Collection and Analysis of Driving Videos Based on Traffic Participants

by

Yu Yao

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Robotics)
in the University of Michigan
2021

Doctoral Committee:

Professor Ella M. Atkins, Chair
Associate Professor David J. Crandall
Assistant Research Scientist Xiaoxiao Du
Professor Odest Chadwicke Jenkins
Assistant Professor Ramanarayan Vasudevan

Don't give up.

—Yu Yao

Yu Yao
brianyao@umich.edu
ORCID iD: 0000-0002-2329-2355

To my family.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor Prof. Ella M. Atkins, who offered me the opportunity to become a PhD student that I had been dreaming to be; who has patiently guided me through my research and life path in the past five years; who has encouraged me to explore the unfamiliar machine learning and computer vision areas and corrected me for every false step I would have taken in research. Ella is not only a fantastic academic advisor but also an "American mom" to me. I still remember the early days when I just arrived at the US, feeling helpless about living in a new country and being lost about my future career. It was Ella and her lab who embraced me academically and in life, and without notice, five years flew by just like day one. It's been a long journey full of joy, passion, stress and challenges, and I can't say thank you enough to Ella and our lab.

Second, I would like to thank my co-advisor Dr. Xiaoxiao Du, who has supported my research during the past year. Our day-to-day brainstorming was full of excitement and inspiration, which resulted in fruitful research outputs. I would also like to thank the other professors in my research committee, Prof. David J. Crandall, Prof. Odest Chadwicke Jenkins and Prof. Ramanarayan Vasudevan for their helpful suggestions and constructive discussions. I want to especially thank Prof. Crandall for his unselfish support on my computer vision research.

In addition, I'm more than grateful to my two internship experiences in Honda Research Institute (HRI) and Toyota Research Institute (TRI) and would like to thank my mentors and colleagues, Dr. Behzad Dariush, Dr. Chiho Choi, Dr. Yi-ting Chen, Dr. Adrien Gaidon, Dr. Wadim Kehl, Yusuke Kanzawa and Arjun Bhargava. I'm grateful of meeting my internship colleagues Haiming Gang, Yuchen Cui, Chao Cao, Yeping Hu, and Dr. Mingfei Gao in summer 2018, which was the moment I turned my research interest towards machine learning and computer vision for the rest of my PhD period. I would like to express special thanks to Dr. Mingze Xu and Nan Li who have been great collaborators, co-authors and friends during my PhD career. I also thank my friends Haohuan Wang and Qingyu Chen for the great memory we shared during our master period and for their help to my career planning.

Last but not least, I would like to dedicate this PhD dissertation to my family. Without the support of my parents, I won't be able to spend five years on the academic research I love without the worries behind. Thank you, my beloved wife Xinming. We met in the University of Michigan five years ago as fresh graduate students. For the later five years you have been accompanying me, supporting my PhD career and life mentally and physically without hesitation, while pursuing for your own two master degrees. You are a wonderful and talented woman full of empathy, responsibility and love, and my words cannot express my love to you enough. Part of the journey is the end, but I take this as a brand new start, and I can't wait to start it with you together.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

ABSTRACT

Autonomous vehicle (AV) prototypes have been deployed in increasingly varied environments in recent years. An AV must be able to reliably detect and predict the future motion of traffic participants to maintain safe operation based on data collected from high-quality onboard sensors. Sensors such as camera and LiDAR generate high-bandwidth data that requires substantial computational and memory resources. To address these AV challenges, this thesis investigates three related problems: 1) What will the observed traffic participants do? 2) Is an anomalous traffic event likely to happen in near future? and 3) How should we collect fleet-wide high-bandwidth data based on 1) and 2) over the long-term?

The first problem is addressed with future traffic trajectory and pedestrian behavior prediction. We propose a future object localization (FOL) method for trajectory prediction in first person videos (FPV). FOL encodes heterogeneous observations including bounding boxes, optical flow features and ego camera motions with multi-stream recurrent neural networks (RNN) to predict future trajectories. Because FOL does not consider multi-modal future trajectories, its accuracy suffers from accumulated RNN prediction error. We then introduce BiTraP, a goal-conditioned bidirectional multi-modal trajectory prediction method. BiTraP estimates multi-modal trajectories and uses a novel bi-directional decoder and loss to improve longer-term trajectory prediction accuracy. We show that different choices of non-parametric versus parametric target models directly influence predicted multi-modal trajectory distributions. Experiments with two FPV and six bird's-eye view (BEV) datasets show the effectiveness of our methods compared to state-of-the-art. We define pedestrian behavior prediction as a combination of action and intent. We hypothesize that current and future actions are strong intent priors and propose a multi-task learning RNN encoder-decoder network to detect and predict future pedestrian actions and street crossing intent. Experimental results show that one task helps the other so they together achieve state-of-the-art performance on published datasets.

To identify likely traffic anomaly events, we introduce an unsupervised video anomaly detection (VAD) method based on trajectories. We predict locations of

traffic participants over a near-term future horizon and monitor accuracy and consistency of these predictions as evidence of an anomaly. Inconsistent predictions tend to indicate an anomaly has happened or is about to occur. A supervised video action recognition method can then be applied to classify detected anomalies. We introduce a spatial-temporal area under curve (STAUC) metric as a supplement to the existing area under curve (AUC) evaluation and show it captures how well a model detects temporal and spatial locations of anomalous events. Experimental results show the proposed method and consistency-based anomaly score are more robust to moving cameras than image generation based methods; our method achieves state-of-the-art performance over AUC and STAUC metrics.

VAD and action recognition support event-of-interest (EOI) distinction from normal driving data. We introduce a Smart Black Box (SBB), an intelligent event data recorder, to prioritize EOI data in long-term driving. The SBB compresses high-bandwidth data based on EOI potential and on-board storage limits. The SBB is designed to prioritize newer and anomalous driving data and discard older and normal data. An optimal compression factor is selected based on the trade-off between data value and storage cost. Experiments in a traffic simulator and with real-world datasets show the efficiency and effectiveness of using a SBB to collect high-quality videos over long-term driving.

CHAPTER I

Introduction

1.1 Motivation

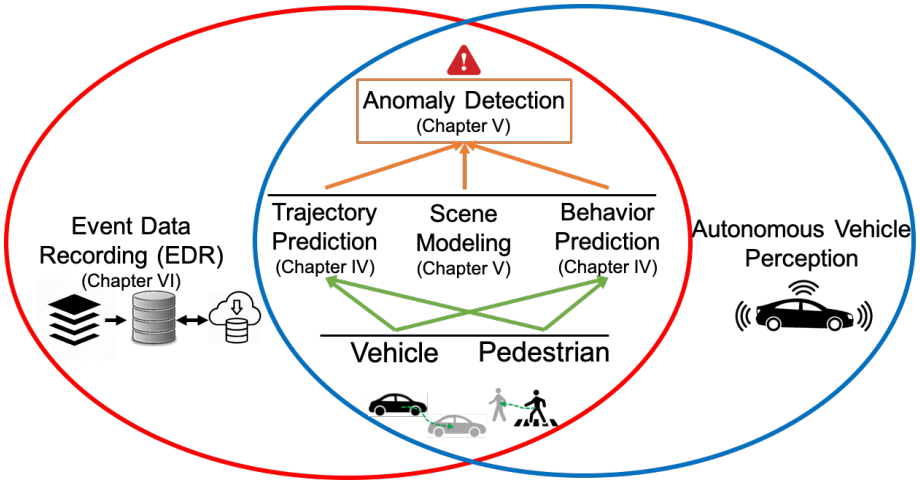


Figure 1.1: This dissertation proposes an EDR (red oval), then investigates its overlap with AV perception (blue oval). Machine learning solutions are proposed for trajectory and behavior prediction of traffic participants as well as anomaly detection. The arrows from A to B means A can be used by B or A is a part of B.

Autonomous vehicles (AVs) have the potential to transform the world as we know it, revolutionizing transportation by making it faster, safer, and less labor intensive. The AV perception system (blue oval in Fig. 1.1) detects, localizes and tracks on-road objects, predicts their trajectories and behaviors [5, 6, 7, 8, 9], parses driving scenes [10, 6], and constructs informative maps [5, 11, 8] using high-bandwidth sensors such as camera, radar and Light Detection and Ranging (LiDAR). Due to the broad range of potential conditions necessitating use of machine learning, AV systems need to be tested for billions of driving miles for validation and verification (V&V) before

being deployed in real life. This motivates the emergence of intelligent event data recorder (EDR) units to efficiently records events of interest (EOIs, *e.g.*, anomalous data) that challenge AV perception (red oval in Fig. 1.1). For collision avoidance, a perception system needs to understand and predict the trajectory (*i.e.*, the path a participant will follow) and behavior (*e.g.*, crossing street or turning left) of surrounding traffic participants. To ensure valuable data is recorded, the intelligent EDR needs these functions to detect anomalies that challenge a perception system, shown by the overlap of the two ovals in Fig. 1.1. This dissertation tackles all these problems. Specifically, we explore methods for trajectory and behavior modeling, leverage trajectory prediction to develop traffic video anomaly detection (VAD) methods and finally design an intelligent EDR system based on them. This dissertation describes machine learning research with video data recorded from on-board cameras since video provides richer scene and object information than radar, and cameras are less expensive and easier to distribute and deploy than LiDAR units.

1.1.1 Trajectory and Behavior Prediction in Driving Videos

(a) Future vehicle localization in HEV-I dataset [7]

(b) Predestrian behavior prediction in PIE dataset [12]

Figure 1.2: Understanding and predicting behavior of traffic participants is essential for autonomous driving systems. Typical tasks include but are not limited to (a) Vehicle trajectory prediction and (b) Pedestrian trajectory and intent prediction.

It is important for AV systems to accurately perceive and safely react to the extremely diverse driving scenarios in the real world. This requires AV perception systems to understand the behavior of surrounding traffic participants (*e.g.*, vehicles, pedestrians and cyclists) and accurately predict their future trajectories and behaviors. For example, in Fig. 1.2(a), is the ego car (the car with this camera mounted on it) allowed to go straight or turn left? By accurately predicting the future locations of the white car and the red car in the image, we might decide to stop or yield to avoid

collision. Similarly, in Fig. 1.2(b), the ego car needs to wait a couple of seconds, yielding to the pedestrian who has the intent to cross the street to reach the goal (the sidewalk to the right). In this dissertation, we define *object future trajectory* as the sequence of positions in bird's eye view (BEV), or bounding boxes in first-person view (FPV) coordinates. We separately consider the vehicle and pedestrian trajectory modeling problems due to their different visual features, motion patterns and roles in traffic. We define the behavior as the semantic action (*e.g.*, walking, standing, crossing) of an object and the intent of that object to conduct a specific action (*e.g.*, will cross).

Our early work [7] reveals that vehicle modeling from driving videos is relatively straightforward, since vehicles usually follow basic traffic rules and routines with strict motion constraints. For example, a car usually would not run into a sidewalk nor would it suddenly make a 180° turn. However, pedestrian behavior is more complicated and difficult to predict due to a pedestrian's interactions with the environment and unconstrained motions. For example, a pedestrian walking into an intersection could suddenly stop, run or turn around, any of which should impact how AV systems make decisions. Therefore, pedestrian behavior and trajectory prediction are considered crucial and difficult problems that we need to address [13, 12, 14, 15].

1.1.2 Anomaly Detection in Driving Videos

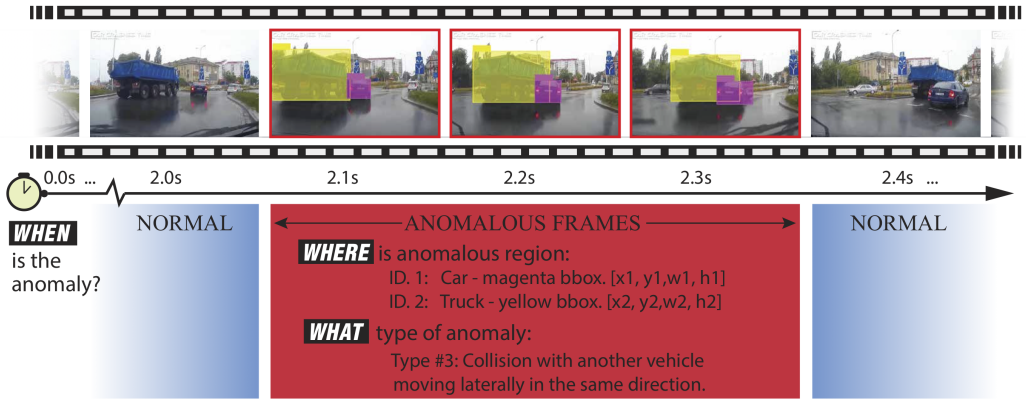


Figure 1.3: Overview of anomaly detection, localization and classification in egocentric traffic videos [1]

Trajectory and behavior of traffic participants are usually modeled from normal traffic scenarios where no accident or near-accident happens, *e.g.*, a vehicle making a proper turn at an intersection or a person walking across a street without redundant action. However, AV systems can be confused and even fail in anomalous situations,

for example when a pedestrian suddenly stops in the middle of the road or when a car cuts into the ego car's lane with insufficient space. Existence of such challenging scenarios requires an AV to effectively detect, localize and classify anomalies (Fig. 1.3) in different driving scenarios [16, 17, 18]. Moreover, detecting anomalies can guide an intelligent event data recorder (*i.e.*, a Smart Black Box) to collect challenging data for V&V purposes as will be introduced in Section 1.1.3. Supervised classification methods [19, 20, 21] require large amounts of annotated training data that are difficult to generate due to the fact that anomalous events are rare in naturalistic traffic videos. Therefore this dissertation pursues unsupervised methods for VAD in driving videos. We also present benchmarks of supervised video action recognition and online action detection methods for completeness.

1.1.3 Intelligent Event Data Recorder

Naturalistic Field Operation Tests (NFOTs) have been performed to enable AV V&V [22], which requires millions or billions of miles of data before enough events of interest (EOIs) occur to accurately measure system performance [23]. Emerging AVs with redundant high-bandwidth sensors (*e.g.*, camera and LiDAR) generate as much as 1 GB/second of raw data, a figure that scales to around 2160 TB/year given an average driving time per person of 660 hours per year [24]. Therefore, effective capture of this large data for NFOT is challenging, making it necessary to develop methods for efficient data compression and discard capabilities. This dissertation presents a value-driven, high-bandwidth EDR to tackle the aforementioned problems called the Smart Black Box (SBB). Levering the VAD and anomaly classification methods mentioned above, the SBB recognizes valuable data (*e.g.*, anomalous event data) and applies proper compression factors to optimize the usage of on-board storage in long-term driving data collection processes.

1.2 Problem Statement and Research Approach

This dissertation aims to: 1) Extend state-of-the-art (SOTA) in traffic participant trajectory and behavior modeling; 2) Develop effective unsupervised methods for traffic VAD in driving videos; 3) Design an intelligent EDR that efficiently collects, compresses and manages long-term high-bandwidth data (*i.e.*, videos). This section defines the problems involved in the above three tasks which will be used across this dissertation.

1.2.1 Trajectory and Behavior Prediction

Trajectory prediction forecasts future location (in BEV) or bounding boxes (in FPV) of traffic participants (*i.e.*, vehicles, pedestrians) and is also called future object localization (FOL) [7] in FPV scenarios. We define the trajectory prediction problem as:

$$\hat{\mathbf{Y}} = TrajectoryPredictor(\mathbf{X}, \mathbf{I}, \mathbf{E}), \tag{1.1}$$

where $\mathbf{X} = \{X_{t-\tau+1}, X_{t-\tau+2}, ..., X_t\}$ is the past trajectory of a traffic participant, $\mathbf{I} = \{I_{t-\tau+1}, I_{t-\tau+2}, ..., I_t\}$ is the observed image sequence, and $\mathbf{E} = \{E_{t-\tau+1}, E_{t-\tau+2}, ..., E_t\}$ is past ego motion. For BEV, $X_t = [x_t, y_t]$ is location coordinates, while for FPV, $X_t = [left_t, top_t, right_t, bottom_t]$ is bounding box coordinates. $\mathbf{Y} = \{Y_{t+1}, Y_{t+2}, ..., Y_{t+\delta}\}$ is a single future trajectory for single-modal trajectory prediction. For multi-modal trajectory prediction, $\mathbf{Y}$ is either a set of possible future trajectories or a distribution of future trajectories.

Chapter IV proposes a multi-stream recurrent neural network (RNN) encoder-decoder (ED) model to encode X and I object features and predict future trajectories. An ego-motion ED stream is added to predict ego car odometry as compensation to FOL in FPV. Then we introduced a goal-conditioned bi-directional trajectory predictor which estimates multi-modal goals first and then predicts multi-modal trajectories using a bi-directional RNN decoder.

Behavior prediction is defined as a combination of semantic action detection and prediction, and intent detection:

$$\hat{a}_t = ActionDetector(\mathbf{X}, \mathbf{I}), \tag{1.2}$$

$$[\hat{a}]_{t+1}^{t+\delta} = ActionPredictor(\mathbf{X}, \mathbf{I}), \tag{1.3}$$

$$\hat{i}_t = IntentDetector(\mathbf{X}, \mathbf{I}), \tag{1.4}$$

where $\hat{a}_t$ is the detected present action at t, $[\hat{a}]_{t+1}^{t+\delta}$ is the predicted action over the future δ frames, and $\hat{i}_t$ is the detected intent at t.

In Chapter IV, we propose a multi-task learning approach for behavior prediction. Past image observations are encoded, and a decoder is used to predict the future action. The future action prediction output is used as an extra input to the encoder to estimate the present action and intent.

1.2.2 Video Anomaly Detection

Video anomaly detection (VAD) has been extensively studied for surveillance camera applications, action recognition, and scene understanding. VAD takes continuous video frames as input and computes one anomaly score for each frame, as shown in Eq. (1.5):

$$s_1, s_2, ..., s_T = VAD(I_1, I_2, ..., I_T), \tag{1.5}$$

where $I_1, I_2, ..., I_T$ are T continuous video frames and $s_1, s_2, ..., s_T$ are corresponding anomaly scores.

Chapter V tackles the VAD problem by modeling normality in traffic participant motion and appearance features in an unsupervised manner. Specifically, we adapted the FOL method to an online mode and trained it on normal driving data to learn trajectory normality. The FOL model is then deployed on videos containing anomalies to detect inaccurate or inconsistent predictions as anomalies. This model is further extended towards multi-normality learning [1] using a margin learning approach [25]. We further combine this trajectory based method with an appearance based method [2], resulting in an Ensemble VAD method.

1.2.3 Intelligent Event Data Recorder

Current high-quality AV sensors generate high-bandwidth data (*e.g.*, HD videos, LiDAR point clouds) that cannot be all stored onboard or uplinked in long-term driving. Moreover, datalink may not be continuously available or may incur nontrivial costs when a car is not parked near open wifi. To this end, we introduce an intelligent EDR called the *Smart Black Box* to realize memory-efficient high-bandwidth data collection. The SBB addresses three core problems: 1) How do we quantify data value? 2) How do we manage each data recording buffer? 3) How do we optimize the usage of on-board storage to save the most valuable data? Chapter VI investigates each problem respectively and conducts experiments on simulation and real-world data to evaluate the performance of the proposed SBB.

1.3 Innovations and Contributions

Specific innovations of this dissertation are:

- A new task for future vehicle localization in ego-centric videos, and an innovative

multi-stream architecture that combines both object-motion and ego-motion prediction in one model.

- A bi-directional trajectory predictor based on multi-modal goal estimation. Our work is the first to carefully compare multi-modal trajectory prediction results with different latent distribution models.
- A novel multi-task formulation for pedestrian behavior prediction that leverages future action prediction as an important indication of pedestrian intent.
- A new unsupervised traffic VAD method in ego-centric videos that is robust to moving cameras. A new anomaly metric using prediction consistency that is robust to object detection/tracking failures.
- A novel evaluation metric called spatial-temporal area under curve (STAUC) to better evaluate the ability of unsupervised video anomaly detection algorithms to localize anomalies in spatial and temporal domains.
- Our work is first to use real time anomaly detection to prioritize high-bandwidth data collection for on-road vehicles.

