take (planned future trajectory)
- **Control.** Executing a planned trajectory through steering, throttling, and braking

In this section, we show how short-term actions (referred to as short-term intents) are helpful for human and vehicle trajectory prediction. We will dive into detail about the task of trajectory prediction, current state-of-the-art methods, typical datasets, and our contributions of utilizing short-term actions (short-term intents) for trajectory prediction. This section aims to provide a concrete application of Chapters 1-5 to a real-world scenario. Please keep in mind the intent and action are used synonymously hereinafter.

6.1 Introduction

Over the past few years, there has been extensive research into predicting the future trajectories of dynamic agents in scenes, such as pedestrians and vehicles. This is an incredibly important and challenging task for safety-critical applications such as autonomous vehicles or social robot navigation. While these methods have been significantly advanced over recent years, very few benchmarks specifically test if these models can accurately reason about key maneuvers such as sudden turns and lane changes of vehicles or pedestrians crossing the road. Traditional trajectory error metrics may not

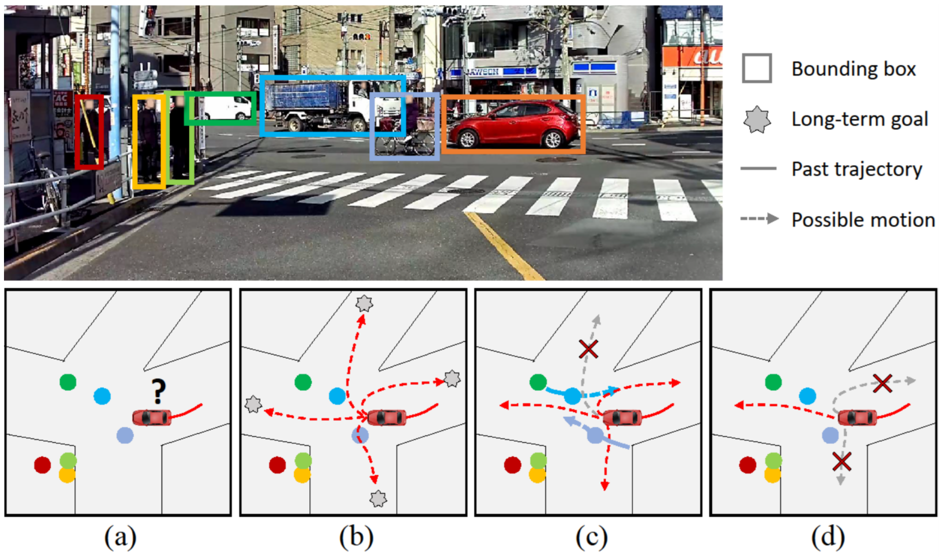


Figure 6.1: We show that reasoning about long-term goals and short-term intents plays a significant role in trajectory prediction. With a lack of comprehensive benchmarks for this purpose, we introduce a new dataset for intention and trajectory prediction. An example use case is illustrated in (a) where we predict the trajectory of the target vehicle. In (b), long-term goals are estimated from agent's own motion. Interactions in (c) and environmental constraints such as road topology and lane restrictions in (d) influence the agent's short-term intent and thus future trajectories.

capture performance on frame-level maneuvers, which is critical for safe planning.

An intelligent trajectory prediction system should be able to understand and model dynamic human behaviors. The study of human behavior as goal-directed entities has a long and rich interdisciplinary history across the subfields of psychology [10], neuroscience [106] and computer vision [77]. The human decision-making process is inherently hierarchical, consisting of several levels of reasoning and planning mechanisms that operate in tandem to achieve respective short and long term desires. Recent works have shown that explicitly reasoning about long-term goals [77, 15, 120] and short-term intents [71, 90, 11] can assist with trajectory prediction.

In this work, we propose to couple the tasks of heterogeneous (vehicles, pedestrians, etc.) multi-agent trajectory forecasting and intention prediction. We believe it is critical to explicitly reason about agents' long-term goals as well as their short-term intents. In our work, we define goals to be a final position an agent wants to reach for a given prediction horizon [78, 120], while intent refers to *how* an agent accomplishes their goal [87]. For example, consider a vehicle at an intersection. At the highest level, say they want to reach their ultimate goal of turning left to their final goal point, which in turn might be necessary for some higher-level end (such as going home). However, the exact motion of their trajectory is subject to many factors including i) agent's own will, ii) social interactions, iii) environmental constraints, iv) contextual cues. Thus, when reasoning about the agent's intent to turn left it is important to consider not only agent dynamics but also how intent is subject to change based on map topology or neighboring agents (see Figure 6.1). We believe this complex hierarchy of short-term intents and long-term goals is ubiquitous and in fact, crucial, for agent motion planning and hence by extension, for motion prediction. We propose an architecture that considers long-term goals similar to [78, 120, 77, 15] but adds a key component of frame-wise intention estimation which is used to condition the trajectory prediction module. By forcing the model to learn discrete short-term intents of agents, we observe improved performance by the prediction module.

Equally rich & successful is the contemporary history of the use of datasets for benchmarking progress in computer vision. Ushered by seminal works such as MNIST [62] and benchmarks such as ImageNet [57], benchmarking progress and learning from data has played a key role in the success of modern deep learning. Currently, there exists no public datasets that allow for explicit frame-wise intention prediction for heterogeneous agents in highly complex environments. Although few datasets are designed to study pedestrian intents or actions [88, 90, 71, 76] from egocentric view, it is an inherent limitation to extensive study of tasks for autonomous driving. Thus, we propose a joint trajectory and intention prediction dataset that contains RGB images with corresponding LiDAR point clouds with detailed, frame-wise labels for pedestrians *and* vehicles. The LOKI dataset allows explicit modeling of agents' future intent and extensive benchmarking for both tasks. It also shows promising directions to jointly reason about intentions and trajectories while considering different external factors such as agents' predilection, social interactions and environmental factors. We show that by modeling short-term intent and long-term goals with explicit supervision via intention labels, better trajectory prediction accuracy can be achieved. In addition, predicting a specific intention at each frame adds a layer of abstraction to our model that improves understanding prediction decisions, an important step towards maintaining safety-critical applications.

In conclusion, the contribution of our work is twofold. **First,** we propose the first publicly available heterogeneous dataset which contains frame-wise intention annotations and captures trajectories of up to 20 seconds containing both 2D and 3D labels with RGB and LiDAR inputs. **Second,** we illustrate the efficacy of separately reasoning about both long-term goals and short-term intents through ablation studies. Specifically, we highlight how the subtask of intention prediction improves prediction performance, and propose a model that outperforms state-of-the-art multimodal benchmarks by upto 27%. We believe our highly flexible dataset will allow the trajectory prediction community to further explore topics within the intention-based prediction space. In addition, the problem of intention estimation is an involved task in and of itself for which our work provides a strong baseline.

6.2 Related Work

Over the past few years, there has been a rapid improvement in the field of trajectory prediction owing to the success of deep neural networks and larger publicly available datasets [97, 90, 71, 88, 12, 9, 56, 93, 96]. There have been numerous subtopics of interest within the trajectory prediction community including compliant trajectory prediction, multi-modal trajectory prediction, and goal-oriented prediction [98, 36, 70, 77, 78, 60, 15, 3, 19, 55, 90, 120].

Contextual Trajectory Prediction

Earlier works in the field of trajectory prediction focused on unimodal trajectory prediction – predicting a single future path for each agent. These works underscored the importance of social [2, 3, 43] and scene compliance [114] when making predictions. Over the past few years, trajectory prediction studies have extended these ideas to multi-modal frameworks to account for multiple plausible futures each agent can have. In SocialGAN, Gupta et al. [36] introduce a socially-aware multi-modal framework that uses generative adversarial networks to sample a varying number of future trajectories for each agent. Since then, there has been a major emphasis and many interesting approaches to with multimodal forecasting [98, 63, 78, 36, 64, 15, 55, 99].

Goal-based Prediction

When modeling vehicle and human trajectories, it is natural to formulate the problem as a goal-directed task. Because humans are not completely stochastic agents and have a predilection towards certain actions, very recent trajectory forecasting studies have shown the effectiveness of goal-conditioned predictions [92, 19, 78, 77, 116, 90, 119, 15, 120, 18]. Recently, [78] and [120] showed that considering agents' final goal points can immensely aid in forecasting trajectories. However, both of these works only consider positional information as their goal states. In our work, we propose and show the effectiveness of considering *both* long-term positional goals as well as short-term intended actions.

Intention Datasets

To better understand agent intent in traffic scenes, a few works have proposed datasets that contain intention labels to study underlying intent in addition to the traditional trajectory prediction task. The JAAD [90], PIE [88] and STIP [71] datasets are recent datasets designed to study pedestrian intent. The JAAD dataset focuses on traffic scene analysis and behavior understanding of pedestrian at intersection crossing scenarios. The

PIE dataset expands on JAAD further and contains more annotations for both intention estimation and trajectory prediction. PIE [90] only predicts intent at the current timestep and focuses on shorter horizon predictions (1.5 seconds). The STIP dataset solves the limitation of only being able to do single-shot intention prediction, as it contains frame-wise intention labels for up to 3 seconds. However, this dataset only contains "crossing/not crossing" labels for pedestrians and does not focus on trajectory prediction. All these datasets only consider intentions of pedestrians at intersections which may not capture the intents of all agents in a highly complex traffic environment with both vehicles and pedestrians.

IntentNet [11] does consider intents for vehicle trajectory prediction; however, they do not consider frame-wise intentions. Furthermore, the dataset and labels are not publicly available. TITAN [76] is another driving action dataset collected from egocentric view. Although it can be potentially used for intention prediction of traffic agents, it only contains ego-view tracklets and lacks environmental and LiDAR information that can be crucial to find agents' intent. Both these works also only focus on short term predictions (less than 3 seconds).

To the best of our knowledge, currently no publicly available dataset contains detailed, frame-wise annotations to allow for heterogeneous multi-agent trajectory forecasting *and* intention prediction in joint camera and lidar space. Our dataset contains very diverse traffic scenarios through long data collection periods in different locations, weather conditions, roads and lighting. Table 6.1 shows the details of our LOKI dataset in comparison to other recently available intention datasets (PIE, JAAD, STIP).

6.3 LOKI Dataset

Exploring predictions in a large traffic environment is a complex problem because the future behavior of each traffic participant is not only indicated by the past behavior, but also highly impacted by the future goals and intentions. With a lack of comprehensive benchmarks for this purpose, we introduce a large scale dataset that is designed for the task of joint intention and trajectory prediction. Our dataset is collected from central Tokyo, Japan using an instrumented vehicle that is equipped with a camera (SEKONIX SF332X-10X), LiDAR (Velodyne VLP-32C), GPS and vehicle CAN BUS. The RGB camera and four LiDAR sensors are placed on top of the vehicle to obtain better environment coverage. The recordings are suburban and urban driving scenarios that contain diverse actions and interactions of heterogeneous agents, captured from different times of the day.

From our recordings, we extracted 644 scenarios with average 12.6 seconds length. The merged LiDAR data and synced RGB image were down sampled to 5HZ for annotation.