Specific contributions of this dissertation are:

- Smart Black Box (SBB) software for intelligent event data recording and a corresponding simulation test bed.
- Software for a multi-stream RNN encoder-decoder network for FOL in driving videos, which achieves SOTA performance on two FPV datasets.
- Software for a bi-directional trajectory prediction algorithm that achieves SOTA performance on two FPV datasets and six BEV datasets.
- Software for a multi-task pedestrian behavior prediction algorithm that achieves SOTA results on the PIE dataset [12].
- Software implementing an unsupervised anomaly detection algorithm for driving video anomaly detection which achieves SOTA performance on three datasets.
- Three published datasets with annotations: one for future vehicle localization and two for traffic video anomaly detection. Benchmarks of baselines and SOTA methods on our datasets are provided.

1.4 Outline

Chapter III introduces the datasets we have collected, annotated and published for this work. Annotation methods, dataset statistics and comparative studies with existing datasets are presented to show the advantages of our datasets. Chapter IV summarizes our work on object trajectory prediction and pedestrian behavior prediction. We first introduce future object localization (FOL) in FPV. We then investigate object goal estimation and present a bi-directional trajectory prediction method to improve long-term (*e.g.*, 4.8 seconds to the future) trajectory prediction accuracy. Finally, we present a multi-task behavior prediction method by using future action prediction as an important indication of pedestrian crossing intent. Experiments on real-world datasets (HEV-I [7], KITTI [5], JAAD [26], PIE [12], ETH-UCY [27, 28], nuScenes [8]) show the effectiveness of our methods.

Chapter V presents our work on unsupervised video anomaly detection (VAD) in driving videos. We propose object trajectory modeling methods and combine models with a scene prediction method to explore VAD for autonomous driving. A novel spatio-temporal evaluation metric is introduced as a supplement to existing VAD evaluation methods. Benchmarks of state of the art VAD, action recognition and online action detection methods on our datasets are provided to support further research.

Chapter VI introduces the design of a Smart Black Box (SBB), an intelligent driving event data recorder (EDR) pipeline to record high-bandwidth raw data as a supplement to current low-bandwidth EDR. Extensive simulation and real-world data collection results show the efficiency and effectiveness of the SBB for long-term data recording. Chapter VII concludes the dissertation and discusses future work directions.

CHAPTER II

Related Work

This chapter reviews the literature referenced throughout the dissertation. Each of the following subsections contains background corresponding to each subsequent research investigation.

2.1 Object Trajectory Prediction

Trajectories in Birds'-eye View (BEV) and First-person View (FPV). Trajectory prediction in BEV predicts future position sequences represented by (x, y) coordinates. Early work in this area includes social force [29], Gaussian procession regression [30] and inverse reinforcement learning (IRL) [31] that assumes a single-modal future and predicts a single best trajectory. Recently, recurrent neural networks (RNNs) such as long short-term memory networks (LSTMs) and gated recurrent units (GRUs) have been applied in an encoder-decoder (ED) format to encode past observations and decode future trajectory. Heterogeneous information such as neighbors, maps, and semantic actions are used as extra inputs to improve trajectory prediction accuracy. As one of the earliest and most important work, Alahi *et al.* [32] proposed a Social-LSTM to model pedestrian trajectories as well as their interactions with neighbors. Following this trend, recent work models context and interactions by improved social-pooling [33, 34], attention networks [35], gated relation networks [36], and graph models [15].

While the above methods are designed for third-person views from static cameras, recent work has considered vision in first-person (egocentric) videos that captures the natural field of view of the person or agent (e.g., vehicle) hosting or "wearing" the camera to study that agent's actions [37, 38], trajectories [39], interactions [40, 41], etc. Object trajectory prediction in FPV predicts future bounding box sequences or human

keypoint sequences. Bhattacharyya *et al.* [42] predict future locations of pedestrians from vehicle-mounted cameras, modeling observation uncertainties with a Bayesian LSTM network. Yagi *et al.* [43] predicts pedestrian trajectories in first-person videos using a convolution-deconvolution framework. We propose future vehicle localization (FVL), a multi-stream GRU-ED to predict future locations of vehicles. FVL is one of the earliest works that focus on FPV [7]. Dash camera ego-motion is modeled using another GRU stream to compensate prediction in [42] and our following work [44]. Human biomechanical models [45] and disentangled pose key-points [46, 14] have been used as extra cues to enhance prediction accuracy in FPV. Malla *et al.* [9] and Rasouli *et al.* [12] incorporated semantic action and intention detection as a prior to boost trajectory prediction accuracy.

Multi-modal Trajectory Prediction. Multi-modal trajectory prediction methods estimate trajectory distributions therefore cover different possible futures, making them appropriate for applications such as path planning in robotics and autonomous driving [47, 48, 49]. Bayesian LSTMs or confidence regression modules are used to estimate epistemic and aleatoric uncertainty [42, 50]. However they are still single-modal methods that assume a single Gaussian distribution for future trajectories. A number of researchers have developed GAN and CVAE methods for multi-modal trajectory prediction. GAN-based methods generate multi-modal trajectories from sampled unit Gaussian noise thus cannot explicitly learn trajectory distributions [51, 52, 53]. Gupta *et al.* [51] develop Social-GAN which captures global context for a Generative Adversarial Network. Sadeghian *et al.* [52] build an attentive GAN to better leverage the social and physical constraints for the trajectories. Kosaraju *et al.* [53] design a bicycle-GAN with local and global discriminators to tackle interactions in different scales. Probabilistic approaches, particularly conditional variational autoencoder (CVAE) based models, have been developed for multi-modal trajectory prediction. Different from GANs [51, 53], CVAEs can explicitly learn the form of a target distribution conditioned on past observations by learning the latent distribution from which it samples. Some CVAE methods assume the target trajectory follows a non-parametric (NP) distribution thus produce multi-modal predictions by sampling from a Gaussian latent space. Lee *et al.* [54] first used CVAE for multi-modal trajectory prediction by incorporating Gaussian latent space sampling to a long short-term memory encoder-decoder (LSTM-ED) model. CVAE with LSTM components has since been used in many applications [55, 56, 57]. Other CVAE-based methods assume parametric trajectory distributions. Ivanovic *et al.*[58] assumed the target trajectory follows a Gaussian Mixture Model (GMM) and designed a Trajectron network to predict GMM

parameters using a spatio-temporal graph. Trajectron++ [48] extended Trajectron to account for dynamics and heterogeneous input data. Our work extends existing CVAE models to include goal estimation and shows improved multi-modal prediction results. Our work also provides novel insights in comparisons between CVAE target distributions (NP and GMM). We propose a bidirectional trajectory predictor (BiTraP) [49] and present a through comparison between NP and parametric GMM models in CVAE-based multi-modal trajectory prediction.

Trajectory Conditioned on Goals. Incorporating goals has been shown to improve trajectory prediction. Rehder *et al.* [59] proposed a particle-filter based method to estimate goal distribution as a prior for trajectory prediction. We drew inspiration from [60] that computed forward and backward rewards based on current position and goal. The path is planned using Inverse Reinforcement Learning (IRL). Our work is distinct due to its bi-directional temporal propagation and integration combined with a CVAE to achieve multi-modal prediction. Rhinehart *et al.* [61] estimated multi-modal semantic action as goals and planned conditioned trajectories using imitative models. Deo *et al.* [62] used IRL to estimate goal states and fused results with past trajectory encodings to generate predictions. Most recently, Mangalam *et al.* [63] designed a PECNet which showed state-of-the-art results on BEV trajectory prediction datasets. However, PECNet only concatenated past trajectory encodings and end-point encodings, which we believe did not fully take advantage of goal information. We have designed a bi-directional trajectory decoder [49] in which current trajectory information is passed forward to the end-points (goals) and goals are recurrently propagated back to the current position. Experiment results show that our goal estimation can help generate more accurate trajectories than achieved in previous studies.

2.2 Pedestrian Action and Intent Detection

Pedestrian action and intent detection have been considered as a branch of the greater video action recognition problem. Most existing work detects pedestrian action or intent by detecting pedestrian bounding boxes or skeleton first and then performing action recognition on local features.

Action Recognition attempts to assign video frames or clips into categories, and thus can be applied to traffic anomaly classification, e.g. front-collision, turning-collision, vehicle-pedestrian collision, etc. Two-stream networks [64] and temporal segment networks (TSN) [65] leverage RGB and optical flow data. Tran *et al.* [66]

first proposed 3D convolutional networks (C3D) for spatio-temporal modeling, followed by an inflated model [67]. Recent work substitutes 3D convolution with 2D and 1D convolution blocks (R(2+1)D [68]) to improve effectiveness and efficiency. Feichtenhofer *et al.* [69] propose the SlowFast model to extract video features from low and high frame rate streams. Online action detection in untrimmed, streaming videos is addressed by De Geest *et al.* [70], while Gao *et al.* [71] propose a reinforce encoder-decoder (RED) to tackle action prediction and online action recognition. Shou *et al.* [72] model temporal consistency with a generative adversarial network (GAN). Xu *et al.* [73] propose a temporal recurrent network (TRN) that predicts future actions to aid in online action detection. Gao *et al.* [74] use reinforcement learning to detect the start time of actions.

Pedestrian Intent and Action Detection tries to classify semantic actions (i.e. walking, crossing etc.) and a pedestrian's intent to cross the road (e.g., will cross vs. will not cross). Compared to generalized action recognition, pedestrian action and intent has not been widely studied until recently. Rasouli *et al.* [75] created a joint attention in autonomous driving (JAAD) dataset for pedestrian action and intent studies. Along with the JAAD dataset, an image classification method using AlexNet and a fully convolutional network (FCN) was presented as the baseline for action (walking and looking) and intent detection (will cross and will not cross) [75]. Local visual features are extracted from the image regions located at pedestrian bounding boxes as the context information for neural networks. Following JAAD, Rasouli *et al.* [12] introduced the Pedestrian Intent Estimation (PIE) dataset. Bounding box trajectories and context features are combined to predict pedestrian intent of crossing a road. Bouhsain *et al.* [76] added a bounding box prediction encoder decoder in parallel to an intent detection encoder decoder to improve the accuracy, resulting in a multi-task network. Pedestrian pose (skeleton) key points, as more informative features than bounding boxes, have been used to predict crossing intent [77, 78]. Wei *et al.* [79] used 3D pose obtained from an RGB-D camera to characterize human attention and then predict intent. Researchers have also studied how interactions with the environment and traffic participants influence pedestrian crossing intent. Xie *et al.* [80] proposed to learn the environment's attractive force on a pedestrian using a bird's eye view map. Pedestrian intent is then detected by predicting the optimal trajectories of all pedestrians in the scene. Leveraging recent advances in graph convolutional networks (GCN), Liu *et al.* [15] built traffic graphs centered at either the pedestrian or the ego car to model the relation between pedestrian intent, ego car motion, and other traffic participants. Schneemann *et al.* [81] detected context

such as road edges, crosswalks, and waiting areas using semantic segmentation and detect pedestrian intent with a support vector machine (SVM). Zhang *et al.* [82] processed heterogeneous information such as vehicle speed, pedestrian speed, cross walk detection, etc., using an attended LSTM network and detect crossing intent by an SVM.

Distinct from crossing intent, pedestrian action is multi-class and scenario dependent, e.g., walking, standing, participating in a phone call, loading a car, etc. Despite significant progress in the field of video action recognition, few researchers have studied pedestrian action detection. Liang *et al.* [33] combined pose key points, image features, semantic features and interaction features to detect human action in surveillance cameras. Malla *et al.* [9] introduced a Trajectory Inference using Targeted Action priors Network (TITAN) dataset which contains more detailed hierarchical action annotations in driving videos and provided results from baseline action recognition methods I3D [67] and R3D [68]. TITAN is considered more of a trajectory prediction dataset than a pedestrian action dataset. Our work considers pedestrian intent as their future action and proposes an entangled model to detection pedestrian intent by predicting his/her future action.

2.3 Anomaly Detection for Autonomous Vehicles

Traditional Anomalous Event Detection can be straightforwardly accomplished based on pre-defined rules or safety envelope violations such as sudden deceleration or insufficient following distance [22, 83]. Other events can be detected from driver behavior recognition or driving environment characterization. Driver behavior has been assessed to-date by training statistical models or feature extraction models based on CAN bus signals [84, 85, 86] and driver observation camera data frames [87]. Most environment detection research focuses on recognizing behaviors in surrounding vehicles [88, 89, 90] and/or pedestrians [91, 92, 93], after which an EOI analysis of human-vehicle interaction is possible. The detection of road conditions such as potholes has also been investigated in [94]. Although many event and anomaly detection techniques have been designed for automated driving and risk recovery [84, 95], few are used to guide data collection and compression.

Existing Video Anomaly Detection (VAD) datasets are typically from surveillance cameras. For example, UCSD Ped1/Ped2 [96], CUHK Avenue [97], and ShanghaiTech [2] were collected from campus surveillance cameras and include anomalies like prohibited objects and abnormal movements, while UCF-Crime [98] includes ac-

cidents, robbery, and theft. Anomaly detection in egocentric traffic videos has very recently attracted attention. Chan *et al.* [19] propose the StreetAccident dataset of on-road accidents with 620 video clips collected from dash cameras. The last ten frames of each clip are annotated as anomalous. We [44] proposed the A3D dataset containing 1,500 anomalous videos in which abnormal events are annotated with start and end times. Fang *et al.* [99] introduce the DADA dataset for driver attention prediction in accidents, while Herzig *et al.* [21] extract a collision dataset with 803 videos from BDD100K [6]. Our newer DoTA dataset is much larger (4,677 videos) and, more importantly, contains richer annotations that support traffic video anomaly analysis from spatial, temporal and categorical perspectives.

Existing VAD models mainly focus on detecting the start and end of anomalous events and implicitly relate to spatial localization. Hasan *et al.* [100] propose a convolutional Auto-Encoder (ConvAE) to model the normality of video frames by reconstructing stacked input frames. A Convolutional LSTM Auto-Encoder (ConvLSTMAE) is used in [101, 102, 103] to capture regular visual and motion patterns. Luo *et al.* [104] propose a stacked RNN for temporally-coherent sparse coding (TSC-sRNN). Liu *et al.* [2] detect anomalies by looking for differences between predicted future frames and actual observations. Gong *et al.* [105] propose an MemAE network to query pre-saved memory units for reconstruction, while Wang *et al.* [106] design generalized one-class sub-spaces for discriminative regularity modeling. Other work has recently studied object-centric approaches. Ionescu *et al.* [107] propose K-means to cluster object features and train multiple support vector machine (SVM) classifiers with confidence as anomaly score. Morais *et al.* [46] model human skeleton regularity with local-global autoencoders and compute per-object anomaly scores. VAD in egocentric traffic scenarios is a challenging problem due to dynamic foreground and background, perspective projection, and complicated scenes. We benchmark state-of-the-art VAD methods and their variants on DoTA dataset.

2.4 Intelligent Event Data Recorder

The automobile event data recorder (EDR) was developed by manufacturers to analyze precursor and crash data with impact-triggered recording [108][17]. The EDR captures low-bandwidth data including but not limited to vehicle speed, engine speed, throttle, brake status, and acceleration. Improvements on EDRs have extended the event list an EDR can detect [83] and support system execution replay [109]. An in-vehicle data recorder was proposed in [110] to record heterogeneous data from

asynchronous onboard sensors. However, these publications do not address the trade-off between data compression losses and finite storage constraints that exist in long-term high-bandwidth data collection.

We design the Smart Black Box (SBB), an intelligent EDR that operates long-term to record high-bandwidth data without human intervention. An essential challenge is to selectively remove recorded data as needed to make room for new data. Two models have been adopted in previous data recorders. The most common data recorder continuously writes data until the storage is full then terminates the recording. The newest data is discarded in this model once storage is filled. Some EDR systems use circular buffers to record data so that once the storage is full, the oldest data is aged out with a first-in-first-out (FIFO) queue. This model values new data over old data without processing data contents. In [111], multi-resolution storage was supported by generating coarse representations of raw data, called summaries. Summaries in the database are then aged out by a user-defined aging function. Such an approach was applied in [112] where the elimination process considered the frequency at which data is queried. This method focuses on database maintenance and query instead of on-line data collection. In this dissertation, we age out data based on data value metrics and storage limits. The optimization of compression quality is solved and the reproducibility of computer vision algorithms such as object detection [3] and semantic segmentation [4] on compressed data is analyzed.

CHAPTER III

Datasets

This chapter summarizes the datasets we have created and published in our work, including one dataset for future vehicle localization (HEV-I) and two datasets for video anomaly detection (A3D and DoTA). Content, annotations, and statistical summaries of each dataset are described below.

3.1 Honda Egocentric View-Intersection (HEV-I) Dataset

Figure 3.1: HEV-I dataset samples. HEV-I contains videos collected from urban (row 1) and suburban (row 2) intersections. Different weather and lighting conditions (row 2 and row 3) are included for diversity.

The problem of future object localization in egocentric cameras is particularly challenging when multiple vehicles execute different motions (e.g. an ego-vehicle is turning left but yields to another moving car). However, to the best of our knowledge, most existing autonomous driving datasets are proposed for scene understanding tasks [5, 10] that do not contain diverse motions. We introduce a new egocentric

vision dataset, **Honda Egocentric View-Intersection (HEV-I)**, that focuses on intersection scenarios where vehicles exhibit diverse motions due to complex road layouts and vehicle interactions. HEV-I was collected from different intersection types in the San Francisco Bay Area, and consists of 230 videos each ranging between 10 to 60 seconds. Videos were captured by an RGB camera mounted on the windshield of the car, with 1920×1200 resolution (reduced to 1280×640 in this paper) at 10 frames per second (fps). Figure 3.1 shows samples of HEV-I video frames. Different scenarios are included for diversity.

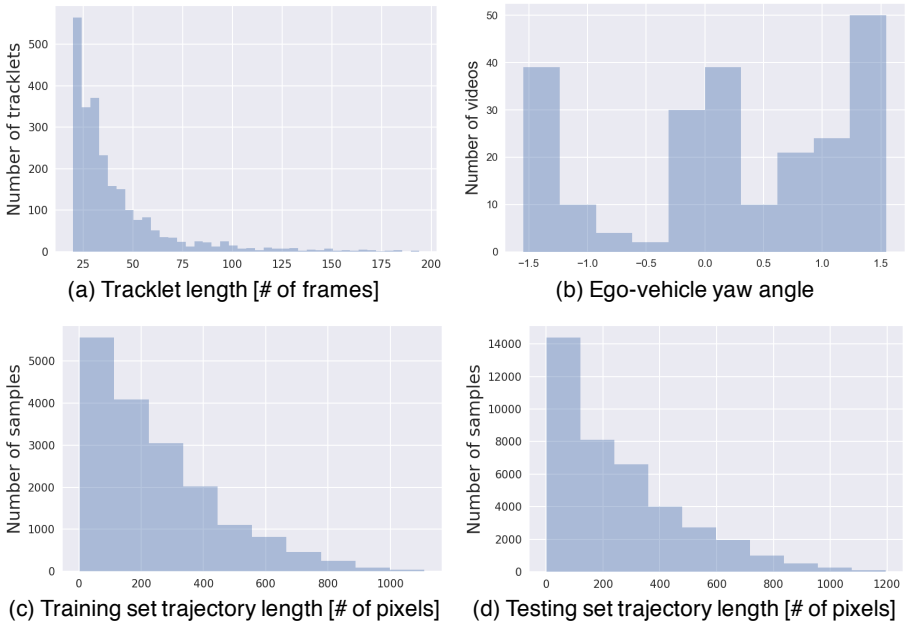


Figure 3.2: HEV-I dataset statistics.

Table 3.1: Comparison with KITTI dataset. The number of vehicles is tallied after filtering out short sequences.

Dataset	# videos	# vehicles	scene types
KITTI	38	541	residential, highway, city road
HEV-I	230	2477	urban intersections

Statistics of HEV-I are shown in Fig. 3.2. As shown, most vehicle tracklets are short in Fig. 3.2 (a) because vehicles usually drive fast thus leave the field of the first-person view quickly. Fig. 3.2 (b) shows the distribution of ego vehicle yaw angle (in *rad*) across all videos, where positive indicates turning left and negative indicates turning right. It can be seen that HEV-I contains a variety of different ego motions. Distributions of training and test sample trajectory lengths (in pixels) are presented

in Fig. 3.2 (c) and (d). Although most lengths are shorter than 100 pixels, the dataset also contains numerous longer trajectories. This is important because longer trajectories are typically more difficult to predict. Compared to existing data like KITTI, the HEV-I dataset contains more videos and vehicles, as shown in Table 3.1. Most object vehicles in KITTI are parked on the road or driving in the same direction on highways, while in HEV-I, all vehicles are at intersections and performing diverse maneuvers. This dissertation develops a future object localization (FOL) network evaluated on the HEV-I dataset. Baselines such as linear and quadratic fitting and a state-of-the-art 1D convolutional network method [43] are also benchmarked on HEV-I for comparative study. The HEV-I dataset can be found at: https://usa.honda-ri.com/hevi.

3.2 AnAn Accident Detection (A3D) Dataset

To evaluate our methods on realistic traffic scenarios, we introduce a new dataset **AnAn Accident Detection (A3D)** of on-road abnormal event videos compiled from 1500 video clips selected from a YouTube channel of dashboard camera streams from different cars in East Asia. Each video contains an abnormal traffic event at different temporal locations. We labeled each video with anomaly start and end times under the consensus of three human annotators. The annotators were instructed to label the anomalies based on common sense, with the start time defined to be the point at which the accident is inevitable and end time the point when all participants recover a normal *moving* condition or fully stop.

We compare our A3D dataset with existing video anomaly detection datasets in Table 3.2. A3D includes a total of 128,175 frames (ranging from 23 to 208 frames per clip) at a rate of 10 frames per second and is clustered into 18 types of traffic accidents each labeled with a brief description. A3D includes driving scenarios with different weather conditions (e.g., sunny, rainy, snowy, etc.), places (e.g., urban, countryside, etc.), and participant types (e.g., cars, motorcycles, pedestrians, animals, etc.). In addition to start and end times, each traffic anomaly is labeled with a binary value indicating whether the ego-vehicle is involved to provide a better understanding of the event. Note that this data could especially benefit the first-person vision community. For example, rear-end collisions are the most difficult to detect from traditional anomaly detection methods. About 60% of accidents in the dataset involve the ego-vehicle; others are observed from a third-person perspective. The A3D dataset can be found at: https://github.com/MoonBlvd/tad-IROS2019.

3.3 Detection of Traffic Anomaly (DoTA) Dataset

To further extend our A3D dataset, we introduce **DoTA**, the first publicly-available traffic video anomaly dataset with temporal, spatial, and categorical annotations. To build DoTA, we collected more than 6,000 video clips mainly from two YouTube channels [1][2] which provides traffic accident videos for driver education purposes. We selected diverse dash camera accident videos from different areas (e.g., East Asia, North America, Europe etc.) under different weather (e.g., sunny, cloudy, raining, snowing, etc.) and lighting conditions (day and night). We avoided videos with accidents that were not visible or where the camera dislodged during the accident, resulting in 4,677 videos with 1280×720 resolution. Each video contains one and only one anomalous event. Though the original videos are at 30 fps, we extracted frames at 10 fps for annotations and experiments. Table 3.2 compares DoTA with other ego-centric traffic anomaly datasets. The DoTA dataset can be found at: https://github.com/MoonBlvd/Detection-of-Traffic-Anomaly.

We annotated the dataset using a custom tool based on Scalabel[3]. Labeling traffic anomalies is subjective, especially for properties like start and end times. To produce high quality annotations, each video was labeled by three annotators, and the temporal and spatial (categorical) annotations were merged by taking average (mode) to minimize individual biases.

Temporal Annotations. Each DoTA video is annotated with anomaly start and end times, which separates it into three temporal partitions: precursor, which is normal video preceding the anomaly, the anomaly window, and post-anomaly, which is normal activity following the anomaly. Duration distributions are shown in Fig. 3.4(a). Since early detection is essential for on-road anomalies [19, 20], we asked the annotators to estimate the anomaly start as the time when the anomaly was inevitable. The anomaly end was approximated as the time when all anomalous objects are out of the field of view or are stationary. Our annotation is different from [99] where a frame is marked as an anomaly start if half of an anomaly participant appears in the camera view; such a start time can be too early because anomaly participants often appear for a while before they start to behave abnormally. Our annotation is also distinct from [19] and [44] where the anomaly start is marked when a crash happens, which does not support early detection.

[1] https://youtube.com/user/CarCrashesTime
[2] https://youtube.com/channel/UC-Oa3wml6F3YcptlFwaLgDA
[3] https://scalabel.ai/

Table 3.2: Comparison of published driving video anomaly datasets. The top section shows surveillance datasets, the middle section shows previous driving video datasets and the bottom section presents our A3D and DoTA datasets.

Dataset	type	# videos	# frames	Annotations
UCSD Ped [113]	Surveillance	98	18.5K (30fps)	temporal
CUHK [97]		37	30.6K (30fps)	temporal
UCF-Crime [98]		1,900	13,8M (30fps)	temporal
ShanghaiTech [104]		437	317K (30fps)	temporal
StreetAccident [19]	Dashcam	620	62K (20fps)	temporal
DADA [99]		2,000	648K (30fps)	temporal, spatial (eye-gaze)
A3D [44]	Dashcam	1,500	128K (10fps)	temporal
DoTA [1]		**4,677**	**732K** (10fps)	temporal, spatial (tracklets), categories

Table 3.3: Traffic anomaly categories in the DoTA dataset.

ID	Short	Anomaly Categories
1	ST	Collision with another vehicle that starts, stops, or is stationary
2	AH	Collision with another vehicle moving ahead or waiting
3	LA	Collision with another vehicle moving laterally in the same direction
4	OC	Collision with another oncoming vehicle
5	TC	Collision with another vehicle that turns into or crosses a road
6	VP	Collision between vehicle and pedestrian
7	VO	Collision with an obstacle in the roadway
8	OO	Out-of-control and leaving the roadway to the left or right
9	UK	Unknown

Spatial Annotations. DoTA is the first traffic anomaly dataset to provide detailed spatio-temporal annotation of anomalous objects. Each anomaly participant is assigned a unique track ID, and each participant's bounding box is labeled from anomaly start to anomaly end or until the object is out of view. We consider seven common traffic participant categories: person (pedestrian), car, truck, bus, motorcycle, bicycle, and rider, following the BDD100K style [6]. Statistics of object categories and per-video anomalous object numbers are shown in Figs. 3.4(c) and 3.4(d). DADA [99] also provides spatial annotations by capturing video observers' eye-gaze for driver attention studies. However, researchers have shown that eye-gaze does not always coincide with the anomalous region, and that gaze can have $\sim$1 to 2 seconds of delay from anomaly start. Our tracklets thus provide improved annotation for spatio-temporal anomaly detection studies.

Anomaly Categories. Each DoTA video is assigned one of the 9 categories listed in Table 3.3 as defined in [114]. We have observed that the same anomaly category

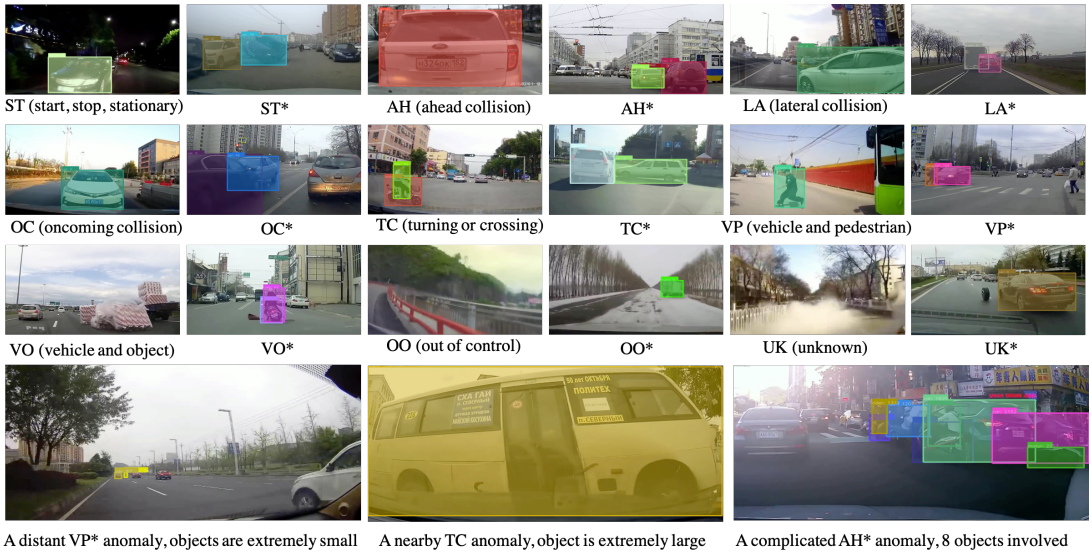


Figure 3.3: DoTA Samples. Spatial annotations are shown as shadowed bounding boxes. Short anomaly category labels with * indicate non-ego anomalies.

(a) Duration distribution

(b) Anomaly category distribution

(c) Object categories

(d) # anomalous objects

(e) Ego-car involvement

Figure 3.4: DoTA dataset statistics.

with different viewpoints are visually distinct, as shown in Fig. 3.3. Thus we split each category into ego-involved and non-ego (marked with *), resulting in 18 categories total. Sometimes the category can be ambiguous, particularly when one anomaly is followed by another. For example, an oncoming out-of-control (OO*) vehicle might result in an oncoming collision (OC) with the ego vehicle. In such cases, we annotate anomaly category as the dominant one in a video, typically the longer-lasting anomaly. Video distribution statistics are shown in Fig. 3.4(b).

CHAPTER IV

Object Trajectory and Behavior Prediction

4.1 Introduction

Understanding and predicting traffic objects (i.e. pedestrians, vehicles, etc.) movement behaviors is crucial for autonomous systems to safely navigate interactive environments. By correctly forecasting pedestrian trajectories, a robot or autonomous vehicle can plan safe and socially-aware paths in traffic [32, 33, 47, 115] and produce alarms about anomalous motions (e.g., crashes or near collisions) [46, 44, 1, 116, 117]. Early work in trajectory prediction often assumed a deterministic future, where only one trajectory is predicted for each object given past observations [118, 29, 30]. Despite that strict traffic rules and physical constraints apply to on-road objects, they can still move with a high degree of stochasticity so multiple plausible and distinct future behaviors can exist [51, 119]. Recent studies [120, 54, 121, 58, 48] have shown predicting a distribution of multiple potential future trajectories (i.e., multi-modal prediction) rather than a single best trajectory can more accurately model future motions of pedestrians.

Recurrent neural networks (RNNs), notably long short-term memory networks (LSTMs) and gated recurrent units (GRUs), have demonstrated success in trajectory prediction [33, 45, 7, 12]. In ego-centric view, visual and optical flow features provide reach information on object appearance and motion. In this Chapter, we first present the Future Object Localization (FOL), a multi-stream RNN encoder decoder network that combines bounding box sequences, optical flow features and vehicle ego motions to predict future bounding box sequence. FOL encodes past trajectories and optical flow features using two RNN encoders and predicts the future trajectory from the fused hidden state with an RNN decoder. A third RNN stream is used to predict future ego motion which is used as an extra input to the trajectory decoder.

Most RNN based models recurrently predict future trajectories based on previous output thus their performance tends to deteriorate rapidly over time ($>$ 560 ms) [119, 122]. In this Chapter, we propose to address this problem with a novel goal-conditioned bi-directional trajectory predictor, named *BiTraP*. BiTraP first estimates future goals (end-points of the future trajectories) of pedestrians and then predicts trajectories by combining forward passing from current position and backward passing from estimated goals. We believe that predicting goals can improve long-term trajectory predictions, as pedestrians in real world often have desired goals and plan paths to reach these goals [63]. Compared to existing goal-conditioned methods [63, 59, 61] where goals were used as an input to a forward decoder, BiTraP takes goals as the starting position of a backward decoder and predicts future trajectories from two directions, thus mitigating the accumulated error over longer prediction horizons.

Recently, generative models such as the generative adversarial network (GAN) [51] and conditional variational autoencoder (CVAE) [123, 54], were developed to predict multi-modal distributions of future trajectories. Our BiTraP model predicts multi-modal trajectories based on CVAE which learns target future trajectory distributions conditioned on the observed past trajectories through a stochastic latent variable. The two most common forms of the latent variable follow either a Gaussian distribution or a categorical distribution, resulting in either a non-parametric target distribution [54, 63] or a parametric target distribution model such as a Gaussian Mixture Model (GMM) [58, 48]. There has been limited research on how latent variable distributions impact predicted multi-modal trajectories. To fill this gap, we conducted extensive comparison studies using two variations of our BiTraP method: a non-parametric model using Gaussian latent variables (BiTraP-NP) and a GMM model using categorical latent variables (BiTraP-GMM). We implemented two types of loss functions, best-of-many (BoM) L2 loss [124] and negative log-likelihood (NLL) loss [48] to evaluate different predicted trajectory behaviors (e.g., spread and diversity). We show that latent variable distribution choices are closely related to the diversity of predicted distributions, which provides guidance for selecting trajectory predictors for robot navigation and collision avoidance systems.

This chapter presents methods to model two typical traffic participant types: vehicle and pedestrian. This Chapter first introduces FOL, a multi-stream RNN encoder decoder networks for vehicle trajectory prediction in ego-centric driving videos [7] in Section 4.2. Following FOL, a bi-directional model for multi-modal trajectory prediction (BiTraP) [49] is introduced in Section 4.3. The pedestrian action and intent

detection is then discussed and corresponding experiment results are presented in Section 4.4, followed by a conclusion section in Section 4.5.

4.2 Future Object Localization (FOL)

We proposed a FOL problem which predicts the long-term ($\geq 1s$) future locations and scales of vehicles in image coordinates. Consider a vehicle visible in the egocentric field of view, and let its past bounding box trajectory be $\mathbf{X} = \{X_{t-\tau+1}, X_{t-\tau+2}, ..., X_t\}$, where $X_t = [c_t^x, c_t^y, w_t, h_t]$ is the bounding box of the vehicle at time t (i.e., its center location and width and height in pixels, respectively). Similarly, let the future bounding box trajectory be given by $\mathbf{Y} = \{Y_{t+1}, Y_{t+2}, ..., Y_{t+\delta}\}$. Given image evidence observed from the past τ frames, $\mathbf{O} = \{O_{t-\tau+1}, O_{t-\tau+2}, ..., O_t\}$, and its corresponding past bounding box trajectory $\mathbf{X}$, our goal is to predict $\mathbf{Y}$. We designed a two-stream encoder-decoder architecture [7] for FVL as shown in Fig. 4.1. Our method can be generalized to predict trajectories of other traffic participants. Therefore we call it future object localization (FOL) and use this term through the paper.

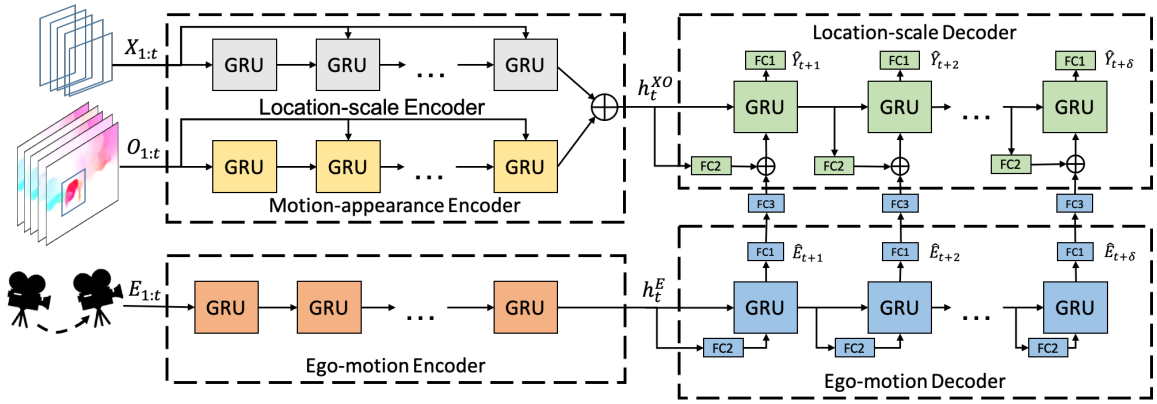


Figure 4.1: Future object localization (FOL) framework.

4.2.1 Two-stream Encoder

Location-Scale Encoding One straightforward approach to predict the future location of an object is to extrapolate a future trajectory from the past. However, in perspective images, physical object location is reflected by both its pixel location and scale. For example, a vehicle located at the center of an image could be a nearby lead vehicle or a distant vehicle across the intersection, and such a difference could cause a completely different future motion. Therefore, this paper predicts both the location

and scale of participant vehicles, i.e., their bounding boxes. The scale information is also able to represent depth (distance) as well as vehicle orientation, given that distant vehicles tend to have smaller bounding boxes and crossing vehicles tend to have larger aspect ratios.

Motion-Appearance Encoding. Another important cue for predicting a vehicle's future location is pixel-level information about motion and appearance. Optical flow is widely used as a pattern of relative motion in a scene. For each feature point, optical flow gives an estimate of a vector $[u, v]$ that describes its relative motion from one frame to the next caused by the motion of the object and the camera. Compared to sparse optical flow obtained from traditional methods such as Lucas-Kanade [125], dense optical flow offers an estimate at every pixel, so that moving objects can be distinguished from the background. Also, dense optical flow captures object appearance changes, since different object pixels may have different flows, as shown in the left part of Fig. 4.1. In this paper, object vehicle features are extracted by cropping and resizing operation using bilinear interpolation from the optical flow map. The cropping region is expanded from the bounding box to contain contextual information around the object, so that its relative motion with respect to the environment is also encoded. The resulting relative motion vector is represented as $O_t = [u_1, v_1, u_2, v_2, ...u_n, v_n]_t$, where n is the size of the pooled region.

We use two encoders for temporal modeling of each input stream and apply the late fusion method:

$$h_t^X = GRU_X(\phi_X(X_{t-1}), h_{t-1}^X; \theta_X) \tag{4.1a}$$

$$h_t^O = GRU_O(\phi_O(O_{t-1}), h_{t-1}^O; \theta_O) \tag{4.1b}$$

$$\mathcal{H} = \phi_{\mathcal{H}}(Average(h_{t_0}^X, h_{t_0}^O)) \tag{4.1c}$$

where GRU represents the gated recurrent units [126] with parameter θ, $\phi(\cdot)$ are linear projections with ReLU activations, and h_t^x and h_t^o are the hidden state vectors of the GRU models at time t.

4.2.2 Ego-Motion Cue

Ego-motion information of the moving camera has been shown to be necessary for accurate future object localization [7, 42]. In this work, the ego-motion is represented by 2D rotation matrices $R_t^{t+1} \in \mathbb{R}^{2\times2}$ and translation vectors $T_t^{t+1} \in \mathbb{R}^2$ [43], which together describe the transformation of the camera coordinate frame from time t to

$t+1$. The relative, pairwise transformations between frames can be composed to estimate transformations across the prediction horizon from the current frame:

$$R_{t_0}^{t_0+i} = \prod_{t=t_0}^{t_0+i-1} R_t^{t+1} \tag{4.2a}$$

$$T_{t_0}^{t_0+i} = T_{t_0}^{t_0+i-1} + R_{t_0}^{t_0+i-1} T_{t_0+i-1}^{t_0+i} \tag{4.2b}$$

The rotation and translation matrix are then converted to a ego-motion vector $E_t = [\psi_{t_0}^t, x_{t_0}^t, z_{t_0}^t]$, where $t > t_0$, $\psi_{t_0}^t$ is the yaw angle extracted from $R_{t_0}^t$, and $x_{t_0}^t$ and $z_{t_0}^t$ are translations from the coordinate frame at time t_0. We use a right-handed coordinate fixed to ego vehicle, where vehicle heading aligns with positive x. Estimated future motion is then used as input to the trajectory decoding model.

We predict the future ego-motion by using a separate RNN encoder-decoder module to encode ego-position change vector $E_t - E_{t-1}$ and decode future ego-position changes $\mathbf{E} = \{\hat{E}_{t+1} - E_t, \hat{E}_{t+2} - E_t, ..., \hat{E}_{t+\delta} - E_t\}$. We use the change in ego-position to eliminate accumulated odometry errors. The output $\mathbf{E}$ is then combined with the hidden state of the location-scale decoder to form the input into the next time step.