	PIE [90]	JAAD [88]	STIP [71]	LOKI (ours)
# of scenarios	-	346	556	**644**
# of agents	1.8K	2.8K	3.3k	**28K**
# of labeled agents	1.8K	0.6K	3.3	**28K**
# of classes	1	1	1	**8**
# of bboxes	740K	391K	350k	**886K**
# of agent types	1 (Ped)	1 (Ped)	1 (Ped)	**8 classes**
Avg. agent per frame	2.5	5.2	3.2	**21.6**
Annotation freq.	-	-	2 FPS	**5 FPS**
Frame-wise labels	no	✓	✓	✓
RGB Images	✓	✓	✓	✓
LiDAR Point cloud	no	no	no	✓
2D Bounding box	✓	✓	✓	✓
3D Bounding box	no	no	no	✓
Lane Info	no	no	no	✓
Pedestrian attributes	no	✓	no	✓

Table 6.1: Comparison of LOKI dataset with PIE [90], JAAD [88] and STIP [71].

The total number of agents is over 28K including 8 classes (*i.e.*, Pedestrian, Car, Bus, Truck, Van, Motorcyclist, Bicyclist, Other) of traffic agents, which results in 21.6 average agents in a scene. We annotated all these agents' bounding boxes (total 886K) in the RGB image (2D) as well as LiDAR point cloud space (3D) by linking with a same track-ID. The comparison with existing benchmarks is shown in Table 6.1. The LOKI dataset is annotated with unique attributes that can influence agents' intent such as interaction related labels, environmental constraints and contextual information.

Dataset Annotation

Considering that LiDAR point clouds better capture positional relations among agents than RGB images, we annotate 3D bounding box of agents with their orientation, potential destination of pedestrians, road entrance / exit, and agents' intention as well as action labels in this space. In contrast, in the RGB image space we leverage its contextual clarity to annotate environmental labels such as lane information (what actions can be made from this lane), lane number for vehicles (relative position with respect to the autonomous agent), the gender and age for pedestrian, the state of traffic light, and the type of traffic sign. Note that we also annotate 2D bounding box, potential destination and road entrance / exit information in the RGB space to inspire the potential research in the egocentric

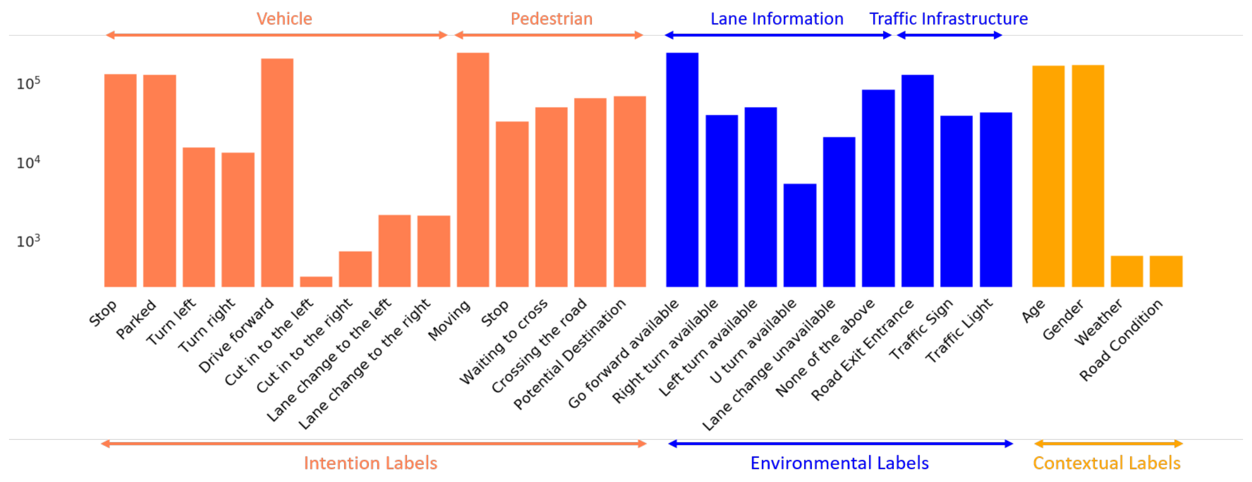


Figure 6.2: Distribution of labels sorted according to the different types of intention among the different classes

view. By using the consistent tracking ID between the same agent in the 3D LiDAR space and 2D image space, our labels can be shared across different spaces.

To dig into more complex prediction researches, our dataset provides denser agents per frame and more meticulous intention attributes compared to other datasets. We have three types of labels in the LOKI dataset: Intention labels, Environmental labels and Contextual labels to explore how these can affect the future behavior of agents (details and visuals are in Figure 6.2 and Figure 6.3).

Intention labels Intentions are defined to be "how" an actor decides to reach a goal via a series of actions [87]. At each frame, we annotated the current actions of the traffic participants and then used future actions to generate our intention labels. For example, if the current action of vehicle is "Moving" and the future action in 1 second is "Stopped", the vehicle's current intention is to stop. Various intention horizons can be explored; we use $0.8s$, as we explore how short-term intent can help guide trajectory prediction.

Environmental labels The environment of driving scene can heavily impact the intention of agent especially for the driving area users, so we include the environmental information such as "Road Exit and Entrance" positions, "Traffic light", "Traffic Sign", "Lane Information" in the LOKI dataset. Those labels determined by the structure of the road and the traffic rules that can be applied to any agent in the scene. The lane information includes the allowed actions of the current lane where the vehicle is on and the relative position between other vehicle and ego-vehicles.

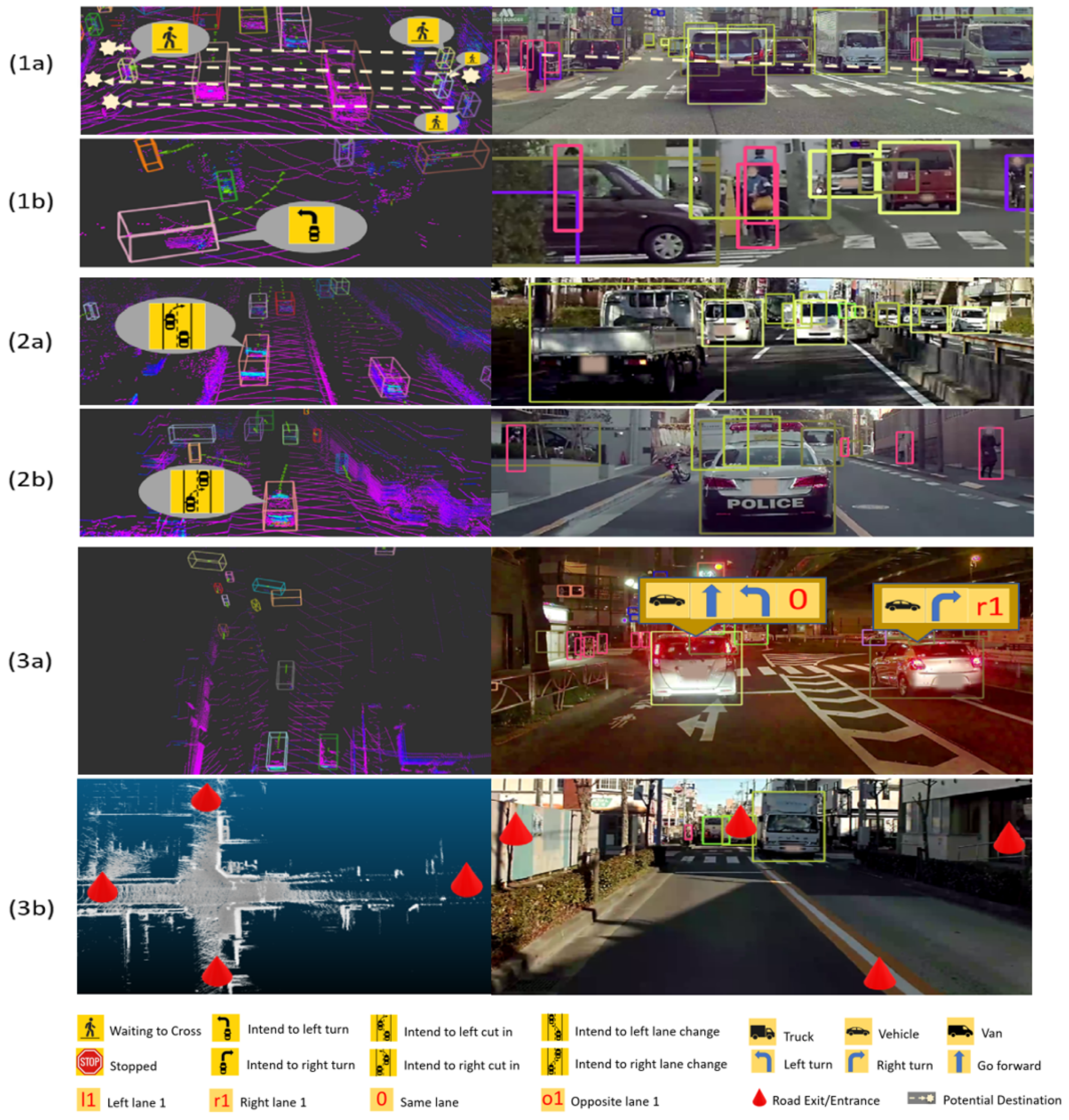


Figure 6.3: Visualization of three types of labels: (1a-1b) Intention labels for pedestrian; (2a-2b) Intention labels for vehicle; and (3a-3b) Environmental labels. The left part of each image is from laser scan and the right part is from RGB camera. In (1a), the current status of pedestrian is “Waiting to cross” and “Stop”, and the potential destination shows the intention of pedestrian. In (3a), the blue arrow indicates the possible action of the current lane where the vehicle is on, and the red words present the lane position related to the ego-vehicle.

Contextual labels There are some other factors may also affect the future behavior of agent. We define the "Weather", "Road condition", "Gender", "Age" as external contextual labels. These factors are the characters of the agent or environment which can cause the different intentions even under similar environment condition.

6.4 Proposed Method

Problem Formulation

In this work, we tackle the problem of multi-agent trajectory forecasting while concurrently predicting agent intentions. The type of intentions vary between agent classes: vehicles and pedestrians. We formulate the problem as follows. Suppose in a given scene, $\mathcal{S}$, we have N agents, $A_{1:N}$. Given the past $t_{obs} = 3s$ of trajectory history in BEV coordinates, the problem requires forecasting the future $t_{pred} = 5s$ coordinates of the agent in top-down image space. Since our dataset allows for frame-wise intention predictions depending on agent type (pedestrians vs. vehicles), we define another task to predict discrete intentions for each agent at each timestep, in addition to the traditional trajectory prediction problem.

Model Design

Long-term Goal Proposal Network

Intuitively, agents have a predetermined, long-term goal that they want to reach. Many recent goal-directed works have focused on modeling this through estimating final "endpoint" or "goal state" distributions as done in [78, 77, 120, 19, 15]. Inspired by agents' rational decision-making process and the success of prior works, we design a goal network similar to the method proposed in [78]. For each agent, A_k, we use a Conditional Variational Autoencoder (CVAE) to estimate the final long term goal G_k that is simply the estimated position in BEV $u_{k_f} = (x_{k_f}, y_{k_f})$ where f indicates the final frame. The inputs into the CVAE are the encodings from the Observation RNN Encoder. The goal network only consider agents' own history, as agents have a predetermined long term goal irrespective of other agents.