4.2.3 Trajectory Decoder

We use another GRU for decoding future bounding boxes. The decoder hidden state is initialized from the final fused hidden state of the past bounding box encoder and the optical flow encoder:

$$h_{t+1}^Y = GRU_Y(f(h_t^Y, E_t), h_t^Y; \theta_Y) \tag{4.3a}$$

$$Y_{t+i} - X_t = \phi_{out}(h_{t+i}^Y) \tag{4.3b}$$

$$f(h_t^Y, E_t) = Average(\phi_Y(h_t^Y), \phi_e(E_t)) \tag{4.3c}$$

where h_t^Y is the decoder's hidden state, $h_t^Y = \mathcal{H}$ is the initial hidden state of the decoder, and $\phi(\cdot)$ are linear projections with ReLU activations applied for domain transfer. Instead of directly generating future bounding boxes $\mathbf{Y}$, our RNN decoder generates the relative location and scale of the future bounding box from the current frame as in (4.3b), similar to [43]. In this way, model output is a relative trajectory which improves the performance.

4.2.4 Experiments on HEV-I and KITTI Dataset

Baselines and Ablations. We compare the performance of the proposed method with several baselines: 1) *Linear Regression (Linear)* extrapolates future bounding boxes by assuming the location and scale change are linear; 2) *Constant Acceleration (ConstAccel)* assumes the object has constant horizontal and vertical acceleration in the camera frame, i.e. that the second-order derivatives of X are constant values; 3) *Conv1D* is adapted from [43], by replacing the location-scale and pose input streams with past bounding boxes and dense optical flow. To evaluate the contribution of each component of our model, we also implemented multiple simpler baselines for ablation studies: 1) *RNN-ED-X* is an RNN encoder-decoder with only past bounding boxes as inputs; 2) *RNN-ED-XE* builds on *RNN-ED-X* but also incorporates future ego-motion as decoder inputs; 3) *RNN-ED-XO* is a two-stream RNN encoder-decoder model with past bounding boxes and optical flow as inputs; 4)*RNN-ED-XOE* is our best model as shown in Fig.4.1 with awareness of future ego-motion.

Evaluation Metrics. To evaluate location prediction, we use final displacement error (FDE) and average displacement error (ADE) [32, 43], where ADE emphasizes more on the overall prediction accuracy along the horizon. To evaluate bounding box prediction, we propose a final intersection over union (FIOU) metric that measures overlap between the predicted bounding box and ground truth at the final frame.

4.2.4.1 Results on HEV-I Dataset

Quantitative Results. As shown in Table 4.1, we split the testing dataset into easy and challenging cases based on the FDE performance of the *ConstAccel* baseline. A sample is classified as easy if the *ConstAccel* achieves FDE lower than the average FDE (58.00), otherwise it is classified as challenging. Intuitively, easy cases include target vehicles that are stationary or whose future locations can be easily propagated from the past, while challenging cases usually involve diverse and intense motion, e.g. the target vehicle suddenly accelerates or brakes. In evaluation, we report the results of easy and challenging cases, as well as the overall results on all testing samples.

Our best method (*RNN-ED-XOE*) significantly outperforms naive baselines including *Linear* and *ConstAccel* on all cases (FDE of **24.92** vs. 72.37 vs. 58.00). It also improves about 15% from the state-of-the-art *Conv1D* baseline. The improvement on challenging cases is more significant since future trajectories are complex and temporal modeling is more difficult. To more fairly compare the capability of RNN-

Table 4.1: Quantitative results of proposed methods and baselines on HEV-I dataset with metrics FDE(↓)/ADE(↓)/FIOU(↑).

Models	Easy Cases	Challenging Cases	All Cases
Linear	31.49 / 17.04 / 0.68	107.93 / 56.29 / 0.33	72.37 / 38.04 / 0.50
ConstAccel	20.82 / 13.86 / 0.74	90.33 / 49.06 / 0.35	58.00 / 28.05 / 0.53
Conv1D [43]	18.84 / 12.09 / 0.75	37.95 / 20.97 / 0.64	29.06 / 16.84 / 0.69
RNN-ED-X	23.57 / 11.96 / 0.74	43.15 / 22.24 / 0.60	34.04 / 17.46 / 0.67
RNN-ED-XE	22.28 / 11.60 / 0.74	42.27 / 22.39 / 0.61	32.97 / 17.37 / 0.67
RNN-ED-XO	17.45 / 8.68 / 0.78	32.61 / 16.72 / 0.66	25.56 / 12.98 / 0.72
RNN-ED-XOE	**16.72** / **8.52** / **0.80**	**32.05** / **16.63** / **0.66**	**24.92** / **12.86** / **0.73**

ED and convolution-deconvolution models, we compare *RNN-ED-XO* with *Conv1D*. These two methods use the same features as inputs to predict future vehicle bounding boxes, but rely on different temporal modeling frameworks. The results (FDE of **25.56** vs 29.06) suggest that the RNN-ED architecture offers better temporal modeling compared to *Conv1D*, because the convolution-deconvolution model generates future trajectory in one shot while the RNN-ED model generates a new prediction based on the previous hidden state. Ablation studies also show that dense optical flow features are essential to accurate prediction of future bounding boxes, especially for challenging cases. The FDE is reduced from 34.04 to 25.56 by adding optical flow stream (*RNN-ED-XO*) to *RNN-ED-X* model. By using future ego-motion, performance can be further improved as shown in the last row of Table 4.1.

Qualitative Results. Fig. 4.2(a) shows four sample results of our best model (in green) and the *Conv1D* baseline (in blue). Each row represents one test sample and each column corresponds to each time step. The past and prediction views are separated by the yellow vertical line. Example (a) shows a case where the initial bounding box is noisy because it is close to the image boundary, and our results are more accurate than those of *Conv1D*. Example (b) shows how our model, with awareness of future ego-motion, can predict object future location more accurately while the baseline model predicts future location in the wrong direction. Examples (c) and (d) show that for a curved or long trajectory, our model provides better temporal modelling than *Conv1D*. These results are consistent with our evaluation observations.

Failure Cases. Although our proposed method generally performs well, there are still limitations. Fig.4.2(b) (a) shows a case when the ground truth future path is curved due to uneven road surface, which our method fails to consider. In Fig.4.2(b) (b), the target vehicle is occluded by pedestrians moving in the opposite direction,

which creates misleading optical flow that leads to an inaccurate bounding box (especially in $t = t_0$ frame). Future work could avoid this type of error by better modeling the entire traffic scene as well as relations between traffic participants.

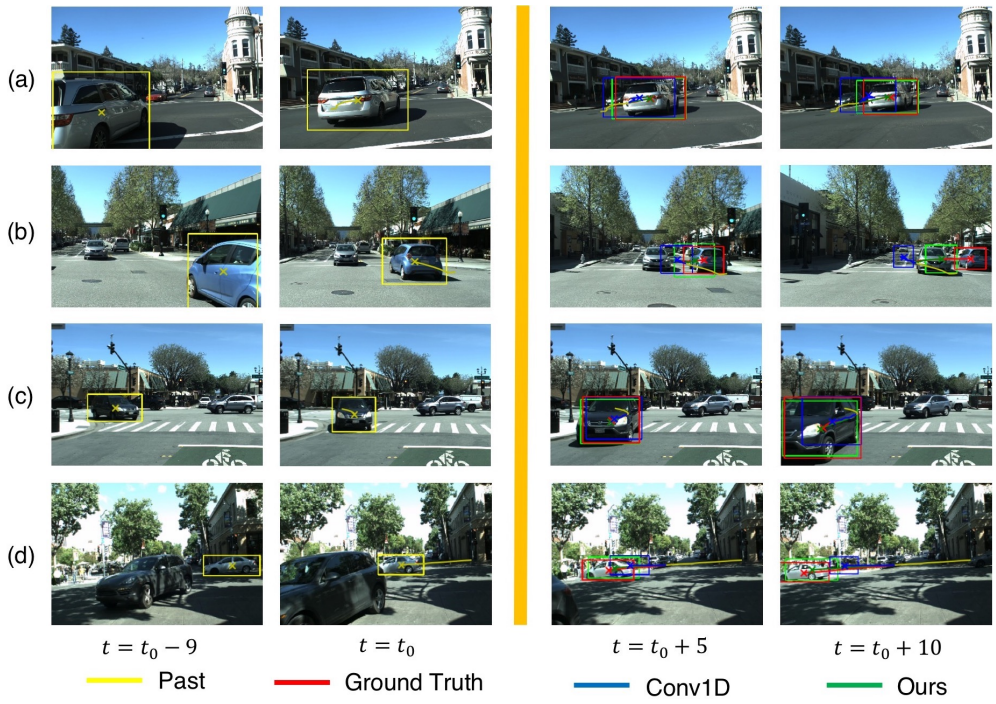
4.2.4.2 Results on KITTI Dataset

Table 4.2: Quantitative results on KITTI dataset. We compare our best model with baselines for simplicity.

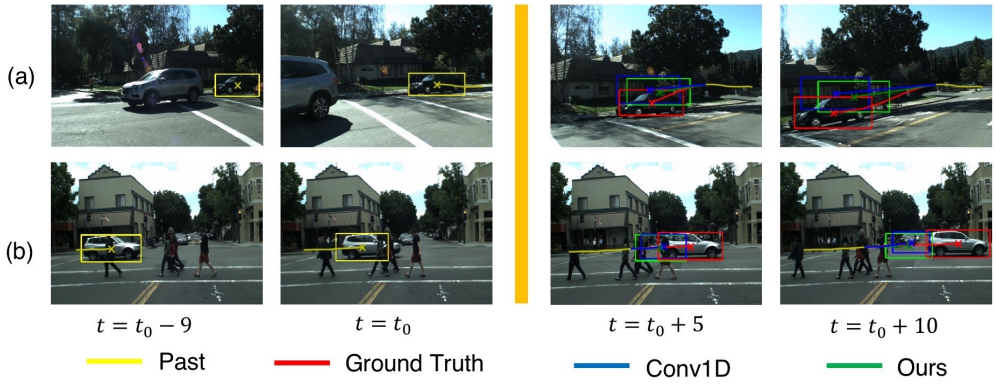
Models	FDE ↓	ADE ↓	FIOU ↑
Linear	78.19	38.21	0.33
ConstAccel	55.66	25.78	0.39
Conv1D [43]	44.13	24.38	0.49
Ours	**37.11**	**17.88**	**0.53**

We also evaluate our method on a 38-video subset of the KITTI raw dataset, including city, road and residential scenarios. Compared to HEV-I, the road surface of KITTI is more uneven and vehicles are mostly parked on the side of the road with occlusions. Another difference is that in HEV-I, the ego-vehicle often stops at intersections to yield to other vehicles, resulting in static samples with no motion at all. We did not remove static samples from the dataset since predicting a static object is also valuable.

To evaluate our method on KITTI, we first generate the input features following the same process of HEV-I dataset, resulting in $\sim$ 8000 training and $\sim$ 2700 testing samples. Performance of baselines and our best model are shown in Table 4.2. Both learning-based models are trained for 40 epoches and the best models are selected. The results show that our method outperforms all baselines including the state-of-the-art Conv1D (FDE of **37.11** vs 78.19 vs 55.66 vs 44.13). We also observe that both learning-based methods did not perform as well as they did on HEV-I. One possible reason is that KITTI is much smaller so that the models are not fully trained. In general, we conclude that the use of the proposed framework results in more robust future vehicle localization across different datasets.



(a) Qualitative results on HEV-I dataset (better in color).



(b) Failure cases on HEV-I dataset (better in color).

4.3 Bi-direction Trajectory Prediction with Goal Estimation

In this section, we introduce Bi-directional Trajectory Prediction (BiTraP), a multi-modal trajectory prediction network with estimation based on a conditional variational autoencoder (CVAE).

4.3.1 Preliminaries

A CVAE is a conditional generative model designed to output target data Y based on latent variable Z and observation X [123]. A CVAE consists of three modules: a **conditional prior network** $p_\theta(Z|X)$ to model latent variable Z conditioned on observation X, a **recognition network** $q_\phi(Z|X,Y)$ to capture dependencies between Z and target Y, and a **generation network** $p_\psi(Y|X,Z)$ to generate the target Y, where ϕ, θ, and ψ represent network parameters. Stochastic latent variable $Z \in \mathbb{R}^d$ is sampled from a pre-defined distribution format such as a Gaussian distribution. The CVAE samples Z and generates target Y conditioned on observation X. The objective of a typical CVAE model is to maximize its variational lower bound

$$\max_{\theta,\phi,\psi} \mathbb{E}_{q_\phi(Z|X,Y)}\Big[\log p_\psi(Y|X,Z)\Big] - KL\Big(q_\phi(Z|X,Y)||p_\theta(Z|X)\Big), \tag{4.4}$$

where the first term maximizes the expectation of the log-likelihood of the target in the predicted distribution; the K-L (Kullback–Leibler) divergence term minimizes the difference between the recognition network and the conditional prior network. We designed a modified CVAE with two generation networks and optimize both networks end-to-end.

Our BiTraP model performs goal-conditioned multi-modal bi-directional trajectory prediction in either first-person view (FPV) or bird's eye view (BEV). Let $\mathbf{X}_t = [X_{t-\tau+1}, X_{t-\tau+2}, ..., X_t]$ denote observed past trajectory at time t, where X_t is bounding box location and size (x, y, w, h) in pixels for FPV [7, 12] and (x, y) position in meters for BEV [48]. Given $\mathbf{X}_t$, we first estimate goal G_t of the person then predict future trajectory $\mathbf{Y}_t = [Y_{t+1}, Y_{t+2}, ..., Y_{t+\delta}]$, where τ and δ are observation and prediction horizons, respectively. Define goal $G_t = Y_{t+\delta}$ as the future trajectory endpoint, which is given in training and unknown in testing. We adopt a CVAE model to realize multi-modal goal and trajectory prediction. BiTraP contains four sub-modules: **conditional prior network** $p_\theta(Z|\mathbf{X}_t)$ to model latent variable Z from observations, **recognition network** $q_\phi(Z|\mathbf{X}_t, \mathbf{Y}_t)$ to capture dependencies between Z and $\mathbf{Y}_t$, **goal generation network** $p_\omega(G_t|\mathbf{X}_t, Z)$, and **trajectory generation**

network $p_\psi(\mathbf{Y}_t|\mathbf{X}_t, G_t, Z)$ where ϕ, θ, ω and ψ represent network parameters. Either parametric or non-parametric models can be used to design networks p_ψ and p_ω for CVAE. Non-parametric models do not assume the distribution format of target $\mathbf{Y}_t$ but learn it implicitly by learning the distribution of Z. Parametric models assume a known distribution format for $\mathbf{Y}_t$ and predict distribution parameters. We design non-parametric and parametric models in Sections 4.3.2 and 4.3.4, respectively, and explain different loss functions to train these models in Sections 4.3.3 and 4.3.5.

4.3.2 BiTraP with Non-parametric (NP) Distribution

BiTraP-NP is built on a standard recurrent neural network encoder-decoder (RNN-ED) based CVAE trajectory predictor as in [54, 63, 124, 56], except it predicts goal first and then predict trajectories leveraging goals. Following previous work, we assume Gaussian latent variable $Z \sim \mathcal{N}(\mu_Z, \sigma_Z)$ and a non-parametric target distribution format. Fig. 4.3 shows the network architecture of BiTraP-NP.

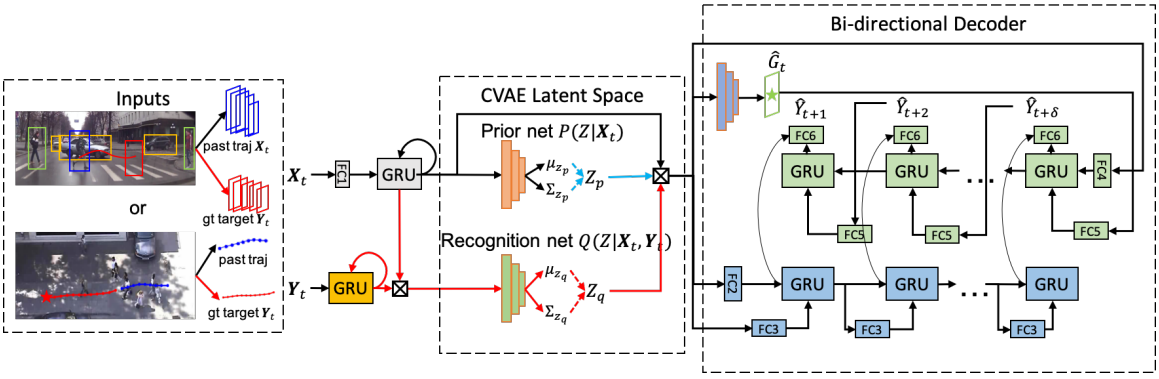


Figure 4.3: Overview of our BiTraP-NP network. Red, blue and black arrows show processes that appear in training only, inference only, and both training and inference, respectively.

Encoder and goal estimation. First, observed trajectory $\mathbf{X}_t$ is processed by a gated-recurrent unit (GRU) encoder network to obtain encoded feature vector h_t. In training, ground truth target $\mathbf{Y}_t$ is encoded by another GRU yielding h_{Y_t}. Recognition network $q_\phi(Z|\mathbf{X}_t, \mathbf{Y}_t)$ takes h_t and h_{Y_t} to predict distribution mean μ_{Z_q} and covariance Σ_{Z_q} which capture dependencies between observation and ground truth target. Prior network $p_\theta(Z|\mathbf{X}_t)$ assumes no knowledge about target and predicts μ_{Z_p} and Σ_{Z_p} using h_t only. Kullback–Leibler divergence (KLD) loss between $\mathcal{N}(\mu_{Z_p}, \Sigma_{Z_p})$ and $\mathcal{N}(\mu_{Z_q}, \Sigma_{Z_q})$ is optimized so that dependency between $\mathbf{Y}_t$ and $\mathbf{X}_t$ is implicitly learned by the prior network. Latent variable Z is sampled from $\mathcal{N}(\mu_{Z_q}, \Sigma_{Z_q})$ and concatenated with h_t to predict multi-modal goals $\hat{G}_t$ with goal generation network

$p_\omega(G_t|\mathbf{X}_t, Z)$. In testing, we directly draw multiple samples from $\mathcal{N}(\mu_{Z_p}, \Sigma_{Z_p})$ and concatenate h_t to predict estimated goals $\hat{G}_t$. We use 3-layer multi-layer perceptrons (MLPs) for *prior*, *recognition* and *goal generation networks*.

Trajectory Decoder. Predicted goals $\hat{G}_t$ are used as inputs to a bi-directional *trajectory generation network* $p_\psi(\mathbf{Y}_t|\mathbf{X}_t, \hat{G}_t, Z)$, the trajectory decoder, to predict multi-modal trajectories. BiTraP's decoder contains forward and backward RNNs. The forward RNN is similar to a regular RNN decoder (Eq. (4.5)) except its output is not transformed to trajectory space. The backward RNN is initialized from encoder hidden state h_t. It takes estimated goal $\hat{Y}_{t+\delta} = \hat{G}_t$ as the initial input (Eq. (4.6)) and propagates from time $t+\delta$ to $t+1$ so backward hidden state is updated from the goal to the current location. Forward and backward hidden states for the same time step are concatenated to predict the final trajectory way-point at that time (Eq. (4.7)). These steps can be formulated as

$$h^f_{t+1} = GRU_f(h^f_t, W^i_f h^f_t + b^i_f), \tag{4.5}$$

$$h^b_{t+\delta-1} = GRU_b(h^b_{t+\delta}, W^i_b \hat{Y}_{t+\delta} + b^i_b), \tag{4.6}$$

$$\hat{Y}_{t+\delta-1} = W^o_f h^f_{t+\delta-1} + W^o_b h^b_{t+\delta-1} + b^o, \tag{4.7}$$

where, f, b, i and o indicate "forward", "backward", "input" and "output" respectively, and h^f_t and $h^b_{t+\delta}$ are initialized by passing h_t through two different fully-connected networks.

4.3.3 Residual Prediction and BoM Loss for BiTraP-NP

Instead of directly predicting future location [12] or integrating from predicted future velocity [48], BiTraP-NP predicts change with respect to the current location based on residuals $\hat{Y}_{t+\delta} = Y_{t+\delta} - X_t$. There are two advantages of residual prediction. First, it assures the model will predict the trajectory starting from the current location, providing smaller initial loss than predicting location from scratch. Second, the residual target can be less noisy than the velocity target due to the fact that trajectory annotation is not always accurate. Standard CVAE loss includes NLL loss of the predicted distribution which is not applicable to NP methods due to their unknown distribution format. L2 loss between predictions and targets can be used as a substitution [54]. To further encourage diversity in multi-modal prediction, we use best-of-many (BoM) L2 loss as in [124]. The final loss function for BiTraP-NP is a combination of the goal L2 loss, the trajectory L2 loss and the KL-divergence loss

between prior and recognition networks, written as

$$L_{NP} = \min_{i \in N} \left\| G_t - X_t - \hat{G}_t^i \right\| + \min_{i \in N} \sum_{\tau=t+1}^{t+\delta} \left\| Y_\tau - X_t - \hat{Y}_\tau^i \right\| + KLD, \tag{4.8}$$

where $\hat{G}_t$ and $\hat{Y}_\tau$ are the predicted goal and trajectory waypoints with respect to current position X_t.

4.3.4 BiTraP with GMM Distribution

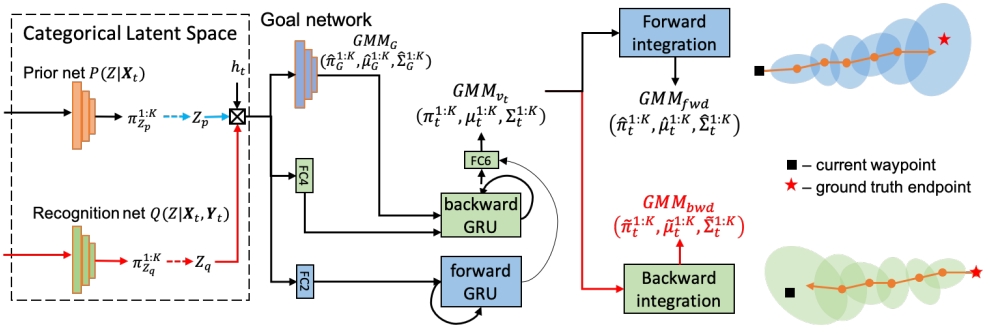


Figure 4.4: Latent space sampling and decoder modules of BiTraP-GMM. The ellipse shows one of K GMM components at each timestep. The rest of the network is the same as BiTraP-NP in Fig. 4.3.

Parametric models predict trajectory distribution parameters instead of trajectory coordinates. BiTraP-GMM is our parametric variation of BiTraP assuming a GMM for the trajectory goal and at each way-point [58, 48]. Let $p(Y_{t+\delta})$ denote a K-component GMM at time step $t+\delta$. We assume $p(Y_{t+\delta}) = \sum_{i=1}^{K} \pi_i \mathcal{N}(Y_{t+\delta} | \mu_{t+\delta}^i, \Sigma_{t+\delta}^i)$, where each Gaussian component can be considered the distribution of one trajectory modality. Mixture component weights π_i sum to one thus form a categorical distribution. Each π_i indicates the probability (confidence) that a person's motion belongs to that modality. We design latent vector Z as a categorical (Cat) variable $Z \sim Cat(K, \pi_{1:K})$ parameterized by GMM component weights $\pi_{1:K}$ rather than separately-computed parameters. Similar to BiTraP-NP, we use three 3-layer MLPs for the *prior*, *recognition* and *goal generation networks*, and a bi-directional RNN decoder for the *trajectory generation network*. Instead of directly predicting trajectory coordinates, generation networks of BiTraP-GMM estimate the $\mu_{t+\delta}^i$ and $\Sigma_{t+\delta}^i$ of the ith Gaussian components at time $t+\delta$. In training, we sample one Z from each category to ensure all trajectory modalities are trained. In testing, we sample Z from $Cat(K, \pi_{1:K})$ so it is more probable to sample from high-confidence trajectory modalities.

4.3.5 Bi-directional NLL Loss for BiTraP-GMM

Similar to [48], our BiTraP-GMM models the pedestrian velocity distribution as a GMM at each time step. The velocity GMM is then integrated forward to obtain the GMM distribution of trajectory waypoints $Y_{t+\delta}$ as shown by blue blocks in Fig. 4.4. We assume linear dynamics for pedestrian and use a single integrator as in Eq. (4.9). The loss function is then the summation of negative log-likelihood (NLL) of the ground truth future waypoints over the prediction horizon, formulated as

$$GMM_{Y_{t+\delta}}(\hat{\pi}_{t+\delta}^{1:K}, \hat{\mu}_{t+\delta}^{1:K}, \hat{\Sigma}_{t+\delta}^{1:K}) = X_t + \int_t^{t+\delta} GMM_{v_\tau}(\pi_\tau^{1:K}, \mu_\tau^{1:K}, \Sigma_\tau^{1:K}) d\tau, \tag{4.9}$$

$$NLL_{fwd} = \sum_{\tau=t}^{t+\delta} -\log p(Y_\tau | \hat{\pi}_\tau^{1:K}, \hat{\mu}_\tau^{1:K}, \hat{\Sigma}_\tau^{1:K}), \tag{4.10}$$

where $\pi_\tau^{1:K}$, $\mu_\tau^{1:K}$, $\Sigma_\tau^{1:K}$ are velocity GMM parameters at time $\tau \in [t+1, t+\delta]$, and the $\hat{\cdot}$ symbol indicates location GMM parameters obtained from integration. $p(\cdot)$ is the GMM probability density function. Such an NLL emphasizes earlier waypoints along the prediction horizon because a waypoint at time $t+1$ is used in integration results over $t+2$, $t+3$, ..., while these later waypoints are not used when computing $t+1$. This goes against our proposed idea which is to leverage a bi-directional temporal model. Therefore, we compute bi-directional NLL loss with reverse integration from the goal, formulated as

$$GMM'_{Y_t}(\tilde{\pi}_t^{1:K}, \tilde{\mu}_t^{1:K}, \tilde{\Sigma}_t^{1:K}) = G_t - \int_{t+\delta}^{t} GMM_{v_\tau}(\pi_\tau^{1:K}, \mu_\tau^{1:K}, \Sigma_\tau^{1:K}) d\tau, \tag{4.11}$$

$$NLL_{bwd} = \sum_{\tau=t+\delta}^{t} -\log p'(Y_\tau | \tilde{\pi}_\tau^{1:K}, \tilde{\mu}_\tau^{1:K}, \tilde{\Sigma}_\tau^{1:K}). \tag{4.12}$$

where $p(\cdot)'$ is the backward GMM probability density function, the $\tilde{\cdot}$ symbol indicates backward location GMM parameters. The final loss function for BiTraP-GMM can be written as

$$L_{GMM} = -\log p_G(G_t | \hat{\pi}_G^{1:K}, \hat{\mu}_G^{1:K}, \hat{\Sigma}_G^{1:K}) + NLL_{fwd} + NLL_{bwd} + KLD, \tag{4.13}$$

where the first term is NLL loss of the goal estimation, $+NLL_{fwd}$ and NLL_{bwd} are computed from forward and backward integration, the KLD term is the KL-divergence similar to Eq. (4.8).

4.3.6 Implementation Details

In this section, we empirically evaluate BiTraP-NP and BiTraP-GMM models on both first-person view (FPV) and bird's eye view (BEV) trajectory prediction datasets. We also provide a comparative study and discussion on the effects of model and loss selection.

Two FPV datasets, Joint Attention for Autonomous Driving (JAAD) [75] and Pedestrian Intention Estimation (PIE) [12], and two benchmark BEV datasets, ETH [27] and UCY [28], were used in our experiments. JAAD contains 2,800 pedestrian trajectories captured from dash cameras annotated at 30Hz. PIE contains 1,800 pedestrian trajectories also annotated at 30Hz, with longer trajectories and more comprehensive annotations such as semantic intention, ego-motion and neighbor objects. ETH-UCY datasets contain five sub-datasets captured from down-facing surveillance cameras in four different scenes with 1,536 pedestrian trajectories annotated at 2.5Hz.

We used the standard training/testing splits of JAAD and PIE as in [12]. A 0.5-second (15 frame) observation length and 1.5-second (45 frame) prediction horizon were used for evaluation. For ETH-UCY, a standard leave-one-out approach based on scene was used per [51, 48]. We observed trajectories for 3.2 seconds (8 frames) and predicted the paths for the next 4.8 seconds (12 frames). We used hidden unit size 256 for all encoders and decoders in BiTraP across all datasets. All models were trained with batch size 128, learning rate (LR) 0.001, and an exponential LR scheduler [48] on a single NVIDIA TITAN XP GPU.

4.3.7 Experiments on JAAD and PIE Datasets

Baselines. We compare our results against the following baseline models: 1) Linear Kalman filter, 2) Vanilla LSTM model, 3) Bayesian-LSTM model (B-LSTM) [42], 4) PIE$_{traj}$, an attentive RNN encoder-decoder model, 5) PIE$_{full}$, a multi-stream attentive RNN model, by injecting ego-motion and semantic intention stream to PIE$_{traj}$, and 6) FOL-X [7], a multi-stream RNN encoder-decoder model using residual prediction. We also conducted an ablation study for a deterministic variation of our model (BiTraP-D), where the multi-modal CVAE module was removed.

Evaluation Metrics. Following [7, 12, 42], our BiTraP model was evaluated using: 1) bounding box Average Displacement Error (ADE), 2) box center ADE (C_{ADE}) and 3) box center Final Displacement Error (C_{FDE}) in squared pixels. For our multi-modal BiTraP-NP and BiTraP-GMM, we compute the best-of-20 results (the minimum ADE and FDE from 20 randomly-sampled trajectories), following [51, 48, 52]. We also

report the Kernel Density Estimation-based Negative Log Likelihood (KDE-NLL) metric for BiTraP-NP and BiTraP-GMM, which evaluates the NLL of the ground truth under a distribution fitted by a KDE on trajectory samples from each prediction model [48, 127]. For all metrics, lower values are better.

Results. Table 4.3 presents trajectory prediction results with JAAD and PIE datasets. Our deterministic BiTraP-D model shows consistently lower displacement errors across various prediction horizons than baseline methods such as PIE_{traj} and FOL-X indicating our goal estimation and bi-directional prediction modules are effective. Our BiTraP-D model, based only on past trajectory information, also outperforms the state-of-the-art PIE_{full}, which requires additional ego-motion and semantic intention annotations. Table 4.3 also shows that non-parametric multi-modal method BiTraP-NP performs better on displacement metrics while parametric method BiTraP-GMM performs better on the NLL metric. This difference illustrates the objectives of these methods: BiTraP-NP generates diverse trajectories, and one trajectory was optimized to have minimum displacement error, while BiTraP-GMM generates trajectory distributions with more similarity to the ground truth trajectory.

Table 4.3: Results on JAAD and PIE datasets. The center row shows deterministic baselines including our ablation model BiTraP-D; the bottom row shows our proposed multi-modal methods. NLL is not available for deterministic methods since they predict single trajectories. Lower values are better.

Methods	JAAD				PIE			
	ADE (0.5/1.0/1.5s)	C_{ADE} (1.5s)	C_{FDE} (1.5s)	NLL	ADE (0.5/1.0/1.5s)	C_{ADE} (1.5s)	C_{FDE} (1.5s)	NLL
Linear [12]	233/857/2303	1565	6111	-	123/477/1365	950	3983	-
LSTM [12]	289/569/1558	1473	5766	-	172/330/911	837	3352	-
B-LSTM [42]	159/539/1535	1447	5615	-	101/296/855	811	3259	-
FOL-X [7]	147/484/1374	1290	4924	-	47/183/584	546	2303	-
PIE_{traj} [12]	110/399/1280	1183	4780	-	58/200/636	596	2477	-
PIE_{full} [12]	-	-	-	-	-/-/556	520	2162	-
BiTraP-D	**93/378/1206**	**1105**	**4565**	-	**41/161/511**	**481**	**1949**	-
BiTraP-NP (20)	**38/94/222**	**177**	**565**	18.9	**23/48/102**	**81**	**261**	16.5
BiTraP-GMM (20)	153/250/585	501	998	**16.0**	38/90/209	171	368	**13.8**

Fig. 4.5 shows trajectory prediction results on sample frames from the PIE dataset. We observed that when a pedestrian intends to cross the street or change directions, the multi-modal BiTraP methods yield higher accuracy and more reasonable predictions than the deterministic variation. For example, as shown in Fig. 4.5(b), the deterministic BiTraP-D model (top row) can fail to predict the trajectory and the end-goal, where a pedestrian intends to cross the street in the future; the multi-modal BiTraP-NP model (bottom row) can successfully predict multiple possible future tra-

jectories, including one where the pedestrian is crossing the street matching ground truth intention. Similar observations can be made in other frames. This result indicates multi-modal BiTraP-NP can predict multiple possible futures, which could help a mobile robot or a self-driving car safely yield to pedestrians. Although BiTraP-NP samples diverse trajectories, it still predicts distribution with high likelihood around ground truth targets and low likelihood in other locations per Fig. 4.5(b)-4.5(d).

(a) (b) (c) (d)

Figure 4.5: Qualitative results of deterministic (top row) vs multi-modal (bottom row) bi-directional prediction. Past (dark blue), ground truth future (red) and predicted future (green) trajectories and final bounding box locations are plotted. In the bottom row, each BiTraP-NP likelihood heatmap fits a KDE over samples. The orange color indicates higher probability.

4.3.8 Experiments on ETH-UCY Datasets

Baselines. We compare our methods with five multi-modal baseline methods: S-GAN [51], SoPhie [52], S-BiGAT [53], PECNet [63] and Trajectron++ [48]. PECNet and Trajectron++ are most recent. PCENet is a goal-conditioned method using non-parametric distribution (thus directly comparable to our BiTraP-NP) while Trajectron++ uses a GMM trajectory distribution directly comparable to our BiTraP-GMM. Note that all baselines incorporate social information while our methods *fully focus on investigating trajectory modeling and do no require social information input.*

Evaluation Metrics. Following [51, 63, 52], we used best-of-20 trajectory ADE and FDE in meters as evaluation metrics. We also report Average and Final KDE-NLL (ANLL and FNLL) metrics as a supplement [127, 48] to evaluate the predicted trajectory and goal distribution.

Results. Table 4.4 shows the best-of-20 ADE/FDE results across all methods. We observed that BiTraP-NP outperforms the state-of-the-art goal based method

(PECNet) by a large margin ($\sim 12\%-51\%$), demonstrating the effectiveness of our bi-directional decoder module. BiTraP-NP also obtains lower ADE/FDE on most scenes ($\sim$ 12%-24% improvement) compared with Trajectron++. Our BiTraP-GMM model was trained using NLL loss, so it shows higher ADE/FDE results compared with BiTraP-NP. This is consistent with our FPV dataset observations in Section 4.3.7. Nevertheless, BiTraP-GMM still achieves similar or better results than PECNet and Trajectron++.

Table 4.4: Trajectory prediction results (ADE/FDE) on BEV ETH-UCY datasets. Lower is better.

Datasets	S-GAN [51]	SoPhie [52]	S-BiGAT [53]	PECNet [63]	Trajectron++ [48]	BiTraP-NP	BiTraP-GMM
ETH	0.81/1.52	0.70/1.43	0.69/1.29	0.54/0.87	0.43/0.86	**0.37/0.69**	0.40/0.74
Hotel	0.72/1.61	0.76/1.67	0.49/1.01	0.18/0.24	**0.12/0.19**	**0.12**/0.21	0.13/0.22
Univ	0.60/1.26	0.54/1.24	0.55/1.32	0.35/0.60	0.22/0.43	**0.17/0.37**	0.19/0.40
Zara1	0.34/0.69	0.30/0.63	0.30/0.62	0.22/0.39	0.17/0.32	**0.13/0.29**	0.14/**0.28**
Zara2	0.42/0.84	0.38/0.78	0.36/0.75	0.17/0.30	0.12/0.25	**0.10/0.21**	0.11/0.22
Average	0.58/1.18	0.54/1.15	0.48/1.00	0.29/0.48	0.21/0.39	**0.18/0.35**	0.19/0.37

To further evaluate predicted trajectory distributions, we report KDE-NLL results in Table 4.5. As shown, BiTraP-GMM outperforms Trajectron++ with lower ANLL and FNLL on *ETH*, *Univ*, *Zara1* and *Zara2* datasets. On *Hotel*, Trajectron++ achieves lower NLL values which may be due to the possible higher levels of inter-personal interactions than in other scenes. We observed improved ANLL/FNLL on *Hotel* (-1.88/0.27) when combining the BiTraP-GMM decoder with the interaction encoder in [48], consistent with our hypothesis.

Table 4.5: Average-NLL/Final-NLL (ANLL/FNLL) results on ETH-UCY datasets. Lower is better.

Datasets	S-GAN [51]	Trajectron++ [48]	BiTraP-NP	BiTraP-GMM
ETH	15.70/-	1.31/4.28	3.80/3.79	**0.96/3.55**
Hotel	8.10/-	**-1.94/0.25**	-0.41/1.26	-1.60/0.51
Univ	2.88/-	-1.13/2.13	-0.84/2.15	**-1.19/2.03**
Zara1	1.36/-	-1.41/1.83	-0.81/1.85	**-1.51/1.56**
Zara2	0.96/-	-2.53/0.50	-1.89/1.31	**-2.54/0.38**

We also computed KDE-NLL results for both Trajectron++ and BiTraP-GMM methods at each time step to analyze how BiTraP affects both short-term and longer-term (up to 4.8 seconds) prediction results. Per Fig. 4.6, BiTraP-GMM outperforms Trajectron++ with longer prediction horizons (after 1.2 seconds on *ETH*, *Univ*, *Zara1*, and *Zara2*). This shows the backward passing from the goal helps reduce error with longer prediction horizon.

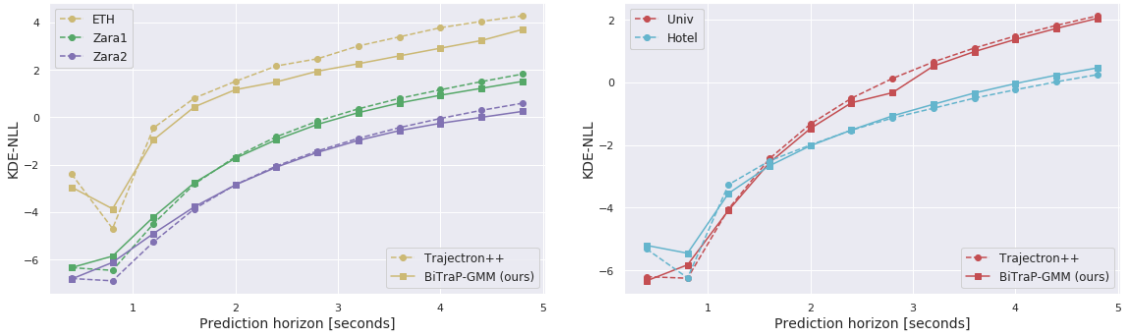
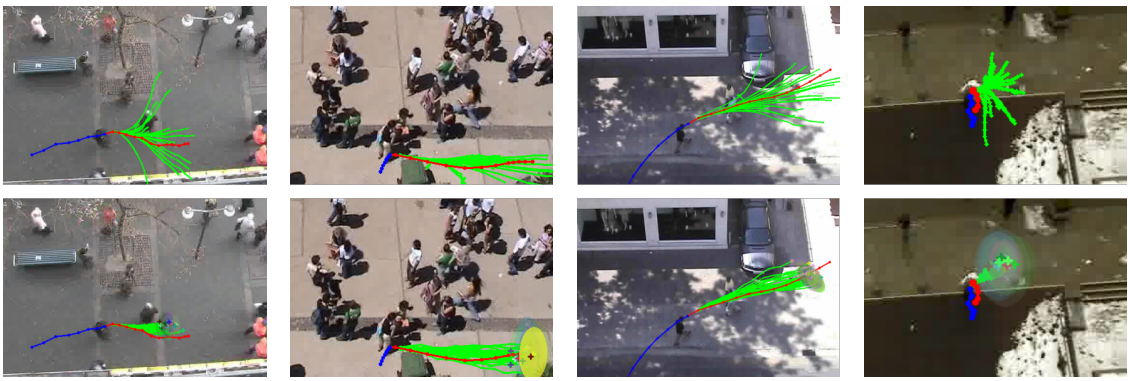


Figure 4.6: KDE-NLL results on the ETH-UCY dataset per timestep up to 4.8 seconds.

Fig. 4.7 shows qualitative examples of our predicted trajectories using the BiTraP-NP and BiTraP-GMM models. As shown, BiTraP-NP (top row) generates future possible trajectories with a wider spread (more diverse), while BiTraP-GMM generates more compact distributions. This is consistent with our quantitative evaluations as reported in Table 4.5, where the lower NLL results of BiTraP-GMM correspond to more compact trajectory distributions. To intuitively present model performance in collision avoidance and robot navigation, we conducted a robot path simulation experiment on ETH-UCY dataset and report collision related metrics in the supplementary material.