Scene Graph + Trajectory Decoder

Our main insight and promising directions from our proposed dataset comes from agents' short-term intentions. As described earlier, we have different intentions for pedestrians and vehicles. Without loss of generality, we will refer to agents, A, and intentions, I, without specifying the type of agent. We believe agents' have intermediate stochastic intents that can change depending on agent behavior, agent-agent interaction, or environmental factors. To account for this, we construct a traffic scene graph $\mathcal{G}$ to account for social and

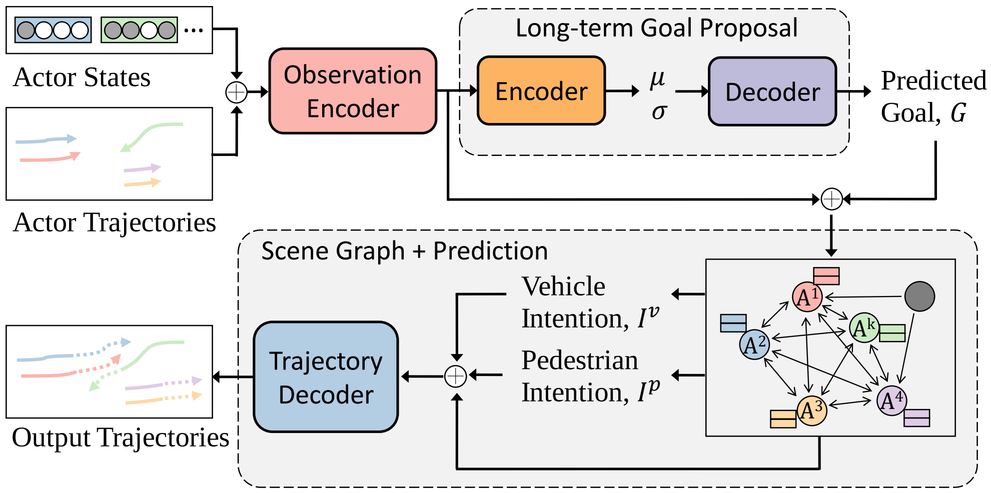


Figure 6.4: Our model first encodes past observation history of each agent to propose a long-term goal distribution over potential final destinations for each agent independently. A goal, G is then sampled and passed into the Joint Interaction and Prediction module. A scene graph is constructed to allow agents to share trajectory information, intentions, and long-term goals. Black nodes denote road entrance/exit information which provides agents with map topology information. At each timesteps, current scene information is propagated through the graph. We then predict an intent (what action will the agent take in the near future?) for each agent. Finally, the trajectory decoder is conditioned on predicted intentions, goals, past motion, and scene before forecasting the next position. This process is recurrently repeated for the horizon length.

environmental factors that may affect intent and trajectory prediction.

More concretely, suppose we have a scene graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where vertices, $\mathcal{V}$, denote agents and road entrances/exits and edges, $\mathcal{E}$, capture agent-agent and agent-map influence. In a given scene, for neighboring agents v_i and v_j, there is a directional edge e_{ij} if agent i affects agents j (within a certain distance threshold away). Road entrance/exit nodes can affect agents but have no incoming edges, as they are static. We connect a directional edge e_{ij} if road entrance/exit node i is within a certain distance from agent j.

We then predict agents' future locations via a daisy chained process described as follows. At each frame, m, our model first shares information between agents via the attention mechanism used in [102]:

$$x_i^{t+1} = \gamma(x_i^t) + \sum_{x_j \in \mathcal{N}(x_i)} \alpha_{ij} * \phi(x_j^t, e_{ij}),$$

where x_i^{t+1} represents the updated node features following attention-based feature aggregation with all of its neighbors $x_j \in \mathcal{N}(x_i)$. We use agents' velocities and relative positions as edge features. These features are encoded by a 2-layer MLP prior to message passing at each timestep. We use the scaled dot-product attention [102] formulation:

$$a_{ij} = softmax(\frac{\psi(x_i)^T \xi(x_j, e_{ij})}{\sqrt{d}})$$

Here, a_{ij} represents the attention coefficient between two nodes i and j and d represents the degree of the node. We use a single-layer for ϕ, γ, ψ, and ξ.

After message passing which allows agents to share their past trajectory, goal, and intention information along with road information through the road entrance/exit nodes, our model then predicts agent intent, which we define to be the agent's future action $m+q$ frames ahead. In our experiments, we set $q = 4$, thus predicting short-term intent $0.8s$ in the future. We then condition trajectory prediction for frame $m+1$ based on agent intent at frame m. This process of information sharing and intention conditioning is recurrently repeated for the next $f - ob$ timesteps where f denotes the last prediction frame number and ob denotes the last observation frame. Formally, at each frame, m, we first estimate the probability distribution over a discrete set of intentions (different set of intentions for pedestrian vs. vehicle) for an agent, A_i:

$$\begin{aligned} P(I_{i_m} | I_{i_{ob:m-1}}, U_{i_{0:m-1}}, G_i, a_{i_{0:ob}}, \cup_{A_j \in \mathcal{N}(A_i)} I_{j_{ob:m-1}}, \\ U_{j_{0:m-1}}, G_j, a_{j_{0:ob}}, R_{ee}) \end{aligned}$$

where I refers to intention, U is position, G is long-term positional goal, a is action, and R_{ee} refers to road entrances/exit labels. The intention networks are two-layer MLPs which predicts intention using each actor's updated hidden states from the most recent message passing. Following this, we then predict the next position of each agent, U, conditioned as follows:

$$\begin{aligned} P(U_{i_{m+1}} | I_{i_{o:m}}, U_{i_{0:m}}, G_i, a_{i_{0:ob}}, \cup_{A_j \in \mathcal{N}(A_i)} I_{j_{o:m}}, \\ U_{j_{0:m}}, G_j, a_{j_{0:ob}}, R_{ee}) \end{aligned}$$

The trajectory decoder module consists of a GRU that updates each actor's current hidden state followed by a 2-layer MLP used to predict positions at each step. The overview of our model is illustrated in Figure 6.4. Specific model architecture details will be provided in the supplementary material.

Loss Functions

Our goal proposal network (GPN) follows the methodology introduced in [78] and is trained via the following loss function:

$$\mathcal{L}_{GPN} = \alpha_1 D_{KL}(\mathcal{N}(\mu, \sigma) \| \mathcal{N}(0, I)) + \alpha_2 \|\hat{G} - G\|_2^2$$

Here α_1 and α_2 are tunable parameters to weight the KL Divergence loss and goal reconstruction loss for training the CVAE. Note that because the intentions are dependent on the long-term goal, we observed that training via conditioning with ground-truth goal positions helps with model convergence.

Our decoder module which is responsible for both intention and trajectory prediction is composed of separate loss terms for each. Our intention loss is defined as follows:

$$\mathcal{L}_{int} = -\sum_{i=0}^{n} w_i * y_i * log(\hat{y_i})$$

Due to heavy class imbalance, we not only augment rare trajectories such as lane changes and turning but also weight the cross entropy loss by w_i, which is the inverse frequency of the class.

Since we predict offsets in position (velocity) rather than position directly for better model convergence, our loss is on the predicted velocity V for all timesteps:

$$\mathcal{L}_{traj} = ||V - \hat{V}||_2$$

We train our network end-to-end by weighting each of the loss terms:

$$\mathcal{L}_{Final} = \lambda_1 \mathcal{L}_{GPN} + \lambda_2 \mathcal{L}_{int} + \lambda_3 \mathcal{L}_{traj}$$

Evaluation Metrics

For trajectory prediction evaluation, we use the standard Average Displacement Error (ADE) and Final Displacement Error (FDE) metrics:

$$ADE = \frac{\sum_{j=t_{ob}+1}^{t_f} \|\hat{u}_j - u_j\|_2}{(t_f - t_{ob})} \tag{6.1}$$

$$FDE = \|\hat{u}_{t_f} - u_{t_f}\|_2 \tag{6.2}$$

where $\hat{u}$ and u are the estimated and ground truth positions respectively. Furthermore, we use the *minADE-N* and *minFDE-N* error metric introduced in [36] for multimodal evaluation. The metric is simply the minimum ADE and FDE out of N future trajectories predicted at test-time.

For intention prediction, we evaluate frame-wise classification accuracy of intents and visualize the confusion matrix to analyze classification performance.

	S-STGCNN		PECNet		Ours		Ours + IC		Ours + IC + SG	
	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE
Pedestrians	0.96	1.98	0.79	1.31	0.61	1.38	0.56	1.24	**0.55**	**1.21**
Vehicles	3.03	7.01	2.52	6.34	2.37	6.20	**2.23**	**5.80**	2.24	5.82
Lane Change	4.41	10.17	2.78	7.60	2.93	7.88	**2.47**	6.78	2.52	**6.71**
Turn	3.48	8.15	2.97	7.44	2.76	7.26	2.69	7.03	**2.69**	**7.02**

Table 6.2: **Trajectory error metrics for N=1 samples**: ADE and FDE of various state-of-the-art baselines and our method using unimodal (single-shot) evaluation. Reported errors are in meters. Lower is better. We show results evaluated on separate classes to gain more insight on prediction performance. We report errors on 1) pedestrians, 2) vehicles (non-static), 3) agents that change lanes, and 4) agents that turn.

	S-GAN		S-STGCNN		PECNet		Ours		Ours + IC		Ours + IC + SG	
	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE
Pedestrians	1.04	2.18	0.63	1.01	0.51	0.70	0.36	0.70	0.37	0.71	**0.34**	**0.64**
Vehicles	3.57	8.05	2.28	4.46	1.59	3.05	1.33	3.09	1.20	2.63	**1.18**	**2.64**
Lane Change	3.50	8.41	3.00	6.09	1.62	2.85	1.42	3.30	1.26	2.70	**1.22**	**2.71**
Turn	3.75	9.01	2.68	5.71	1.96	4.07	1.54	3.59	1.45	3.24	**1.40**	**3.13**

Table 6.3: **Trajectory error metrics for N=20 samples**: ADE and FDE of various state-of-the-art baselines and our method using multimodal evaluation. Reported errors are in meters. Lower is better. We report errors on the same classes described in Table 6.2.

6.5 Experiments

In this section, we present results of our model on trajectory & intent prediction tasks and demonstrate a superior performance against prior state-of-the-art baselines (with publicly available code) across a variety of settings. We benchmark against PECNet [77], a strong scene agnostic trajectory prediction method with state-of-the-art performance on standard intention agnostic prediction datasets. S-STGCNN [81] and S-GAN [36] are strong socially-aware models that achieved prior state-of-the-art on various benchmarks. We also report an interesting ablation on the effect of annotation frequency on the final performance, which confirms our hypothesis for the effectiveness of detailed intent annotations in trajectory prediction.

Trajectory Prediction Performance. We report our model's performance and benchmark it against prior state-of-the-art models for unimodal (single shot, N = 1) prediction in Table 6.2 and for multimodal predictions (N = 20 shots) in Table 6.3. Our ablations are with Ours (without action/intention labels), IC (with action/intention labels for intention conditioning), SG (with scene graph for social reasoning and environmental cues).