(a) Hotel (b) Univ (c) Zara2 (d) ETH

Figure 4.7: Visualizations of BiTraP-NP (first row) and BiTraP-GMM (second row). Twenty sampled future trajectories are plotted. For BiTraP-GMM, we also plot end-point GMM distributions as colored ellipses. Size indicates component Σ_k and transparency indicates component weight π_k.

Ablation study. We conducted two ablation experiments. To show bi-directional decoder effectiveness, we removed the backward decoder from BiTraP-NP and compared its performance with the original BiTraP-NP model (w/o backward (TraP-NP) vs w/ backward). To show bi-directional loss effectiveness in BiTraP-GMM, we com-

pared two BiTraP-GMM models trained with forward loss and bi-directional loss (w/o bi-loss vs w/ bi-loss). A comparison of ADE/FDE and ANLL/FNLL results is presented in Table 4.6. Using a bi-directional decoder (BiTraP-NP) improves ADE/FDE by 10%-28% (ANLL/FNLL by ∼0.4) from the model without backward decoder. By using bi-directional loss (bi-loss), the ADE/FDE of BiTraP-GMM model improves by 5-18% on ETH, and ANLL/FNLL improves by ∼0.25.

Computational time. We provide model inference time of Social GAN [51], Trajectron++ [48] and our BiTraP-NP and BiTraP-GMM models in Table 4.7. Trajectron++ generates scene graphs before running the model so computation time is summed over scene graph generation and model inference. For Social GAN and our method, total time consists of model inference time only. We show computational times for number of samples 20 and 2000. Time differences of BiTraP models between the two numbers are ∼ 3ms, while the difference of S-GAN is extremely large as it generates samples one-by-one. BiTraP-GMM is ∼ 3ms slower than Trajectron++, not significant since both methods run at ∼ 70ms per frame (∼ 14 FPS) on average. BiTraP-NP is about 8x faster than Trajectron++ and BiTraP-GMM since it does not fit a GMM model or perform dynamic integration. Adding the bi-directional decoder slows inference by ∼ 3ms (TraP-NP vs BiTraP-NP). All experiments are conducted on the same machine used for training.

Table 4.6: Ablation study results (ADE/FDE and ANLL/FNLL). Lower is better.

Method	BiTraP-NP				BiTraP-GMM			
	w/o backward (TraP-NP)		w/ backward		w/o bi-loss		w/ bi-loss	
	ADE/FDE	ANLL/FNLL	ADE/FDE	ANLL/FNLL	ADE/FDE	ANLL/FNLL	ADE/FDE	ANLL/FNLL
ETH	0.44/0.96	4.20/4.45	**0.37/0.69**	**3.80/3.79**	0.43/0.80	1.11/3.81	**0.40/0.74**	**0.96/3.55**
Hotel	0.13/0.23	-0.17/1.64	**0.12/0.21**	**-0.41/1.26**	0.16/0.25	-1.32/0.80	**0.13/0.22**	**-1.60/0.51**
Univ	0.21/0.43	-0.21/2.78	**0.17/0.37**	**-0.84/2.15**	0.20/0.41	-1.16/2.06	**0.19/0.40**	**-1.19/2.03**
Zara1	0.15/0.31	-0.37/2.27	**0.13/0.29**	**-0.81/1.85**	0.19/0.35	-0.90/2.12	**0.14/0.28**	**-1.51/1.56**
Zara2	0.12/0.23	-1.70/1.54	**0.10/0.21**	**-1.89/1.31**	0.13/0.25	-2.38/0.64	**0.11/0.22**	**-2.54/0.38**

Table 4.7: Computational times with 20/2000 samples.

Method	Scene Graph	Model inference	Total
S-GAN[51]	N/A	103/10445 ms	103/10300 ms
Trajectron++[48]	11ms	55/58 ms	66/69 ms
TraP-NP	N/A	5.3/5.9 ms	5.3/5.9 ms
BiTraP-NP	N/A	8.3/9.1 ms	8.3/9.1 ms
BiTraP-GMM	N/A	69/72ms	69/72ms

4.3.9 Experiments on NuScenes Dataset

To further present the performance of BiTraP in bird's eye view autonomous driving scenarios, we evaluate on the nuScenes dataset [8]. The nuScenes dataset contains trajectories collected from 850 scenes, 700 for training and 150 for testing [8]. We followed [48] to extract training and testing trajectories and trained our model using the same configurations as in ETH-UCY experiment. Note that we treat the pedestrian/vehicle position at 4 seconds in the future as the target of our goal or end-point during training.

Evaluation metrics. To be comparable with [48], the most-likely (ML) prediction is used to compute the final displacement error (FDE). We also use the kernel density estimation negative log-likelihood (KDE NLL) as in our other experiments.

Table 4.8: Pedestrian-only trajectory prediction results on nuScenes dataset.

Method	KDE NLL				FDE ML			
	@1s	@2s	@3s	@4s	@1s	@2s	@3s	@4s
Trajectron++ base [48]	-2.69	-2.46	-1.76	-1.09	0.03	0.17	0.37	0.60
Trajectron++ $\int$, map [48]	-5.58	-3.96	-2.77	-1.89	**0.01**	0.17	0.37	0.62
BiTraP-GMM (ours)	**-6.08**	**-4.21**	**-2.98**	**-2.05**	0.02	**0.15**	**0.35**	**0.58**

Table 4.9: Vehicle-only trajectory prediction results on nuScenes dataset.

Method	FDE ML			
	@1s	@2s	@3s	@4s
Trajectron++ base [48]	0.18	0.57	2.25	2.24
Trajectron++ $\int$, map [48]	**0.07**	0.45	1.14	2.20
BiTraP-GMM (ours)	0.08	**0.43**	**1.06**	**1.99**

Results. As can be seen in Table 4.8, adding dynamic integration and map encoding to the base Trajectron++ improved the distribution accuracy by a large margin but does not affect the FDE ML, indicating similar modes but smaller variances of the predicted distributions. Trajectron++ based methods used interactions and/or encoded map as inputs while our BiTraP-GMM only takes target pedestrians past trajectory. As in Table 4.8, BiTraP-GMM improves the KDE-NLL at all evaluated time steps and also improves FDE after 2 seconds, showing how does the bi-directional strategy improves prediction accuracy. Note that the Trajectron++ benchmark lacks a ablation with integration but not map encoding (e.g. Trajectron++ $\int$) to show

the necessity of map. However, our experiment shows that map may not be a very important information when predicting pedestrian trajectories on nuScenes dataset since BiTraP-GMM outperforms "Trajectron++ $\int$, map".

4.3.10 Robot Navigation Simulation Experiment Using *BiTraP*

To quantitatively analyze application of the BiTraP-GMM and BiTraP-NP models to robot navigation tasks, we designed a simulated robot navigation experiment based on the ETH-UCY bird's-eye view dataset. In this experiment, given predicted pedestrian trajectory distributions in a scene using our BiTraP models and pre-planned paths for a robot, we show that we are able to compute the collision likelihood for each path, and thus are able to predict collision rate and select the safest path for the robot. Assuming a mobile robot navigates among pedestrians, we present results on two tasks: 1) Select the safest path for the robot and 2) Predict whether a path will collide with any other pedestrians in the scene. In this section, we first introduce our experiment setup. Then, we present evaluation results of our BiTraP models on path selection and collision prediction tasks.

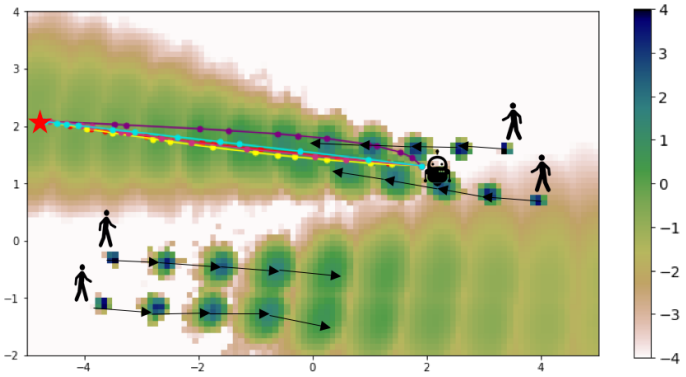


Figure 4.8: Generation of Monte Carlo (MC) robot trajectories for collision detection experiments using Bezier curves. We illustrate five MC trajectory samples including start (robot icon) and end (red star) waypoints. Predicted trajectory distributions of neighbor pedestrians are plotted as a heat map; their walking directions are indicated by black arrows.

Experimental Setup. We selected all samples with more than one pedestrian in the test split [51] from ETH-UCY. Each sample has a node pedestrian (the pedestrian used for testing in previous work) and several neighbor pedestrians (the pedestrians used for social modeling in previous work) as in [51, 48]. We regard the node pedestrian as a "robot" navigating among other neighbor pedestrians. The starting and goal points of the "robot" are the same as the current position and goal point

of the node pedestrian. A sample scene with one "robot" navigating among four other pedestrians is shown in Fig. 4.8. For the robot, 100 Monte Carlo (MC) paths were generated from start state to end point following quadratic and cubic Bezier curves [128]. Other more complex path planners could be used to generate additional experimental datasets. We assume the robot must reach the designated goal in 12 time steps, matching the prediction horizon for the pedestrian node in each scene. We uniformly generate waypoints along the path and randomly shift each by up to $\pm 50\%$ of the step length, resulting in a trajectory sequence containing 12 random waypoints. Other pedestrians follow their original (ground truth) trajectories in the scene. For each neighbor pedestrian, we run BiTraP-NP and BiTraP-GMM separately. Each method samples 2000 future trajectories to fit one Gaussian Kernel Density Estimation (KDE) model for each pedestrian as the predicted future distribution. Then, we compute the maximum KDE log-likelihood of all the waypoints on all robot MC paths and treat this log-likelihood value as a collision score. The higher the collision score, the more likely a collision will happen along this path. Given these collision scores, we compute the safest path collision rate (SPCR) as reported in *Task 1* below. Receiver operating characteristic (ROC) and precision-recall (P-R) curve results are reported in *Task 2*.

Task 1: Predict the Safest Path. We mark the robot MC path in each scene with minimum collision score as the "safest" (lowest collision likelihood) path. Then, we compute Euclidean distances between each safest path waypoint and other pedestrians' ground truth future trajectories. A collision is tallied if the minimum distance between a path and any pedestrians in the scene is less than 0.2 meters. Collision rate is computed as the number of paths with collision divided by the total number of safest paths. Due to the randomness in MC path generation, we conducted the simulation experiment five times with BiTraP-NP and BiTraP-GMM predictors separately and report collision rate mean (μ) and standard deviation (σ) values in Table 4.10. As a comparison, we also present the collision rate of a randomly selected path among the 100 MC paths. The randomly selected paths do not have very high collision rates since the paths are planned based on pedestrian ground truth start and goal positions which are less likely to be involved in a collision. Compare to randomly selected paths, paths selected by our methods reduce the SPCR by a large margin. This shows that our predictors are effective for safest path selection. Both of our BiTraP methods achieve collision rate lower than 1% on *ETH*, *Hotel* and *Zara1* datasets. The *Univ* dataset is more difficult due to its high pedestrian density, and *Zara2* is most difficult because many pedestrian trajectories are quite close to

each other. BiTraP-GMM shows lower SPCRs than BiTraP-NP on four datasets, indicating that it predicted more accurate (compared to ground truth) distributions. On *Zara1*, BiTraP-NP outperforms BiTraP-GMM by a small margin. BiTraP-NP ANLL and FNLL metric values as reported in the main paper are still higher than BiTraP-GMM values. A possible explanation is that BiTraP-NP predicts more diverse distributions thus detects some collisions not identified by BiTraP-GMM.

Task 2: Predict Collision for Any Path. The collision rate metric above only evaluates the safest path as selected by a trajectory predictor thus neglects all other paths. In the real-world, a trajectory predictor must be sufficiently accurate for the robot to accurately predict future collisions with high precision with a low missing rate (high true positive rate, TPR) and a low false alarm rate (low false positive rate, FPR). To show the performance of BiTraP-NP and BiTraP-GMM predictors in terms of these metrics, we plotted the collision prediction ROC curve and P-R curve as follows. First, we collected all MC paths for the robot and tallied their collision scores. By setting a threshold γ, we can classify a path as collided (positive) or not collided (negative) and compute the TPR (i.e., recall), FPR and precision values. The ground truth label of each path is computed in the same way as before. By decreasing γ from a maximum value to minimum value (6 and -10 in this work), we plot the ROC and P-R curves shown in Fig. 4.9. The corresponding area under curve (AUC) and average precision (AP) are presented in Table 4.10. In this work, AP is computed by equally spaced recall levels {1/40, 2/40,...,1} following [129].

Table 4.10: SPCR($\mu \pm \sigma$), AUC and AP results of our methods on ETH-UCY data group.

	Random from 100 (SPCR)	BiTraP-NP (SPCR/AUC/AP)	BiTraP-GMM (SPCR/AUC/AP)
ETH	$0.6 \pm 0.4\%$	$0.3 \pm 0.1\%$/ 92.3/ 24.2	$\mathbf{0.1 \pm 0.1\%}$/ **95.5**/ **26.0**
HOTEL	$0.4 \pm 0.3\%$	$0.1 \pm 0.1\%$/ 86.4/ 22.4	$\mathbf{0.0 \pm 0.0\%}$/ **91.6**/ **29.1**
Univ	$8.5 \pm 1.4\%$	$5.8 \pm 0.5\%$/ 81.0/ 33.4	$\mathbf{3.6 \pm 0.2\%}$/ **87.6**/ **43.4**
Zara1	$2.4 \pm 0.5\%$	$\mathbf{0.6 \pm 0.2\%}$/ 88.9/ 38.6	$0.8 \pm 0.3\%$/ **90.4**/ **41.6**
Zara2	$6.1 \pm 0.6\%$	$3.2 \pm 0.1\%$/ 81.0/ 44.0	$\mathbf{2.5 \pm 0.3\%}$/ **87.5**/ **52.6**

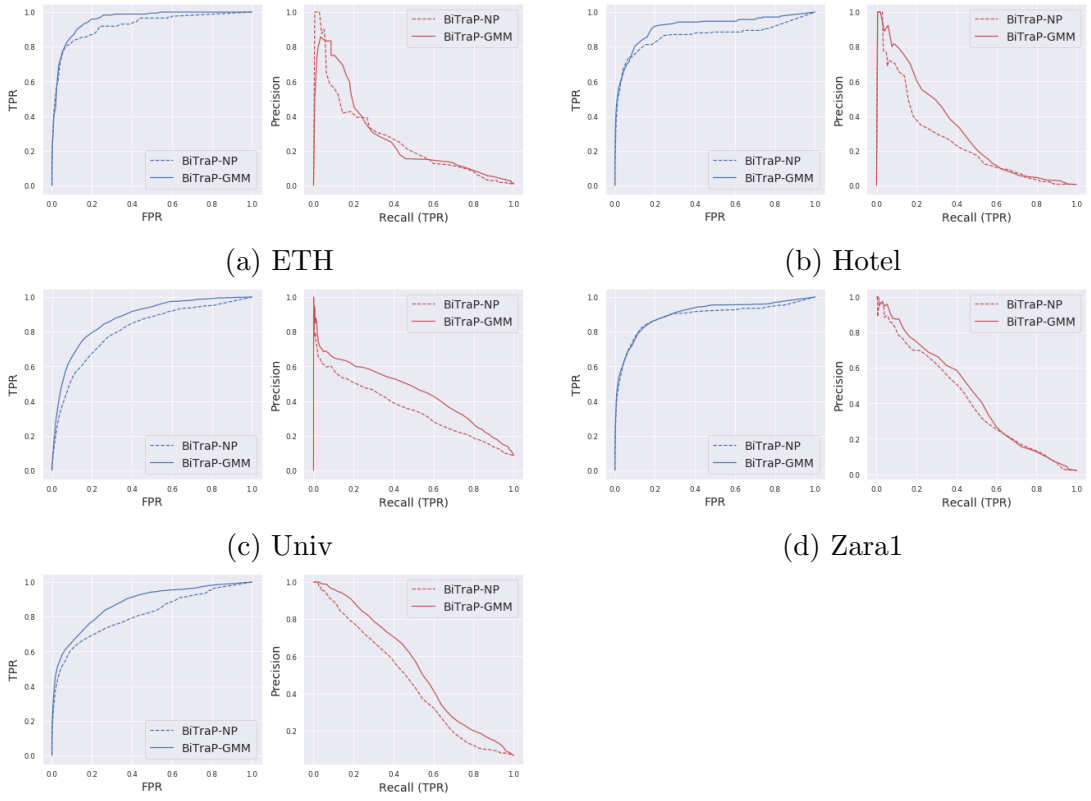
As shown in Fig. 4.9 and Table 4.10, both BiTraP-NP and BiTraP-GMM methods achieve high AUCs (e.g., > 90 on *ETH*). Generally, BiTraP-GMM outperforms BiTraP-NP by a small margin in terms of both AUC and AP (e.g., 95.5 vs 92.3 AUC, and 26.0 vs 24.2 AP on *ETH*). Note that in real-world mobile robot applications missed collision detection (false negative) is unacceptable due to safety. That

is to say, a high TPR (recall) is required. As can be observed in the higher TPR regions (x-axis) of the P-R curves, BiTraP-GMM outperforms BiTraP-NP on *ETH* (Fig. 4.9(a)) and *Hotel* (Fig. 4.9(b)), and both methods perform similarly on *Zara1* (Fig. 4.9(d)). On *Univ* (Fig. 4.9(c)) and *Zara2* (Fig. 4.9(e)), when the TPR is greater than a relatively high value (say 0.8), the FPR are higher (> 0.2) than in the other datasets, indicating increased chance of false alarms on these two datasets.

Compared to the ROC curve, the P-R curve is more suitable for imbalanced datasets due to the fact that it evaluates the fraction of true positives among positive predictions. This fits our case where the ratio of with-collision to no-collision paths is around 1:140, a large imbalance. On *Univ* and *Zara2* (Fig. 4.9(c) and 4.9(e)), BiTraP-GMM has higher precision than BiTraP-NP across almost all recall values. On the other hand, on *ETH*, *Hotel* and *Zara1* (Fig. 4.9(a) 4.9(b) and 4.9(d)), the two methods achieve similar precision at higher recall regions (e.g., when recall > 0.6). This is because when the threshold γ is too low, many paths are predicted as collided by both methods.

The ROC and P-R curves also verified our observation regarding the diversity of the predicted trajectory distribution as described in the main paper. At a fixed TPR on the ROC curves, we observe that BiTraP-NP always has a greater FPR than BiTraP-GMM, consistent with our hypothesis that BiTraP-NP predicts more diverse distributions, thus predicts more false alarms. Similarly, with fixed recall in P-R curves, BiTraP-NP has lower precision due the greater number of false alarms.

In summary, this simulated robot collision experiment demonstrated our proposed BiTraP trajectory predictor can be used in future robotic applications, such as predicting collisions and selecting safest paths in robot navigation tasks. Results from this supplementary experiment are consistent with our main paper's observations and further verify our hypothesis regarding the diversity/compactness of predicted trajectory distributions, i.e., BiTraP-NP predicts more diverse distributions while BiTraP-GMM predicts more compact distributions. The SPCR, ROC (AUC) and P-R (AP) metrics used in this experiment act as a supplement to the currently reported and widely used ADE/FDE and KDE-NLL metrics in the main paper. We believe these additional metrics and experiments offer an intuitive and complementary performance evaluation of the two proposed BiTraP models (NP and GMM) and their applications for tasks such as collision prediction and path selection.



(a) ETH

(b) Hotel

(c) Univ

(d) Zara1

(e) Zara2

Figure 4.9: ROC (left) and P-R (right) curves of BiTraP-NP and BiTraP-GMM on ETH dataset.

4.4 Pedestrian Intent and Action Detection

4.4.1 Crossing Intent Detection using Action Detection and Prediction

Pedestrian crossing intent describes a person's underlying plan to cross a street. Previous work has defined this problem as a binary classification (i.e. "will cross" vs "will not cross") of video clips. Given a fixed-length sequence of observations $\{I_1, I_2, ..., I_t\}$ for a pedestrian in a video, a classification model predicts the probability of this person's crossing intent at time t, notated as i_t. Instead of video clip classification, we propose classifying each observed frame based on itself and the past frames encoded by a RNN network, called online action/intent detection [1, 73]. Online detection has two advantages compare to clip classification: 1) The hidden state contains information from a longer history than a constant video clip; 2) Online detection saves computation time by only processing one frame instead of one clip at each time. A model trained with only binary class intent supervision might miss information that describes pedestrian action and context. For example, a person walking

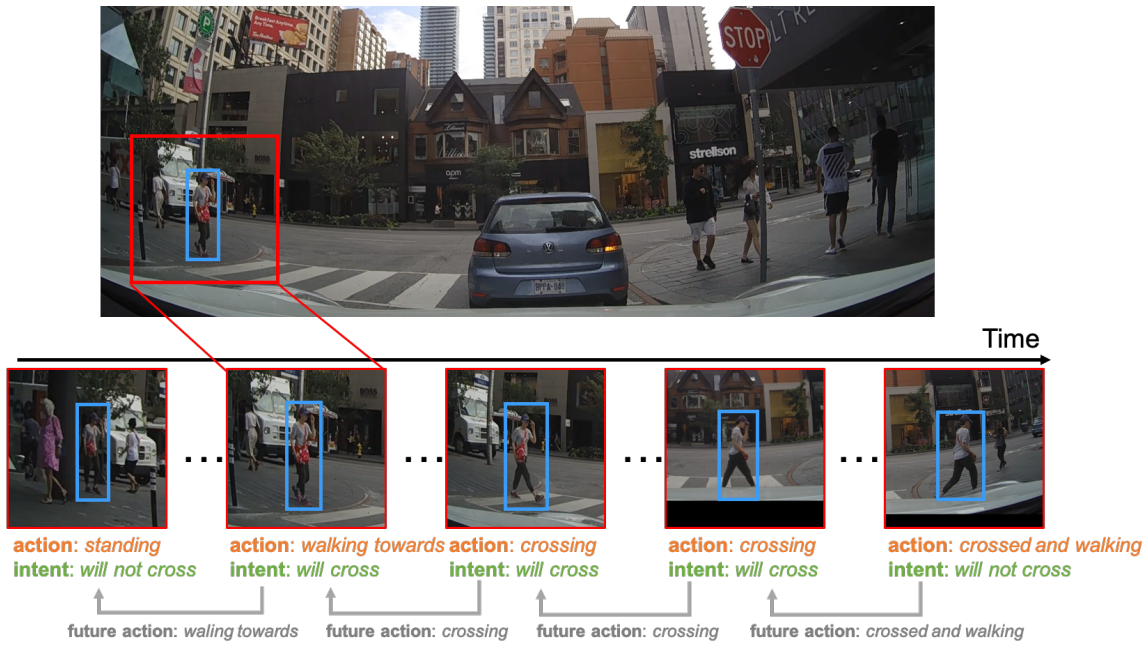


Figure 4.10: Intent detection based on action detection and prediction.

towards a crosswalk and a person waiting in front of a crosswalk can both be classified as "will cross". However, the difference between their actions and contexts, essential to the "will cross" intent, are not captured by such a model. Therefore, we propose to train a multi-task neural network which detects both the crossing intent $[i_1, i_2, ..., i_t]$ and semantic actions $[a_1, a_2, ..., a_t]$. In addition to detecting the present action, we believe that predicting future action is essential to crossing intent understanding, since intent describes the underlying action that also can impact the future. For example, in Fig. 4.10, a predicted "crossing" or "walking towards crosswalk" action indicates a high confidence in "will cross" intent. Thus, we introduce an action predictor module which takes the observation feature to predict the actions in δ frames, notated as $[a_{t+1}, a_{t+2}, ..., a_{t+\delta}]$. In this section, we describe a novel multi-task intent detection network with action detection and prediction.

Intent-action encoder. First, we extracted visual features from an image patch around the target pedestrian using a pre-trained deep convolutional neural network (CNN) as in [12]. As shown in Fig. 4.11, the image patch is a square region defined by an enlarged pedestrian bounding box that includes not only the person but also context information such as ground, curb, crosswalk, etc. The bounding box coordinates $[top, left, bottom, right]$ are passed through a fully-connected (FC) network to obtain location-scale embedding. The intent-action encoder cell (ENCcell) takes the concatenated feature vector as input and recurrently encodes it (over time). To en-

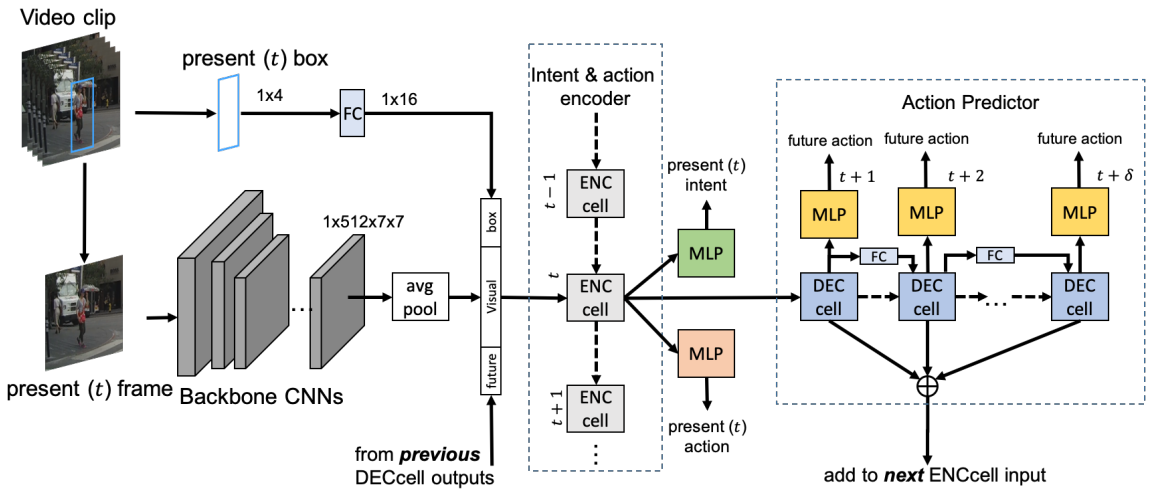


Figure 4.11: Our multi-task intent action detection prediction network.

sure the neural network learns features that represent both crossing intent and more complicated semantic actions, we designed an intent classifier and an action classifier that use the encoder hidden state to predict the present intent and action at the same time. We use a multi-layer perceptron (MLP) network for each classifier.

Action predictor. To further take advantage of future action prediction, we designed a module to predict future action and pass information to the next ENCcell, similar to the temporal recurrent neural network (TRN) structure [73]. The hidden state at each iteration is classified by another MLP network to predict the future action at each forecasting time step. All decoder hidden vectors are collected and the average is computed as the "future input". Such a "future input" is concatenated with the visual and box features to formulate the input to the next ENCcell. During training, at each time step of the encoder, there are δ actions predicted in the future δ frames, resulting in $T \times \delta$ future actions for a training sample with length T.

Multi-task loss. We use binary cross entropy (BCE) loss for the binary intent detection and cross entropy (CE) loss for multi-class action detection and prediction. The total loss is written as

$$Loss = \frac{1}{T}\sum_{t=1}^{T}\Big(\omega_1 BCE(\hat{i}_t, i_t) + \omega_2 CE(\hat{a}_t, a_t) + \omega_3 CE([\hat{a}]_{t+1}^{t+\delta}, [a]_{t+1}^{t+\delta})\Big), \tag{4.14}$$

where T is the training sample length; variables with and without $\hat{\cdot}$ indicate the predicted and ground truth values, respectively; and $[\cdot]_{t+1}^{t+\delta}$ indicates predicted sequences from time $t+1$ to time $t+\delta$, respectively. The three loss terms are intent detection

loss, action detection loss and future action prediction loss with weighting parameter ω_1, ω_2 and ω_3. In this work, we design ω_1 to start with value 0 and increase towards 1 over the training process based on a sigmoid function of training iterations. This procedure ensures the model learns a reasonable action detection and prediction module which can be used as a beneficial prior for the intent detector. For simplicity, ω_2 and ω_3 are both 1 through the training process.

4.4.2 Discussion on Intent Detection Evaluation

Pedestrian intent is defined as a binary classification problem and is commonly evaluated using accuracy and F1 score [12]. To be specific, a prediction is considered positive if the score if greater than 0.5, otherwise negative. The accuracy and F1 score are computed as in (4.15) and (4.16). However, due to the potential imbalance of test datasets, both accuracy and F1 score can be biased towards the majority class. For example, simply classifying all samples to positive on a test set with positive to negative ratio of $4:1$ will result in a high 0.8 accuracy and a 0.89 F1 score.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN}, \tag{4.15}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{TP}{TP + 0.5 \cdot (FP + FN)}, \tag{4.16}$$

$$TPR(recall) = \frac{TP}{TP + FN}, \quad FPR = \frac{FP}{FP + TN}, \tag{4.17}$$

$$\Delta_s = \frac{1}{|P|} \sum_{i \in P} s_i - \frac{1}{|N|} \sum_{i \in N} s_i, \tag{4.18}$$

To address this problem when evaluating imbalanced datasets, we introduce two additional evaluation metrics for crossing intent detection to supplement accuracy and F1 score. Inspired by anomaly detection evaluation metrics, we plot the receiver operating characteristic (ROC) curve and compute the area under curve (AUC) as additional evaluation metrics for crossing intention detection. ROC and AUC are better metrics for imbalanced test sets since they reflect both true positive rate (TPR) and false positive rate (FPR) in shown in (4.17). We also compute the difference between the average scores of all positive samples and all negative samples as in Eq. (4.18), notated as Δ_s. The greater Δ_s, the better a model can distinguish between crossing and not crossing intent.

We present evaluation results on ROC AUC, Δ_s, accuracy and F1 score as follows in Section 4.4.3.

4.4.3 Experiments on PIE datasets

We conducted experiments on the Pedestrian Intent Estimation (PIE) dataset [12]. PIE has 1,842 pedestrians with crossing intent annotated, which splits to 880, 243 and 719 for training, validation and testing. The original pedestrian action annotation contains only two categories, "standing" and "walking". To make the annotations more informative for intent detection, we augmented the existing two categories to seven categories by adding crossing action information (whether the crossing action will happen or has already happened). The amended annotation labels are presented in Table 4.11.

Table 4.11: Semantic actions and crossing intent categories in the PIE dataset.

Action Class	**Explanation**
Standing	The person is standing and will not cross a street.
Waiting	The person is standing and waiting to cross a street.
Going towards	The person is going towards a crossing point.
Crossing	The person is crossing a street.
Crossed and standing	The person finished crossing and is standing.
Crossed and walking	The person finished crossing and is walking.
Other walking	Any walking that does not belong to above actions.
Intent Class	**Explanation**
Will cross	The person is going to cross a street
Will not cross	The person is going to cross a street

Implementation Details. The proposed neural network model was implemented using PyTorch [130]. We used gated recurrent units (GRU) with hidden size 128 for both ENCcell and DECcell, and a single-layer fully connected (FC) network as the classifier for intent detection, action detection and action prediction. To generate the input image patch, we first doubled the pedestrian bounding box width and height and then squared the new box based on the longer edge. The image patch is cropped from the resulting box. For boxes that are out of the frame boundary, we padded with zeros to make square image patches. All input image patches were resized to 224×224. We used an ImageNet-pretrained VGG16 network as the backbone CNN to extract features from image patches. During training, we segmented the pedestrian trajectories to training samples with constant segment length T. Our model detects intent and action for each time step $t \in [1, 2, ..., T]$ and predicts δ future actions at each t. Using constant T made batch training feasible for our model. Due to the class imbalance in PIE [12], we applied a weighted sampler for training data sampling. The

test data is not balanced to ensure fair comparison with other methods. All models were trained with sample length $T = 30$, learning rate $1e-5$, and batch size 128 on an NVIDIA Tesla V100 GPU. During inference, we ran our model on one test sequence at a time to collect the intent and action detection results for each time step.

Baselines and Ablations. We present results of two naive baseline methods to show the drawback of existing accuracy and F1 score metrics. The *Naive-random* method randomly assigns a crossing confidence score between 0 ("not crossing") and 1 ("crossing") to each test sample, while the *Naive-1* method assigns score 1 ("crossing") to all test samples. We also compare to the state-of-the-art (SOTA) method PIE_{intent} as presented in [12]. PIE_{intent} uses a Convolutional LSTM network to encode past visual features and then concatenates encoded features and bounding boxes to predict pedestrian intent at the final frame.

We also evaluate three ablation models of our method: *intent only* is a basic model without action detection and prediction modules. It is similar to the PIE_{intent} model except that we used an average pooling feature extraction and a simpler GRU encoder to replace the ConvLSTM and LSTM networks in PIE_{intent}. The *intent+action* model is a multi-task model with both intent and present action detection modules, and *intent+action+future* is our full model with intent detection, present action detection and future action prediction networks.

Quantitative Results. Table 4.12 shows the intent detection results of baseline methods and our ablation models. As can be seen in the top three rows, *Naive-1* achieves 0.82 accuracy and 0.90 F1 score, higher than the PIE_{intent} method due to the fact that the test set of PIE dataset has approximately a $4:1$ ratio between "crossing" and "not crossing" classes. This observation verifies our previous discussion on the inadequacy of using only accuracy and F1 score as evaluation metrics. The AUC and Δ_s metrics, however, are less sensitive to the data imbalance and show different performance comparison between a naive and a SOTA methods. The Δ_s of naive methods are 0.00 because they do not distinguish crossing and not crossing.

The *intent only* method is our baseline ablation where action detection and prediction are removed. It achieves similar results with PIE_{intent} in terms of AUC and Δ_s, indicating that an average pooling and simple GRU network can perform similarly to a more complicated ConvLSTM encoder in PIE_{intent}. Its higher accuracy and F1 scores may be a result of making more true positive predictions, but its lower AUC indicates that it also yields more false positives. Both PIE_{intent} and *intent-only* have small Δ_s (0.06 and 0.07), indicating that the models have low prediction

Table 4.12: Intent and action detection results on the PIE dataset. The bold numbers are the highest results of each metric. Note that only *intent+action* and *intent+action+future* methods have the action prediction module.

Method	intent				action
	Accuracy↑	F1 score↑	AUC↑	Δ_s ↑	mAP↑
Naive - random	0.50	0.61	0.50	0.00	N/A
Naive - 1	**0.82**	**0.90**	0.50	0.00	N/A
PIE_{intent} [12]	0.79	0.87	0.73	0.06	N/A
intent only	**0.82**	**0.90**	0.72	0.07	N/A
action only	N/A	N/A	N/A	N/A	0.18
intent + action	0.74	0.83	0.76	0.11	0.21
action + future	N/A	N/A	N/A	N/A	0.20
intent + action + future	0.79	0.87	**0.78**	**0.16**	**0.23**

confidences (*i.e.*, , near 0.5) on many test samples. The *intent+action* model significantly increased AUC and Δ_s, indicating an improved capability in distinguishing between crossing and not crossing intent. However, its lower accuracy and F1 scores show that it misses true positives, i.e., some crossing intents were not correctly detected. Together, these metrics show the *intent+action* model has less false alarms with crossing intent, and it distinguishes most crossing/not crossing intents better, but it sacrifices sensitivity to some crossing intent. One explanation is that more negative (not crossing) samples previously misclassified by the *intent only* model are now correctly classified when the action type is known in the *intent+action* model. For example, a person stands close to the curb but does not face the road is just "standing" there instead of "waiting". The *intent+action* model recognizes such differences and will correctly classify this example as "not crossing", while the *intent only* model may lose that information. However, multi-class action detection is more difficult than intent detection and can fail in many cases, resulting in failure of detecting crossing intent. The full *intent+action+future* model further improves the AUC and Δ_s. It also improves the accuracy and F1 scores from the *intent+action* ablation. Future action is a more straightforward indicator to intent, e.g., a high confidence on "waiting" or "crossing" action indicates high possibility that a pedestrian has the intent to cross the road. The higher Δ_s results show that adding future action stream helps the model to better distinguish different crossing intents.

Qualitative results. Fig. 4.12 shows results of four examples in the PIE dataset. The top two rows are two examples of pedestrian walking along the road without crossing intent. Our *intent only* (int-only) method performs similarly to the PIE_{intent}

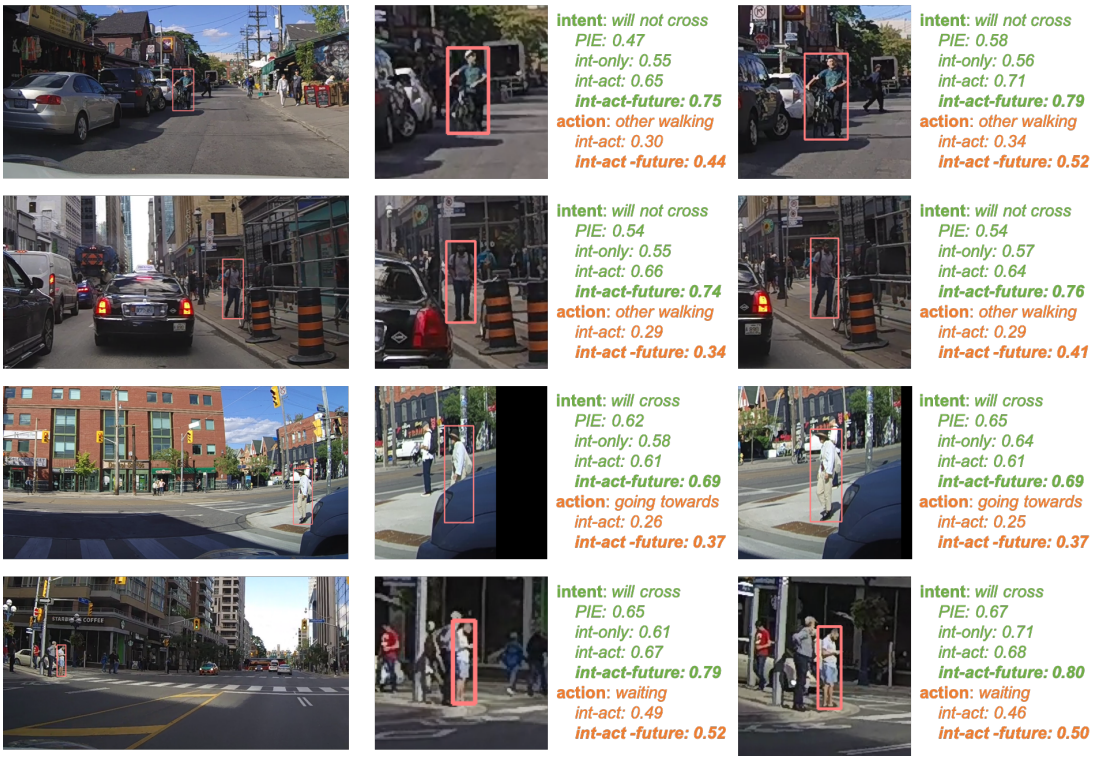


Figure 4.12: Results on four examples in the PIE dataset. The far left column shows an overview of the scene while the right two image patches are cropped from an earlier frame and a later frame in the corresponding pedestrian trajectory sequence. Results are shown to the right of each image patch. The green text shows the ground truth intent class and the confidence of this class predicted by the state-of-the-art PIE method and our three ablation models. The orange text shows the ground truth action class and the confidence of this class predicted by our *intent+action* and *intent+action+future* models.

(PIE), while the *intent+action* (int-act) and *intent+action+future* (int-act-future) methods significantly outperform the other two baselines. The third row shows a pedestrian walking towards the crosswalk to cross the street. When the person gets closer to the crossing point, the PIE_{intent} and *intent only* method predict higher scores for the crossing intent, while the two models with action modules predict higher scores consistently, indicating that action supervision helps intent detection. Similarly, in the fourth row, our methods recognize that the person is waiting in front of a crosswalk and will cross in the future. Generally, we observe better performance on later frames than earlier frames due to the fact that with the ego car and the target pedestrian moving closer to each other, the pedestrian visibility and image quality improves (as can be seen in all examples), which helps intent detection. In terms of action detection performance, adding a future module also significantly improves the model, indicating

that predicting future action can help the present action detection, especially when the prediction module captures future action changes (e.g., in row 3, from "walking towards" to "crossing"). The future prediction module can add extra information to the encoder module which helps detect present action ("walking towards"). When the future action does not change (as in rows 1, 2 and 4), the future prediction module still helps detect the present action as it uses extra memory to extract deeper features from the sequence and merges them with shallow features in the encoder [73].