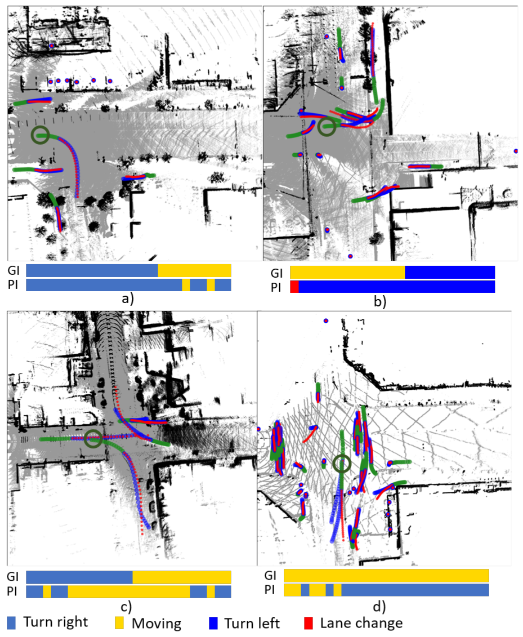
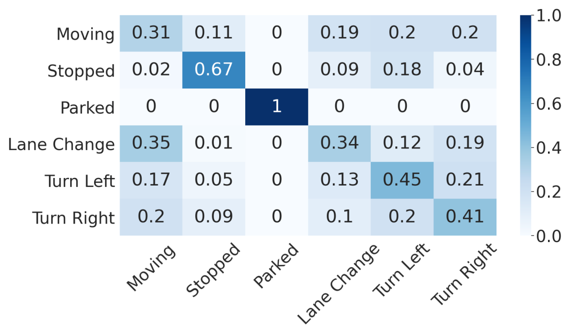


Figure 6.5: Visualization of top-1 trajectory prediction result (green: past observation, blue: ground truth, red: prediction) and frame-wise intention of a particular agent in dark green circle at the start of the observation time step(GI: Ground truth Intention, PI: Predicted Intention) is shown at the bottom of each scenario. More detailed visualizations and comparisons are provided in supplementary material.

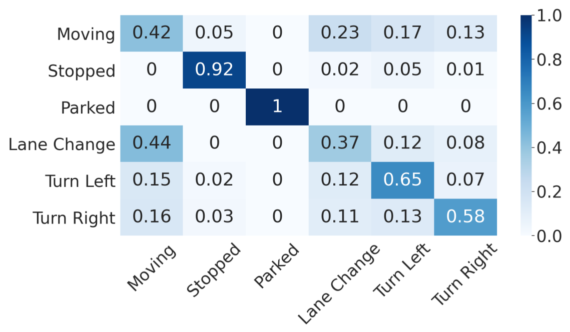
Several interesting trends emerge. First, we observe that in the single shot setting, our intention conditioned model outperforms prior state-of-the-art method by a significant margin of 12% in ADE, 9% in FDE. Second, we see a similar trend in multi-shot prediction setting as well with our model outperforming PECNet by 33% in ADE and 9% in FDE for pedestrians and a delta of 26% in ADE and 13% in FDE for moving vehicles. Third, notice that the performance gap is significant in hard non-linear cases such as lane changes and turns, where our model achieves 30% and 16% better performance in ADE and FDE respectively.

Also noteworthy is the crucial effect of conditioning predictions on intentions and incorporating social and environmental cues through the scene graph, which is also shown in Table 6.2 and Table 6.3. We note that both intention cues and scene graph information are critical to overall performance, with intention improving ADE performance by upto 7% and 8% across all agent types (especially nonlinear trajectories such as lane changes and turns) for the unimodal and multimodal settings. We notice that the scene graph boosts performance by 3% in ADE for the multimodal setting across all agent types.

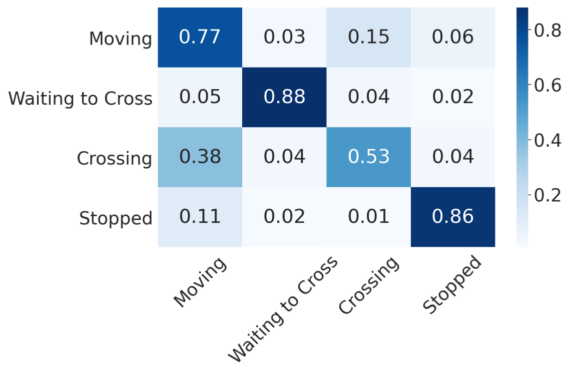
We notice an interesting behavior with pedestrians. Conditioning on pedestrian intent such



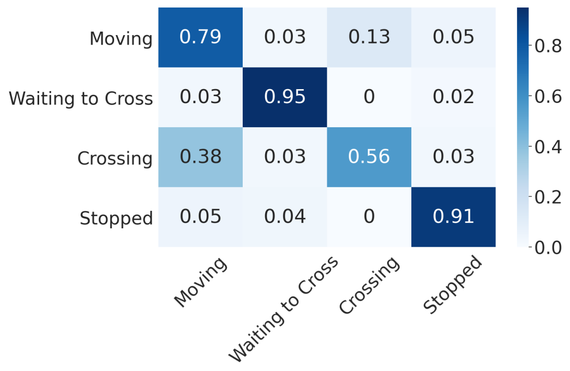
(a) Vehicle (N=1)



(b) Vehicle (N=20)



(c) Pedestrian (N=1)



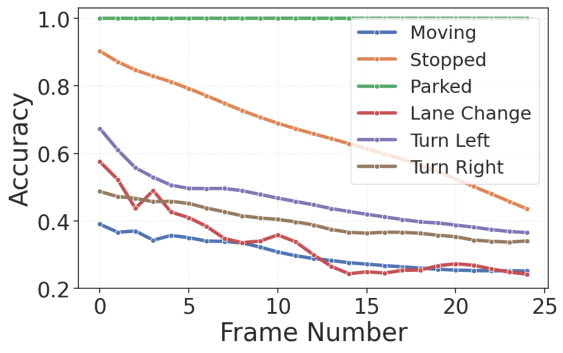
(d) Pedestrian (N=20)

Figure 6.6: Intention prediction confusion matrices. (a-b) results for vehicles under both unimodal and multimodal sampling, (c-d) those for pedestrians.

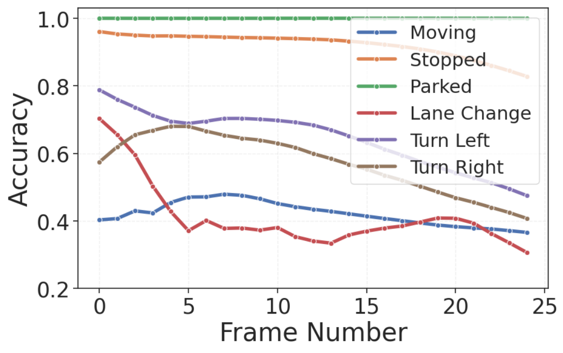
as crossing vs. waiting to cross helps for single-shot prediction as shown in Table 6.2. However, we do not see a benefit for multimodal prediction. We hypothesize that this is because the type of intent we label for pedestrian is not as granular as for vehicles in that it does not change drastically frame-by-frame. This is validated in Figure 6.8 which shows experiments with downsampled intention annotations. We observe that for pedestrians, lower frequency annotations does not diminish performance as compared to vehicles. Because pedestrians have more unconstrained behavior, we cannot have as detailed intent labels that are used for vehicles such as turn left or lane change. This may explain the behavior of why intention conditioning only helps for the single-shot case for pedestrians.

In Figure 6.5, we visualize our model's best-of-20 performance. We observe that predicted trajectories are fairly accurate and with underlying turning intentions. While there are limitations in exact frame-wise intention predictions, we notice it can capture key future actions of turning and can help guide predictions.

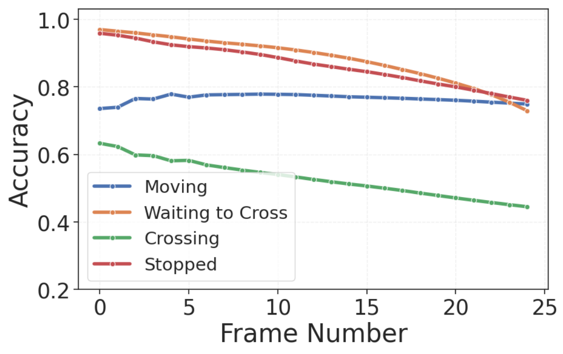
Intention Prediction: In addition to trajectory prediction, our dataset enables for a



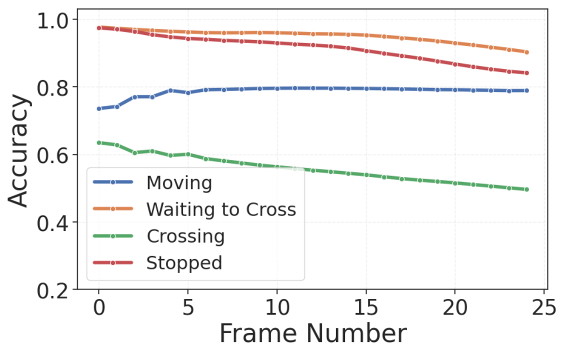
(a) Vehicle (N=1)



(b) Vehicle (N=20)



(c) Pedestrian (N=1)



(d) Pedestrian (N=20)

Figure 6.7: **Accuracy vs. Future Horizon (in frames).** The x axis of each figure is time and the y axis of each figure is accuracy (from 0 to 1). The change of intention prediction accuracy over a time horizon for both unimodal and multimodal predictions. In (a-b) we plot intention accuracy over time for vehicles for N=1 and N=20 samples respectively. In (c-d) we plot intention accuracy over time for pedestrians with N=1 and N=20 samples.

more high level understanding of agent intent to mimic how they plan their trajectory. Figure 6.7 illustrates the performance of intention prediction over a 25 frame (5s) prediction horizon. Our work is the first to baseline both pedestrian and vehicle intent on a frame-wise level. We notice that prediction performance monotonically worsens over the horizon. However, we notice that for vehicles the intention accuracy in the multimodal setting is significantly improved from the unimodal case. This explains why intention conditioning helps more in the multimodal case, as agent intents are much more accurately understood. In contrast, for pedestrians, we notice only a slight improvement in intention performance. We posit this is because the intents for pedestrians do not change as frequently and are not as granular capturing directions such as "turn left"; thus, having more samples does not necessarily increase performance.

To better understand intention estimation, we visualize the confusion matrices as shown in

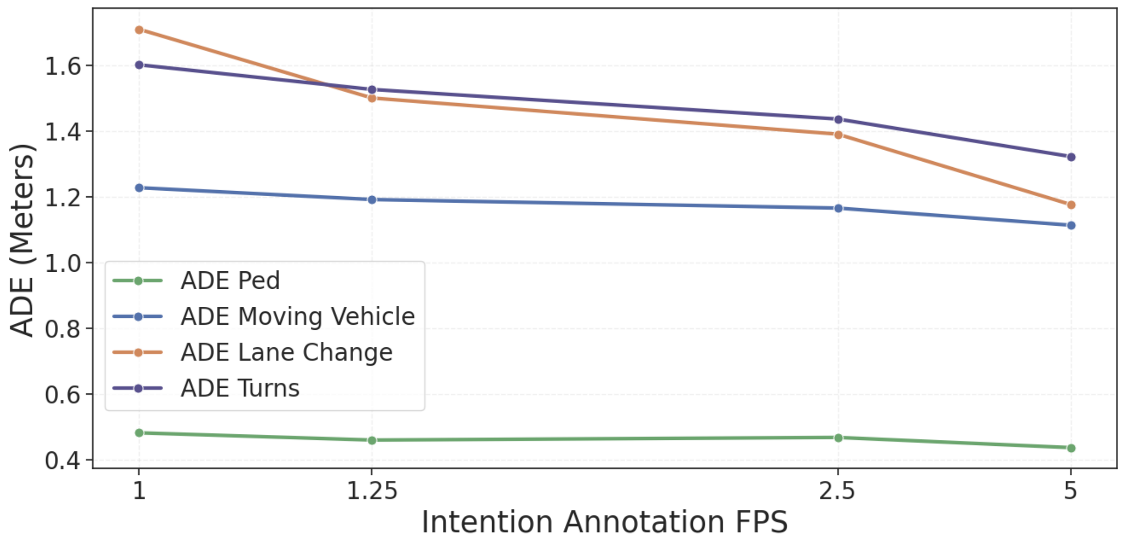


Figure 6.8: ADE Performance based on varying ground-truth intention annotation frequency.

Figure 6.6. For vehicles, we use the following set of discrete actions: *moving*, *stopped*, *parking*, *lane change*, *turn left*, and *turn right*. We observed improved performance for vehicle intention prediction with multimodal goal destination sampling, indicating that our model can correlate long-term goals with short-term intent. For pedestrians, we use *moving*, *waiting to cross*, *crossing*, and *stopped*. The intents for pedestrians do not rapidly change unlike those for vehicles. Thus, we see that multimodal predictions do not actually improve pedestrian intention estimation. These results corroborate the results in Table 6.3 where multimodal predictions with intention fail to outperform predictions without intentions. This is further examined in the next section.

Effect of Annotation Frequency: Our dataset provides very detailed frame-wise intention labels at 5FPS for all agents. To examine the importance of having a dataset with such detailed annotations, we experiment with how changing annotation frequency can affect performance. We provide our model with oracle intentions available at varying frequencies. As shown in Figure 6.8, trajectory prediction performance worsens roughly linearly as the frequency of intention labels reduces. This highlights the importance of our highly detailed annotations, as a choice to annotate every other frame (2.5fps) clearly affects performance. Note that this effect is witnessed for primarily for vehicles, especially those that change lanes or turn. Pedestrian performance is not affected much, as the intention labels used for pedestrians do not change drastically on a frame-by-frame basis. This is also the reason why intention conditioning did not help for multimodal evaluation for pedestrians as seen in Table 6.3.

6.6 Conclusion

In this chapter, we presented a large-scale heterogeneous dataset with detailed, frame-wise intention annotations. This dataset allows for both traditional trajectory prediction as well as understanding how intent changes over a long time horizon. In doing so, this dataset is the first that can be used as a benchmark for intention understanding for both vehicles and pedestrians. Furthermore, we formulate a joint trajectory and intention prediction framework which outperforms the state-of-the-art on trajectory prediction metrics and offers a strong baseline for intention prediction. We bridge the gap between trajectory prediction and intention prediction and show that combining the two can better model agents' decision-making process, assisting in trajectory prediction. We believe our dataset can inspire future works that consider intention prediction in addition to traditional trajectory forecasting. Doing so can give more insight into models' decisions and will be critical in designing and maintaining a safe forecasting system.

Bibliography

[1] Yazan Abu Farha, Alexander Richard, and Juergen Gall. “When will you do what?-anticipating temporal occurrences of activities”. In: *CVPR*. 2018, pp. 5343–5352.

[2] Alexandre Alahi, Vignesh Ramanathan, and Li Fei-Fei. “Socially-aware large-scale crowd forecasting”. In: *CVPR*. 2014, pp. 2203–2210.

[3] Alexandre Alahi et al. “Social lstm: Human trajectory prediction in crowded spaces”. In: *CVPR*. 2016, pp. 961–971.

[4] Georges Aoude et al. “Mobile agent trajectory prediction using Bayesian nonparametric reachability trees”. In: *Infotech@ Aerospace 2011*. 2011, p. 1512.

[5] Relja Arandjelovic and Andrew Zisserman. “Look, listen and learn”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 609–617.

[6] Anurag Arnab et al. “Vivit: A video vision transformer”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 6836–6846.

[7] Yuki M Asano, Christian Rupprecht, and Andrea Vedaldi. “A critical analysis of self-supervision, or what we can learn from a single image”. In: *arXiv preprint arXiv:1904.13132* (2019).

[8] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling”. In: *arXiv preprint arXiv:1803.01271* (2018).

[9] Holger Caesar et al. “nuscenes: A multimodal dataset for autonomous driving”. In: *CVPR*. 2020, pp. 11621–11631.

[10] Susan Carey and Elizabeth Spelke. “Domain-specific knowledge and conceptual change”. In: *Mapping the mind: Domain specificity in cognition and culture* 169 (1994), p. 200.

[11] Sergio Casas, Wenjie Luo, and Raquel Urtasun. “Intentnet: Learning to predict intention from raw sensor data”. In: *CORL*. PMLR. 2018, pp. 947–956.

[12] Ming-Fang Chang et al. “Argoverse: 3d tracking and forecasting with rich maps”. In: *CVPR*. 2019, pp. 8748–8757.

[13] Sayantan Chatterjee, Faheem H Zunjani, and Gora C Nandi. "Real-time object detection and recognition on low-compute humanoid robots using deep learning". In: *2020 6th International Conference on Control, Automation and Robotics (ICCAR)*. IEEE. 2020, pp. 202–208.

[14] Hsu-kuang Chiu, Ehsan Adeli, and Juan Carlos Niebles. "Segmenting the future". In: *IEEE Robotics and Automation Letters* 5.3 (2020), pp. 4202–4209.

[15] Chiho Choi et al. "DROGON: A Trajectory Prediction Model based on Intention-Conditioned Behavior Reasoning". In: *Proceedings of the CORL*. 2020.

[16] Dima Damen et al. "Scaling egocentric vision: The epic-kitchens dataset". In: *ECCV*. 2018, pp. 720–736.

[17] Roeland De Geest and Tinne Tuytelaars. "Modeling temporal structure with lstm for online action detection". In: *WACV*. IEEE. 2018, pp. 1549–1557.

[18] Patrick Dendorfer, Aljosa Osep, and Laura Leal-Taixé. "Goal-GAN: Multimodal Trajectory Prediction Based on Goal Position Estimation". In: *ACCV*. 2020.

[19] Nachiket Deo and Mohan M Trivedi. "Trajectory forecasts in unknown environments conditioned on grid-based plans". In: *arXiv preprint arXiv:2001.00735* (2020).

[20] Alexey Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[21] Haoqi Fan et al. "Multiscale Vision Transformers". In: *arXiv preprint arXiv:2104.11227* (2021).

[22] Mingyuan Fan et al. "Rethinking BiSeNet for real-time semantic segmentation". In: *CVPR*. 2021, pp. 9716–9725.

[23] Christoph Feichtenhofer et al. "Masked Autoencoders As Spatiotemporal Learners". In: *arXiv preprint arXiv:2205.09113* (2022).

[24] Christoph Feichtenhofer et al. "Slowfast networks for video recognition". In: *ICCV*. 2019, pp. 6202–6211.

[25] Basura Fernando et al. "Self-supervised video representation learning with odd-one-out networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 3636–3645.

[26] R James Firby. "An investigation into reactive planning in complex domains." In: *AAAI*. Vol. 87. 1987, pp. 202–206.

[27] Antonino Furnari, Sebastiano Battiato, and Giovanni Maria Farinella. "Leveraging uncertainty to rethink loss functions and evaluation measures for egocentric action anticipation". In: *ECCV*. 2018, pp. 0–0.

[28] Antonino Furnari and Giovanni Maria Farinella. "Rolling-unrolling lstms for action anticipation from first-person video". In: *IEEE transactions on pattern analysis and machine intelligence* 43.11 (2020), pp. 4021–4036.

[29] Jiyang Gao, Zhenheng Yang, and Ram Nevatia. “Red: Reinforced encoder-decoder networks for action anticipation”. In: *arXiv preprint arXiv:1707.04818* (2017).

[30] Harshayu Girase et al. “Loki: Long term and key intentions for trajectory prediction”. In: *ICCV*. 2021, pp. 9803–9812.

[31] Harshayu Girase et al. “Physically Feasible Vehicle Trajectory Prediction”. In: *arXiv preprint arXiv:2104.14679* (2021).

[32] Rohit Girdhar and Kristen Grauman. “Anticipative video transformer”. In: *ICCV*. 2021, pp. 13505–13515.

[33] Rohit Girdhar et al. “Video action transformer network”. In: *CVPR*. 2019, pp. 244–253.

[34] Dayoung Gong et al. “Future Transformer for Long-term Action Anticipation”. In: *CVPR*. 2022, pp. 3052–3061.

[35] Matthias Grundmann et al. “Efficient hierarchical graph-based video segmentation”. In: *2010 ieee computer society conference on computer vision and pattern recognition*. IEEE. 2010, pp. 2141–2148.

[36] Agrim Gupta et al. “Social gan: Socially acceptable trajectories with generative adversarial networks”. In: *CVPR*. 2018, pp. 2255–2264.

[37] Tengda Han, Weidi Xie, and Andrew Zisserman. “Memory-augmented dense predictive coding for video representation learning”. In: *European conference on computer vision*. Springer. 2020, pp. 312–329.

[38] Tengda Han, Weidi Xie, and Andrew Zisserman. “Video representation learning by dense predictive coding”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019, pp. 0–0.

[39] Mark Handley. “Delay is not an option: Low latency routing in space”. In: *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*. 2018, pp. 85–91.

[40] Harish Haresamudram et al. “Masked reconstruction based self-supervision for human activity recognition”. In: *Proceedings of the 2020 international symposium on wearable computers*. 2020, pp. 45–49.

[41] Joel Hasbrouck and Gideon Saar. “Low-latency trading”. In: *Journal of Financial Markets* 16.4 (2013), pp. 646–679.

[42] Kaiming He et al. “Masked autoencoders are scalable vision learners”. In: *CVPR*. 2022, pp. 16000–16009.

[43] Dirk Helbing and Peter Molnar. “Social force model for pedestrian dynamics”. In: *Physical review E* 51.5 (1995), p. 4282.

[44] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[45] Ashish Jaiswal et al. “A survey on contrastive self-supervised learning”. In: *Technologies* 9.1 (2020), p. 2.

[46] Dinesh Jayaraman and Kristen Grauman. “Learning image representations tied to ego-motion”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1413–1421.

[47] Dinesh Jayaraman and Kristen Grauman. “Slow and steady feature analysis: higher order temporal coherence in video”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 3852–3861.

[48] Eugen Käfer et al. “Recognition of situation classes at road intersections”. In: *2010 IEEE International Conference on Robotics and Automation*. IEEE. 2010, pp. 3960–3965.

[49] Jared Kaplan et al. “Scaling laws for neural language models”. In: *arXiv preprint arXiv:2001.08361* (2020).

[50] Shian-Ru Ke et al. “A review on video-based human activity recognition”. In: *Computers* 2.2 (2013), pp. 88–131.

[51] Chanho Kim et al. “Discriminative appearance modeling with multi-track pooling for real-time multi-object tracking”. In: *CVPR*. 2021, pp. 9553–9562.

[52] Dahun Kim, Donghyeon Cho, and In So Kweon. “Self-supervised video representation learning with space-time cubic puzzles”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 8545–8552.

[53] Irena Koprinska and Sergio Carrato. “Temporal video segmentation: A survey”. In: *Signal processing: Image communication* 16.5 (2001), pp. 477–500.

[54] Bruno Korbar, Du Tran, and Lorenzo Torresani. “Cooperative learning of audio and video models from self-supervised synchronization”. In: *Advances in Neural Information Processing Systems* 31 (2018).

[55] Vineet Kosaraju et al. “Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks”. In: *arXiv preprint arXiv:1907.03395* (2019).

[56] Parth Kothari, Sven Kreiss, and Alexandre Alahi. “Human trajectory forecasting in crowds: A deep learning perspective”. In: *arXiv preprint arXiv:2007.03639* (2020).

[57] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[58] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[59] Hilde Kuehne, Ali Arslan, and Thomas Serre. “The language of actions: Recovering the syntax and semantics of goal-directed human activities”. In: *CVPR*. 2014, pp. 780–787.

[60] Sumit Kumar et al. "Interaction-Based Trajectory Prediction Over a Hybrid Traffic Graph". In: *arXiv preprint arXiv:2009.12916* (2020).

[61] Colin Lea, René Vidal, and Gregory D Hager. "Learning convolutional action primitives for fine-grained action recognition". In: *ICRA*. IEEE. 2016, pp. 1642–1649.

[62] Yann LeCun. "The MNIST database of handwritten digits". In: *http://yann. lecun. com/exdb/mnist/* ().

[63] Namhoon Lee et al. "DESIRE: Distant future prediction in dynamic scenes with interacting agents". In: *CVPR*. 2017, pp. 336–345.

[64] Jiachen Li, Hengbo Ma, and Masayoshi Tomizuka. "Conditional Generative Neural System for Probabilistic Trajectory Prediction". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 6150–6156.

[65] Jiangtong Li et al. "Video Semantic Segmentation via Sparse Temporal Transformer". In: *Proceedings of the 29th ACM International Conference on Multimedia*. 2021, pp. 59–68.

[66] Yanghao Li et al. "Improved multiscale vision transformers for classification and detection". In: *arXiv preprint arXiv:2112.01526* (2021).

[67] Yin Li, Miao Liu, and Jame Rehg. "In the eye of the beholder: Gaze and actions in first person video". In: *IEEE transactions on pattern analysis and machine intelligence* (2021).

[68] Tsung-Yi Lin et al. "Focal loss for dense object detection". In: *ICCV*. 2017, pp. 2980–2988.

[69] Tsung-Yi Lin et al. "Focal loss for dense object detection". In: *ICCV*. 2017, pp. 2980–2988.

[70] Matteo Lisotto, Pasquale Coscia, and Lamberto Ballan. "Social and scene-aware trajectory prediction in crowded spaces". In: *ICCV Workshops*. 2019, pp. 0–0.

[71] Bingbin Liu et al. *Spatiotemporal Relationship Reasoning for Pedestrian Intent Prediction*. 2020. arXiv: `2002.08945 [cs.CV]`.

[72] Wei Liu et al. "Ssd: Single shot multibox detector". In: *ECCV*. Springer. 2016, pp. 21–37.

[73] Ze Liu et al. "Video swin transformer". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 3202–3211.

[74] Zhihua Liu et al. "A COVID-19 epidemic model with latency period". In: *Infectious Disease Modelling* 5 (2020), pp. 323–337.

[75] Tahmida Mahmud, Mahmudul Hasan, and Amit K Roy-Chowdhury. "Joint prediction of activity labels and starting times in untrimmed videos". In: *ICCV*. 2017, pp. 5773–5782.

[76] Srikanth Malla, Behzad Dariush, and Chiho Choi. “Titan: Future forecast using action priors”. In: *CVPR*. 2020, pp. 11186–11196.

[77] Karttikeya Mangalam et al. “From Goals, Waypoints & Paths To Long Term Human Trajectory Forecasting”. In: *arXiv preprint arXiv:2012.01526* (2020).

[78] Karttikeya Mangalam et al. “It is not the journey but the destination: Endpoint conditioned trajectory prediction”. In: *ECCV*. Springer. 2020, pp. 759–776.

[79] RJW Mansfield. “Latency functions in human vision”. In: *Vision research* 13.12 (1973), pp. 2219–2234.

[80] Peter McLeod. “Visual reaction time and high-speed ball games”. In: *Perception* 16.1 (1987), pp. 49–59.

[81] Abduallah Mohamed et al. “Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction”. In: *CVPR*. 2020, pp. 14424–14432.

[82] Daniel Neimark et al. “Video transformer network”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 3163–3172.

[83] Kriti Ohri and Mukesh Kumar. “Review on self-supervised image recognition using deep neural networks”. In: *Knowledge-Based Systems* 224 (2021), p. 107090.

[84] Venkata N Padmanabhan and Jeffrey C Mogul. “Improving HTTP latency”. In: *Computer Networks and ISDN Systems* 28.1-2 (1995), pp. 25–35.

[85] Bo Pang et al. “Deep rnn framework for visual sequential applications”. In: *CVPR*. 2019, pp. 423–432.

[86] David A Patterson. “Latency lags bandwith”. In: *Communications of the ACM* 47.10 (2004), pp. 71–75.

[87] Amir Rasouli. “Pedestrian Simulation: A Review”. In: *arXiv preprint arXiv:2102.03289* (2021).

[88] Amir Rasouli, Iuliia Kotseruba, and John K Tsotsos. “Are they going to cross? a benchmark dataset and baseline for pedestrian crosswalk behavior”. In: *ICCV Workshops*. 2017, pp. 206–213.

[89] Amir Rasouli, Iuliia Kotseruba, and John K Tsotsos. “Pedestrian action anticipation using contextual feature fusion in stacked rnns”. In: *arXiv preprint arXiv:2005.06582* (2020).

[90] Amir Rasouli et al. “Pie: A large-scale dataset and models for pedestrian intention estimation and trajectory prediction”. In: *ICCV*. 2019, pp. 6262–6271.

[91] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. In: *CVPR*. 2016, pp. 779–788.

[92] Nicholas Rhinehart et al. “PRECOG: PREdiction Conditioned On Goals in Visual Multi-Agent Settings”. In: *arXiv preprint arXiv:1905.01296* (2019).

[93] A Robicquet et al. “Learning social etiquette: Human trajectory prediction in crowded scenes”. In: *ECCV (ECCV)*. 2020.

[94] Cristian Rodriguez, Basura Fernando, and Hongdong Li. “Action anticipation by predicting future dynamic images”. In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018, pp. 0–0.

[95] Debaditya Roy and Basura Fernando. “Predicting the Next Action by Modeling the Abstract Goal”. In: *arXiv preprint arXiv:2209.05044* (2022).

[96] Andrey Rudenko et al. “Human Motion Trajectory Prediction: A Survey”. In: *arXiv e-prints*, arXiv:1905.06113 (2019). arXiv: `1905.06113`.

[97] Andrey Rudenko et al. “Human motion trajectory prediction: A survey”. In: *The International Journal of Robotics Research* 39.8 (2020), pp. 895–935.

[98] Amir Sadeghian et al. “Sophie: An attentive gan for predicting paths compliant to social and physical constraints”. In: *CVPR*. 2019, pp. 1349–1358.

[99] Tim Salzmann et al. “Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data”. In: *arXiv preprint arXiv:2001.03093* (2020).

[100] Fadime Sener, Dipika Singhania, and Angela Yao. “Temporal aggregate representations for long-range video understanding”. In: *ECCV*. Springer. 2020, pp. 154–171.

[101] Yuge Shi, Basura Fernando, and Richard Hartley. “Action anticipation with rbf kernelized feature mapping rnn”. In: *ECCV*. 2018, pp. 301–317.

[102] Yunsheng Shi et al. “Masked label prediction: Unified massage passing model for semi-supervised classification”. In: *arXiv preprint arXiv:2009.03509* (2020).

[103] Gurkirt Singh et al. “Online real-time multiple spatiotemporal action localisation and prediction”. In: *ICCV*. 2017, pp. 3637–3646.

[104] Chen Sun et al. “Videobert: A joint model for video and language representation learning”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7464–7473.

[105] Zhan Tong et al. “Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training”. In: *arXiv preprint arXiv:2203.12602* (2022).

[106] Vivian V Valentin, Anthony Dickinson, and John P O’Doherty. “Determining the neural substrates of goal-directed learning in the human brain”. In: *Journal of Neuroscience* 27.15 (2007), pp. 4019–4026.

[107] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. “Anticipating visual representations from unlabeled video”. In: *CVPR*. 2016, pp. 98–106.

[108] Haochen Wang et al. “Swiftnet: Real-time video object segmentation”. In: *CVPR*. 2021, pp. 1296–1305.

[109] Limin Wang et al. “Temporal segment networks for action recognition in videos”. In: *IEEE transactions on pattern analysis and machine intelligence* 41.11 (2018), pp. 2740–2755.

[110] Wen Wang et al. “Ttpp: Temporal transformer with progressive prediction for efficient action anticipation”. In: *Neurocomputing* 438 (2021), pp. 270–279.

[111] Donglai Wei et al. “Learning and using the arrow of time”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8052–8060.

[112] Chao-Yuan Wu et al. “MeMViT: Memory-Augmented Multiscale Vision Transformer for Efficient Long-Term Video Recognition”. In: *arXiv preprint arXiv:2201.08383* (2022).

[113] Yu-Syuan Xu et al. “Dynamic video segmentation network”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 6556–6565.

[114] Takuma Yagi et al. “Future person localization in first-person videos”. In: *CVPR*. 2018, pp. 7593–7602.

[115] Ceyuan Yang et al. “Video representation learning with visual tempo consistency”. In: *arXiv preprint arXiv:2006.15489* (2020).

[116] Yu Yao et al. “BiTraP: Bi-directional Pedestrian Trajectory Prediction with Multi-modal Goal Estimation”. In: *IEEE Robotics and Automation Letters* (2021).

[117] Olga Zatsarynna, Yazan Abu Farha, and Juergen Gall. “Multi-modal temporal convolutional network for anticipating actions in egocentric videos”. In: *CVPR*. 2021, pp. 2249–2258.

[118] Bowen Zhang et al. “Real-time action recognition with enhanced motion vector CNNs”. In: *CVPR*. 2016, pp. 2718–2726.

[119] Lingyao Zhang et al. “Map-Adaptive Goal-Based Trajectory Prediction”. In: *arXiv preprint arXiv:2009.04450* (2020).

[120] Hang Zhao et al. “Tnt: Target-driven trajectory prediction”. In: *arXiv preprint arXiv:2008.08294* (2020).

[121] Zeyun Zhong et al. “Anticipative Feature Fusion Transformer for Multi-Modal Action Anticipation”. In: *arXiv preprint arXiv:2210.12649* (2022).

[122] Bolei Zhou et al. “Temporal relational reasoning in videos”. In: *ECCV*. 2018, pp. 803–818.

[123] Xizhou Zhu et al. “Deep feature flow for video recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2349–2358.

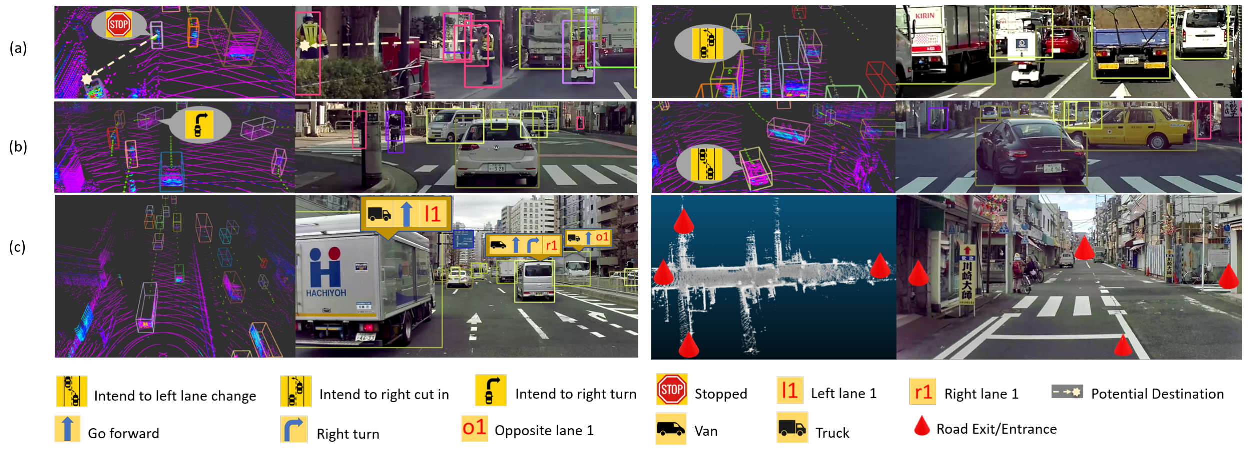


Figure A.1: Visualization of three types of labels: (a-b) Intention labels; and (c) Environmental labels. The left part of each image is from laser scan and the right part is from RGB camera. In (a), the current status of pedestrian is "Stopped", and the potential destination shows where the pedestrian may go in the future. In (c) left, the blue arrow indicates the possible action of the vehicle based on the current lane it is on. The red words show the lane position related to the ego-vehicle.

Appendix A

Additional Information for LOKI

A.1 Details of the LOKI Dataset

The LOKI dataset is collected from central Tokyo, Japan using an instrumented Honda SHUTTLE DBA-GK9 vehicle. Driving scenarios are collected from both suburban and urban areas at different times of the day. The camera, LiDAR, GPS, and vehicle CAN BUS information were recorded. The RGB camera and four LiDAR sensors are placed on top of the vehicle to obtain better environment coverage. In addition, the timestamps were recorded for post multi-sensor synchronization processing. The cameras and LiDARs were placed on top of the vehicle to obtain better environment coverage. The sensors used for recording this dataset are listed below:

- A color SEKONIX SF332X-10X video camera (30HZ frame rate, 1928×1280 resolution and 60 degree field-of-view (FOV)).
- Four Velodyne VLP-32C 3D LiDARs (10 HZ spin-rate, 32 laser beams, range: 200m, vertical FOV 40 degrees).
- A MTi-G-710-GNSS/INS-2A8G4 with output gyros, accelerometers and GPS.

We used the CAN BUS to compensate for the ego-motion while merging the LiDAR data and then transforming it to the virtual position (the center of the vehicle). The calibration is obtained through the extrinsic (the transformation between the virtual LiDAR point and camera) and intrinsic camera parameters.

The recorded agents fall into 8 main classes. The vehicle classes are truck, van, car, bicyclist, motorcyclist, and bus. The pedestrian classes are pedestrian and wheelchair. As described in the main manuscript, we have three types of labels in the LOKI dataset: Intention labels, Environmental labels, and Contextual labels. Intention labels for the vehicle classes include diverse motion states such as stop, lane change, cut-in, etc., which can be observed in suburban and urban driving scenarios. Similarly, we annotated intention labels for the pedestrian classes such as moving, waiting to cross, etc. We additionally annotate a potential destination of stopped / waiting agents under the pedestrian classes, which is a direct indicator of their intention. Note that the potential destination cannot be obtained from the future location of agents as they mostly stay still until the end of the video clip. To further explore how environments and scene context can affect the future behavior of agents, we annotate environmental labels (lane information, traffic light/sign, road entrance/exit) as well as contextual labels (age, gender, weather, road condition). Figure A.1 illustrates different types of labels that we annotated in the LOKI dataset.

A.2 Model Implementation

In this section we provide more information on our data pre-processing, model choices and architecture details of each module.

Data Pre-processing

The LOKI dataset contains diverse traffic scenarios of up to 20 seconds, with the average recorded scene length of 12.6 seconds. With access to longer recordings, our dataset can be used for a multitude of trajectory prediction settings, from very short-term observations and predictions (3 seconds) to much longer observations and prediction horizons (10+ seconds). In our work, we consider a long-term prediction setting with a $3s$ observation horizon and a $5s$ prediction horizon. Thus, we filter all agents that are not observed for at least $8s$ in a given traffic scenario. We use a sliding window of $0.2s$ to augment tracklets. Furthermore, as in other works [11, 99], we further augment our dataset during training with “rare“ examples such as turning and lane changes.

We solve the problem of intention prediction and trajectory prediction jointly as described in the main manuscript. The types of intentions for pedestrian agents and vehicle agents are different due to their different traffic restrictions and trajectory behaviors. For vehicles, we use the following set of discrete actions: *Other/moving*, *Stopped*, *Parked*, *Lane change*, *Turn left*, and *Turn right*. We group *Lane change to the left* and *Lane change to the right* into a single intention type, as the number of instances that contain lane changes are much smaller and we noticed that separating the two did not improve performance. Furthermore, we do not compute a loss or predict trajectories that contain *Cut into the left* and *Cut into the right* , as we noticed that these constitute less than 0.01% of the dataset, making it hard for the model to meaningfully distinguish between turning and lane-changing behavior. For pedestrians, we use the following set of intentions: *Moving*, *Waiting to cross*, *Crossing the road*, and *Stopped*.

Our dataset originally contains frame-wise action labels for each agent. In order to use them as intention labels, we define intention to be a future action [87]. Thus, the intention of an agent at frame m is the agent's action at frame $m + q$ where we fix $q = 4$ frames ($0.8s$) for our work. Note that for the observation period, we do not use intentions and only input observed actions to the model to prevent ground-truth leakage; the intention labels are only used for future trajectory prediction.

Observation Encoder

The observation encoder outputs a representation of past motion history, observed actions, and lane information for each actor independently. In our paper, we refer to past actions and lane information as observed states.

	Layer	Input shape	Output shape
0	encoder_past.GRUCell.enc	[1, 15, 21]	[1, 64]

Table A.1: We use a GRU to encode the observation information for each actor. We use a hidden dimension of 64. The input is 15 observation frames with 21 inputs at each frame (2 from position, 8 from vehicle actions, 5 from pedestrian actions, and 6 from lane information). We use one-hot encoding to represent action types. We also include a None class for both vehicle and pedestrian actions. This allows vehicle agents to choose None for pedestrian action types and pedestrian agents to choose None for vehicle action types.

Long-term Goal Proposal Network

For each actor, we first independently predict a proposed long-term goal position [78, 120]. The proposed destination is similar as in other works [78] and is simply the endpoint of the trajectory, which in our case is 5s in the future. Because there are many plausible futures, we capture multimodality by learning a long-term goal distribution for each actor. To predict multiple trajectories, we sample various goals and condition our Scene Graph + Prediction decoder module on each sampled goal. We follow a similar formulation as proposed in [78] and use a Conditional Variational Autoencoder (CVAE) to learn a latent distribution of the goals.

	Layer	Input shape	Output shape
0	encoder_destination.Linear_1	[1, 2]	[1, 8]
1	encoder_destination.ReLU_1	-	-
2	encoder_destination.Linear_2	[1, 8]	[1, 16]
3	encoder_destination.ReLU_2	-	-
4	encoder_destination.Linear_3	[1, 16]	[1, 16]
5	encoder_latent.Linear_1	[1, 80]	[1, 8]
6	encoder_latent.ReLU_1	-	-
7	encoder_latent.Linear_2	[1, 8]	[1, 50]
8	encoder_latent.ReLU_2	-	-
9	encoder_latent.Linear_3	[1, 50]	[1, 32]
10	decoder_latent.Linear_1	[1, 80]	[1, 1024]
11	decoder_latent.ReLU_1	-	-
12	decoder_latent.Linear_2	[1, 1024]	[1, 512]
13	decoder_latent.ReLU_2	-	-
14	decoder_latent.Linear_3	[1, 512]	[1, 1024]
15	decoder_latent.ReLU_3	-	-
16	decoder_latent.Linear_4	[1, 1024]	[1, 2]

Table A.2: Sub-network architectures used for the goal-proposal network, modeled closely from model [78]. Batch size of 1 used for example.

Scene Graph + Prediction Module

The Scene Graph and prediction module performs joint intention and trajectory prediction while reasoning about various factors that may affect agent intent including i) agent's own will ii) agent-agent interaction and iii) agent-environment influence.

We construct a traffic scene graph to capture interaction and environmental influence. We have two types of nodes: 1) agent nodes 2) road entrance/exit nodes. The agent nodes are for dynamic agents in a scene (vehicles and pedestrians). The road entrance/exit nodes are static nodes that are positional markers that indicate where a road entrance or exit lies. These static nodes are used to provide information regarding map topology. In this work, we assume that these road markers are accessible to the model based on the annotations in our dataset. As described in our main manuscript, we use directional edges to propagate information through the various scene agents. Agents are connected to each other with bidirectional edges if they are within a certain threshold of 20 meters away from each other. Similarly, we connect a directed edge from static nodes to dynamic nodes if the agent is within 35 meters of the road entrance/exit. This graph is flexible in that a variable number of nodes or node types can be added as modifications to this graph.

The Scene Graph + Prediction Module is then used to recurrently propagate information, predict intention, and predict trajectory. At each timestep, we first compute edge attributes between each pair of nodes. We use the edge_attr network to embed nodes' velocities and relative positions between each pair of nodes. We then use the transformer_conv layer (with a single attention head) [102] for message passing and update each node's hidden states based on its neighbors. Following this, we use the vehicle_intention_predictor and pedestrian_intention_predictor networks to predict agent intention at that current timestep. The trajectory_predictor is then conditioned on the hidden state of rnn_future_GRUCell.dec and the current intention prediction to predict the next position of each agent. Finally, the predicted positions are then inputted into rnn_future_GRUCell.dec to update the hidden states of each actor. This entire process is repeated for the prediction horizon length to unroll full trajectories while accounting for interactions and environmental information.

	Layer	Input shape	Output shape
0	trajectory_predictor.Linear_1	[1, 93]	[1, 80]
1	trajectory_predictor.ReLU_1	-	-
2	trajectory_predictor.Linear_2	[1, 80]	[1, 40]
3	trajectory_predictor.ReLU_2	-	-
4	trajectory_predictor.Linear_3	[1, 40]	[1, 2]
5	vehicle_intention_predictor.Linear_1	[1, 80]	[1, 256]
6	vehicle_intention_predictor.ReLU_1	-	-
7	vehicle_intention_predictor.Linear_2	[1, 256]	[1, 128]
8	vehicle_intention_predictor.ReLU_2	-	-
9	vehicle_intention_predictor.Linear_3	[1, 128]	[1, 8]
10	pedestrian_intention_predictor.Linear_1	[1, 80]	[1, 256]
11	pedestrian_intention_predictor.ReLU_1	-	-
12	pedestrian_intention_predictor.Linear_2	[1, 256]	[1, 128]
13	pedestrian_intention_predictor.ReLU_2	-	-
14	pedestrian_intention_predictor.Linear_3	[1, 128]	[1, 5]
15	rnn_future.GRUCell.dec	[1, 1, 80]	[1, 1, 80]
16	edge_attr.Linear_1	[1, 8]	[1, 16]
17	edge_attr.ReLU_1	-	-
18	edge_attr.Linear_2	[1, 16]	[1, 16]
19	transformer_conv_1	[1, 80]	[1, 80]

Table A.3: Sub-network architectures used for the Scene Graph + Prediction module. Batch size of 1 used for example.

Training

Loss Functions

For convenience, we copy the loss functions used for training from our manuscript:

$$\mathcal{L}_{GPN} = \alpha_1 D_{KL}(\mathcal{N}(\mu, \sigma) \| \mathcal{N}(0, I)) + \alpha_2 \|\hat{G} - G\|_2^2$$

$$\mathcal{L}_{int} = -\sum_{i=0}^{n} w_i * y_i * log(\hat{y_i})$$

$$\mathcal{L}_{traj} = ||V - \hat{V}||_2$$

$$\mathcal{L}_{Final} = \lambda_1 \mathcal{L}_{GPN} + \lambda_2 \mathcal{L}_{int} + \lambda_3 \mathcal{L}_{traj}$$

We set $\lambda_1 = 1$, $\lambda_2 = 100$, $\lambda_3 = 200$, $\alpha_1 = 1$, $\alpha_2 = 1$

Training details

We train the entire network end-to-end with the $\mathcal{L}_{Final}$ loss using a batch size of 32 scenarios and learning rate of 1×10^{-4} using the ADAM optimizer. The intention prediction and trajectory forecasting tasks are heavily related with one another; thus we observed that training end-to-end helped with performance compared to modular training. Note that our batches are also grouped with an appropriate adjacency list to denote neighbors (connected edges) in a given batch.

During training, we train with the ground-truth destination as the long-term goal, as we noticed that because short-term intentions are influenced by long-term goals, it is important for the intention prediction networks to get a clean signal while training. During testing, we condition on a sampled goal from the Goal-proposal Network. We also adopt the truncation trick as in [78] to appropriately sample based on a varying number of future trajectories. The latent variable is sampled from different distributions depending on the number of future trajectories to be predicted: for $N = 1$ (single-shot) we sample the from $\mathcal{N}(0, 0)$ while for $N = 20$ (multimodal) we sample from $\mathcal{N}(0, 1.1)$.

A.3 Visualizations

In this section, we provide multiple visualizations that illustrate our proposed model's top-1 predictions (Figure A.2), top-5 multimodal predictions (Figure A.3), and our model's predictions with and without intention conditioning (Figure A.4). Please view the video files provided in the supplementary folder for more detailed visualizations.

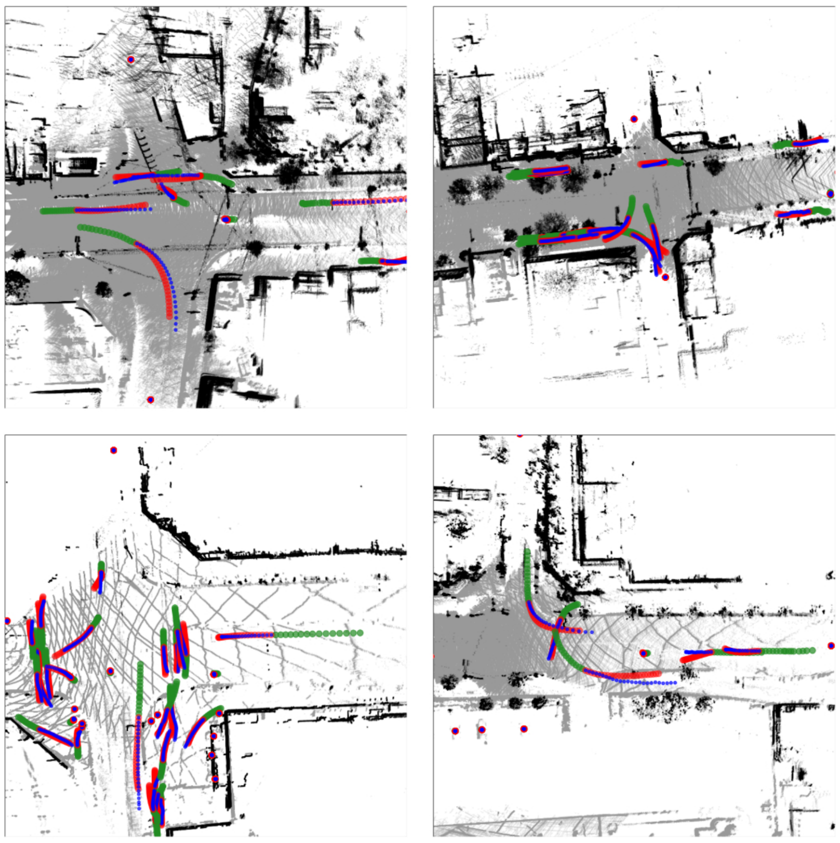


Figure A.2: Visualization of our model's (Ours+IC+SG) top-1 (out of N=20 multimodal setting) predictions. Agent's past trajectory is represented in green. Agent's ground truth future is blue. Agent's predicted trajectories are in red (with increasing opacity to indicate better matches to the ground truth). We observe that our model performs reasonably in complex traffic scenarios.

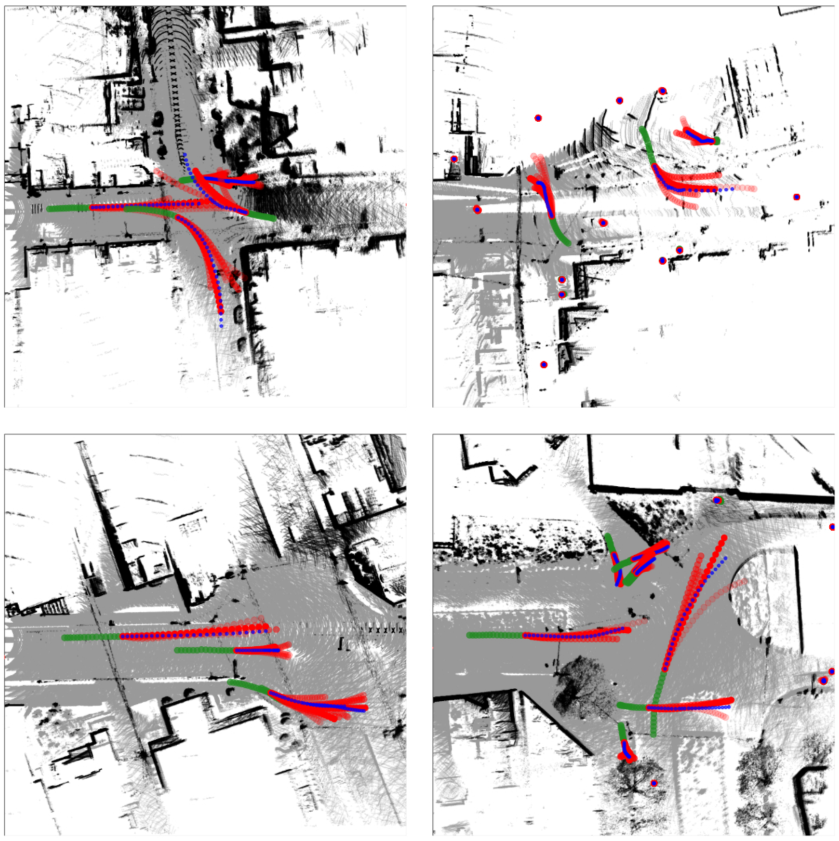


Figure A.3: Visualization of our model's (Ours+IC+SG) top-5 (out of N=20 multimodal setting) predictions. Agent's past trajectory is represented in green. Agent's ground truth future is blue. Agent's predicted trajectories are in red (with increasing opacity to indicate better matches to the ground truth).

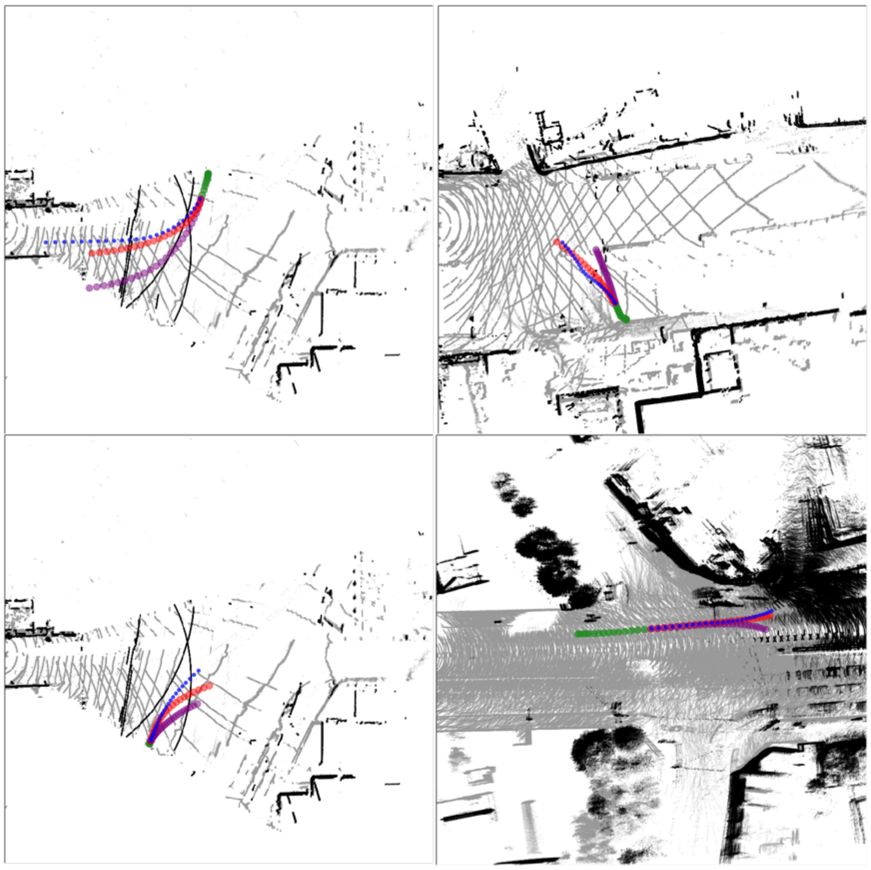


Figure A.4: Comparison of with and without intention priors/scene graph for trajectory prediction. Agent's past trajectory is represented in green. Agent's ground truth future is blue. The top-1 predictions by the model without intention conditioning and scene graph are in purple. The top-1 predictions by the model with intention conditioning and scene graph are in red. We can qualitatively observe the efficacy of intention conditioning and incorporating interaction and environmental cues.