Fig. 4.13 shows details of a challenging example where the target pedestrian goes from the sidewalk to the road to yield on-coming pedestrians, and then walks back to the sidewalk, resulting in a "fake crossing". In the leftmost image, the person moves towards the street, making all methods predict lower scores for the "will not cross" intent. After the yielding action, the target pedestrian starts to go back to the sidewalk so that our *intent+action+future* model has higher confidence that the person's action is "other walking" and the intent is "will not cross". Fig. 4.14 shows two more challenging cases. In the first row, the person is facing towards the street but is waiting at a bus station instead of waiting to cross, which confuses the models and results in a relatively low confidence on the "will not cross" intent. In the second row, the person is facing the street at first but then turns the orientation, making all methods predict lower scores for the "will cross" intent. These challenging cases show the difficulty of pedestrian intent and action detection in real-worlds scenarios, as there are too many variables and a small change could make a big difference in neural network performance.

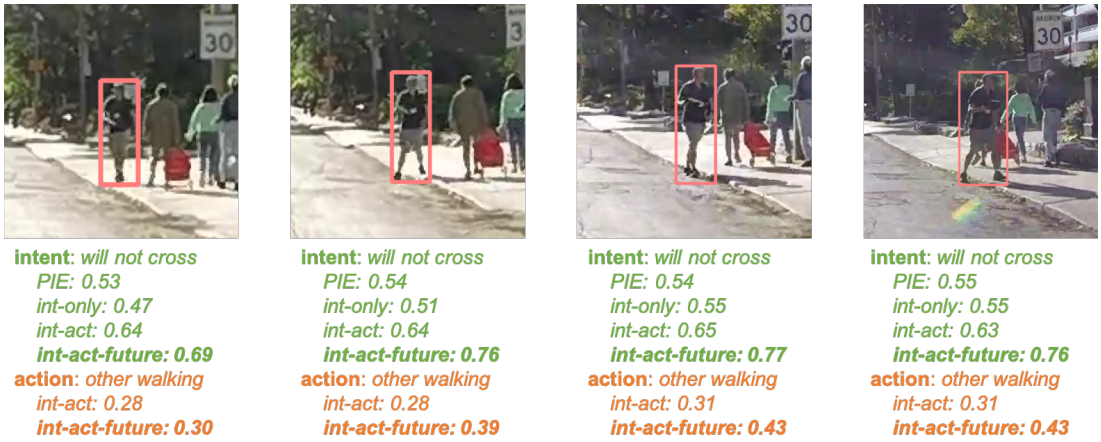


Figure 4.13: An example showing how our method captures a crossing-like case and distinguishes it from a real crossing intent.

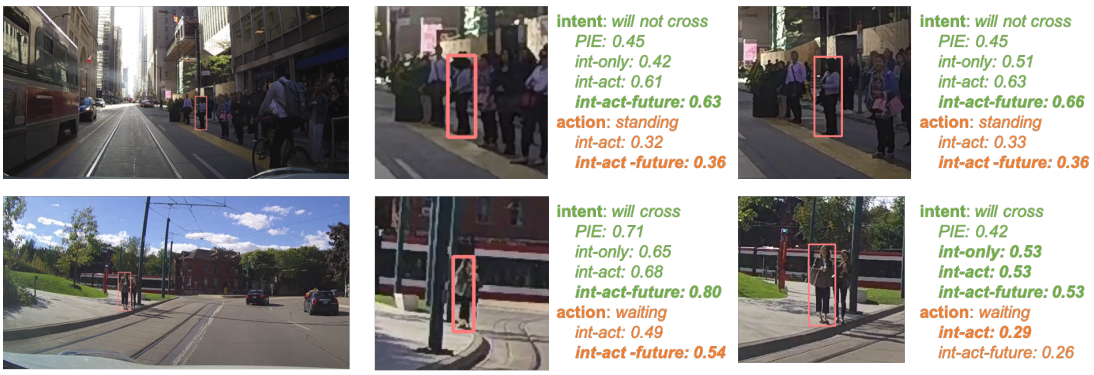


Figure 4.14: Two examples of challenging cases.

4.5 Conclusion

This chapter first presented *FOL*, a multi-stream RNN encoder decoder network for bounding box trajectory prediction in ego-centric videos. We showed that FOL improves the prediction accuracy by encoding optical flow features as additional input and adding camera ego motion prediction streams as a parallel or side task. We then introduced *BiTraP*, a bi-directional multi-modal trajectory prediction method conditioned on goal estimation. We demonstrated that BiTraP can achieve state-of-the-art results for trajectory prediction on both first-person view and bird's eye view datasets. The current BiTraP models, with only observed trajectories as inputs, already surpass previous methods which required additional ego-motion, semantic intention, and/or social information. By conducting a comparative study between non-parametric (BiTraP-NP) and parametric (BiTraP-GMM) models, we observed that the different latent variable choice affects the diversity of future trajectory target distributions. We hypothesized that such differences in predicted distributions directly influence collision rate in robot path planning and showed that collision metrics can be used to guide predictor selection in real world applications.

We also proposed a multi-task learning method for pedestrian crossing intent detection. Pedestrian semantic action along a trajectory has been carefully segmented and predicted to help estimate crossing intent. Experiments on the PIE dataset show the effectiveness of our method for pedestrian crossing intent detection. For future work, we plan to incorporate scene semantics and social components to further boost the performance of trajectory prediction and intent detection. We are also interested in using predicted goals and trajectories to infer and interpret pedestrian intentions and actions.

CHAPTER V

Anomaly Detection in Ego-centric Traffic Videos

5.1 Introduction

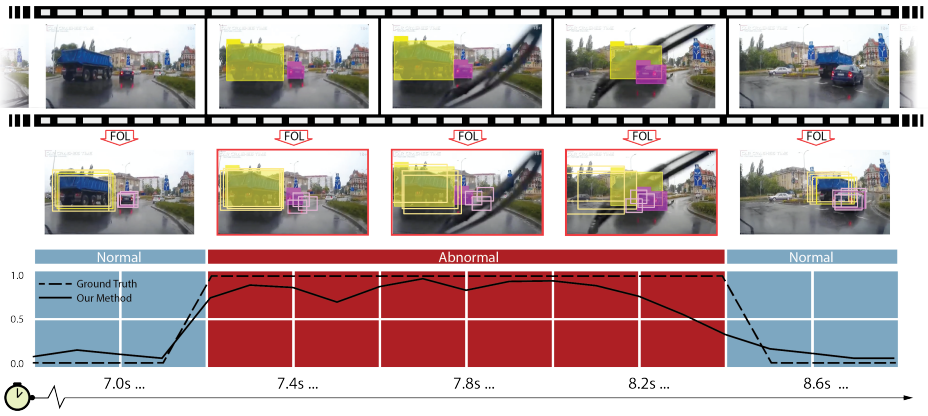


Figure 5.1: Overview of our method based on future object localization (FOL) using sampled video from our DoTA dataset.Annotated bounding boxes (filled) and predicted boxes are presented. For each time step, we collect FOL predictions of all traffic participants from different past time steps and compute the bounding box standard deviation, called consistency, as the anomaly score.

Driving has the potential to transform the world as we know it, revolutionizing transportation by making it faster, safer, cheaper, and less labor intensive. A key technical barrier is building autonomous systems that can accurately perceive and safely react to the huge diversity of situations that are encountered on real-world roadways. The problem is that driving situations obey a long-tailed distribution, such that a very small number of common situations makes up the vast majority of what a driver encounters, and a virtually infinite number of rare scenarios — animals running into the roadway, cars driving on the wrong side of the street, etc. — make up the rest. While each of these individual scenarios is rare, they can and do happen.

In fact, the chances that *one* of them will occur on any given day are actually quite high.

Existing work in computer vision has applied deep learning-based image classification to detect anomalies in video collected by dashboard-mounted cameras [19, 21, 99]. However, the long-tailed distribution of driving events means that unusual events may occur so infrequently that it may be impossible to collect training data for them, or to even anticipate that they might occur [2]. In fact, some studies indicate that driverless cars would need to be tested for *billions* of miles before enough of these rare situations occur to accurately measure system safety [23], much less collect sufficient training data to make them work well.

An alternative approach is unsupervised video anomaly detection (VAD), which avoids modeling all possible driving scenarios by training models that recognize "normal," safe roadway conditions, and then signaling an anomaly when events that do not fit the model are observed. Unlike the fully-supervised classification-based work, these unsupervised approaches may not be able to identify exactly which anomaly has occurred, but nevertheless may provide enough information for the driving system to recognize an unsafe situation and take evasive action. Unsupervised VAD has been widely applied to static surveillance camera datasets [97, 100, 104, 2, 46, 107] by training deep neural networks to reconstruct or predict video frames and computing the reconstruction or prediction errors as anomaly scores. However, these methods do not generalize well to driving videos since frame prediction and reconstruction is extremely difficult when cameras are rapidly moving, as in the driving scenario.

We side-steps this difficult problem by detecting objects and predicting their future locations, as opposed to trying to predict whole frames. We propose a novel approach that learns a future object (e.g., cars, bikes, pedestrians, etc.) localization (FOL) network in the field of view of a dashboard-mounted camera on a moving ego-vehicle. Our future object localization network consists of two modules: 1) An ego-motion RNN encoder-decoder to predict future odometry of the ego-vehicle, 2) A two-stream RNN encoder-decoder incorporating predicted ego-motion into future object bounding box predictions. This model can be easily learned from massive collections of dashboard-mounted video of normal driving, and no manual labeling is required. Existing unsupervised VAD [2, 97, 104, 101, 102, 103] computes prediction error with respect to ground truth as the anomaly score, which may cause problems in our case due to imperfect object detection and tracking. To address this issue, we propose two alternatives: 1) we take object boxes as foreground to generate binary foreground-background masks and compute the IoU between predicted and ground

truth masks as anomaly scores; 2) we collect FOL predictions for each time step from different past time steps and compute the prediction standard deviation, called consistency, as the anomaly score. We compare proposed methods with prediction accuracy methods and show the effectiveness of prediction consistency metric in traffic VAD experiments.

We also introduce a new large-scale benchmark dataset for traffic VAD called Detection of Traffic Anomaly (DoTA). DoTA contains $4,677$ videos with 18 anomaly categories [114] and multiple anomaly participants in different driving scenarios. DoTA provides rich annotation for each anomaly: type (category), temporal annotation, and anomalous object bounding box tracklets. Current anomaly datasets contain only temporal annotations, so they cannot be used to evaluate the accuracy of spatial localization — where in the frame an anomaly is occurring. However, accurately locating the anomalous region is essential for model explainability and downstream applications such as collision avoidance. Taking advantage of this large-scale dataset with rich anomalous object annotations, we propose a novel VAD evaluation metric called Spatio-temporal Area Under Curve (STAUC). STAUC is motivated by the popular frame-level Area Under Curve (AUC). While AUC uses a per-frame anomaly score which is usually averaged from a pixel-level or object-level score map, STAUC takes the score map and computes how much of it overlaps with the annotated anomalous region. This overlap ratio is used as a weighting factor for true positive predictions with STAUC such that AUC is the STAUC upper bound. We benchmark existing VAD baselines and state-of-the-art methods on DoTA using both AUC and STAUC, and show the advantage of using STAUC.

The DoTA dataset can also be used for video action recognition and online action detection given its categorical annotations. Video action recognition takes a video clip as input to predict its anomaly type, e.g. on-coming collision or out-of-control, while online action detection processes a video frame-by-frame to classify each frame as normal or one type of anomaly. We provide benchmarks of state-of-the-art methods such as SlowFast [69] and TRN [73] on on these two tasks. Experiments show that applying generalized video action recognition and online action detection methods to traffic anomaly understanding is far from perfect, motivating more research in this area.

This Chapter is organized as follows: Section 5.2 presented our future object localization-based unsupervised traffic video anomaly detection method and two anomaly score computation methods to address problems caused by imperfect object detection and tracking. Section 5.3 revealed the issue with the current evaluation metric and

introduced a new spatial-temporal area under curve metric as a supplement. Baseline VAD methods are introduced in Section 5.4, followed by experiment in Section 5.6 and 5.7, and a conclusion in Section 5.8

5.2 FOL based Traffic Anomaly Detection

Autonomous vehicles must monitor the roadway ahead for signs of unexpected activity that may require evasive action. A natural way to detect these anomalies is to look for unexpected or rare movements in the first-person perspective of a front-facing, dashboard-mounted camera on a moving ego-vehicle. Prior work [2] proposes monitoring for unexpected scenarios by using past video frames to predict the current video frame, and then looking for major differences. However, this does not work well for moving cameras on vehicles, where the perceived optical motion in the frame is induced by both moving objects and camera ego-motion. More importantly, anomaly detection systems do *not* need to accurately predict all information in the frame, since anomalies are unlikely to involve peripheral objects such as houses or billboards by the roadside. This paper thus assumes that an anomaly may exist if an object's real-world observed trajectory deviates from the predicted trajectory. For example, when a vehicle should move through an intersection but instead suddenly stops, a collision may have occurred.

Our model is trained with a large-scale dataset of normal, non-anomalous driving videos. This allows the model to learn normal patterns of object and ego motions, then recognize deviations without the need to explicitly train the model with examples of every possible anomaly. This video dataset is easy to obtain and does not require hand labeling. Considering the influence of ego-motion on perceived object location, we incorporate a future ego-motion prediction module [7] as an additional input. At test time, we use the model to predict the current locations of objects based on the last few frames of data and determine if an abnormal event has happened based on three different anomaly detection strategies per Section 5.2.4.

5.2.1 Bounding Box Prediction

Following Section 4.2, we denote an observed object's bounding box $X_t = [c_t^x, c_t^y, w_t, h_t]$ at time t, where $(c_t^x,\ c_t^y)$ is the location of the center of the box and w_t and h_t are its width and height in pixels, respectively. We denote the object's future bounding box trajectory for the δ frames after time t to be $\mathbf{Y_t} = \{Y_{t+1}, Y_{t+2}, \cdots, Y_{t+\delta}\}$, where each Y_t is a bounding box parameterized by center, width, and height. Given the

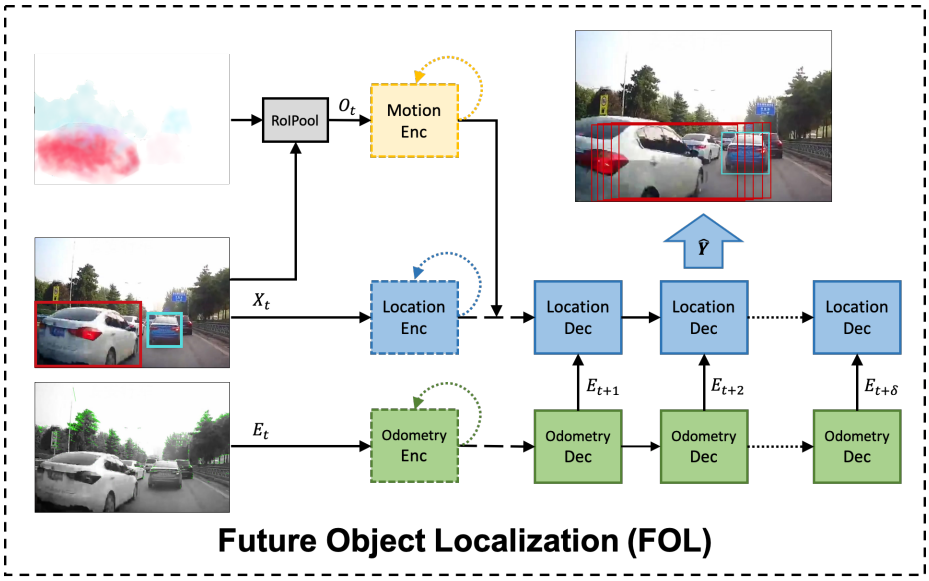


Figure 5.2: Overview of the future object localization model.

image evidence O_t observed at time t, a visible object's location X_t, and its corresponding historical information H_{t-1}, our future object localization model predicts $\mathbf{Y_t}$. This model is inspired by the multi-stream RNN encoder-decoder framework of Yao *et al.* [7], but with completely different network structure [73]. For each frame, [7] receives and re-processes the previous ten frames before making a decision, whereas our model only needs to process the current information, making it much faster at inference time. Our model is shown in Figure 5.2. Two encoders (Enc) based on gated recurrent units (GRUs) receive an object's current bounding box and pixel-level spatiotemporal features as inputs, respectively, and update the object's hidden states. In particular, the spatiotemporal features are extracted by a crop-resize operation using bilinear interpolation from precomputed optical flow fields. The updated hidden states are used by a location decoder (Dec) to recurrently predict the bounding boxes of the immediate future.

5.2.2 Ego-Motion Cue

Ego-motion information of the moving camera has been shown necessary for accurate future object localization [7, 42]. Let E_t be the ego-vehicle's pose at time t; $E_t = \{\phi_t, x_t, z_t\}$ where ϕ_t is the yaw angle and x_t and z_t are positions along the ground plane with respect to the vehicle's starting position in the first video frame. We predict the ego-vehicle's odometry by using another RNN encoder-decoder module to encode ego-position change vector $E_t - E_{t-1}$ and decode future ego-position

changes $\mathbf{E} = \{\hat{E}_{t+1} - E_t, \hat{E}_{t+2} - E_t, ..., \hat{E}_{t+\delta} - E_t\}$. We use the change in ego-position to eliminate accumulated odometry errors. The output $\mathbf{E}$ is then combined with the hidden state of the future object localization decoder to form the input into the next time step.

5.2.3 Missed Objects

We build a list of trackers $Trks$ per [131] to record the current bounding box $Trks[i].X_t$, the predicted future boxes $Trks[i].\hat{\mathbf{Y}}_t$, and the tracker age $Trks[i].age$ of each object. We denote all maintained track IDs as D (both observed and missed), all currently observed track IDs as C, and the missed object IDs as $D - C$. At each time step, we update the observed trackers and initialize a new tracker when a new object is detected. We use a temporarily-missing or occluded object's previously predicted bounding boxes to estimate current location, running future object localization with RoIPool features from predicted boxes (Algorithm 1). Missing object handling is essential in our prediction-based anomaly detection method to eliminate the impact of failed object detection or tracking in any given frame. For example, if an object with a normal motion pattern is missed for several frames, the FOL is still expected to give reasonable predictions except for some accumulated deviations. On the other hand, if an anomalous object is missed during tracking [131], we make a prediction using its previously predicted bounding box whose region can be substantially displaced thus can result in inaccurate predictions. In this case, some false alarms and false negatives can be eliminated by using the metrics presented in Section 5.2.4.3.

5.2.4 Traffic Anomaly Detection using FOL

Unsupervised anomaly detection methods compute anomaly scores based on prediction or reconstruction accuracy [100, 2, 46, 107]. In this section, we first present the basic anomaly metric computed from predicted bounding box accuracy. The key idea is that object trajectories and locations in non-anomalous events can be precisely predicted, while deviations from predicted behaviors suggest an anomaly. Next we propose two different strategies to compute anomaly scores using: 1) the foreground-background mask generated from predictions and 2) the prediction consistency.

5.2.4.1 Predicted Bounding Box Accuracy

One simple method for recognizing abnormal events is to directly measure the similarity between predicted object bounding boxes and their corresponding observa-

Algorithm 1: FOL-Track Algorithm

Input : Observed bounding boxes $\{X_t^{(i)}\}$ where $i \in C$, observed image evidence O_t, trackers of all objects $Trks$ with track IDs D

Output: Updated trackers $Trks$

A is the maximum age of a tracker
for $i \in C$ **do** `// update observed trackers`
  if $i \notin D$ **then**
    initialize $Trks[i]$
  else
    $Trks[i].X_t = X_t^{(i)}$
    $Trks[i].\hat{\mathbf{Y}}_t = FOL(X_t^{(i)}, O_t)$
  end
end
for $j \in D - C$ **do** `// update missed trackers`
  if $Trks[j].age > A$ **then**
    remove $Trks[j]$ from $Trks$
  else
    $Trks[j].X_t = Trks[j].\hat{Y}_{t-1}$
    $Trks[j].\hat{\mathbf{Y}}_t = FOL(Trks[j].X_t, O_t)$
  end
end

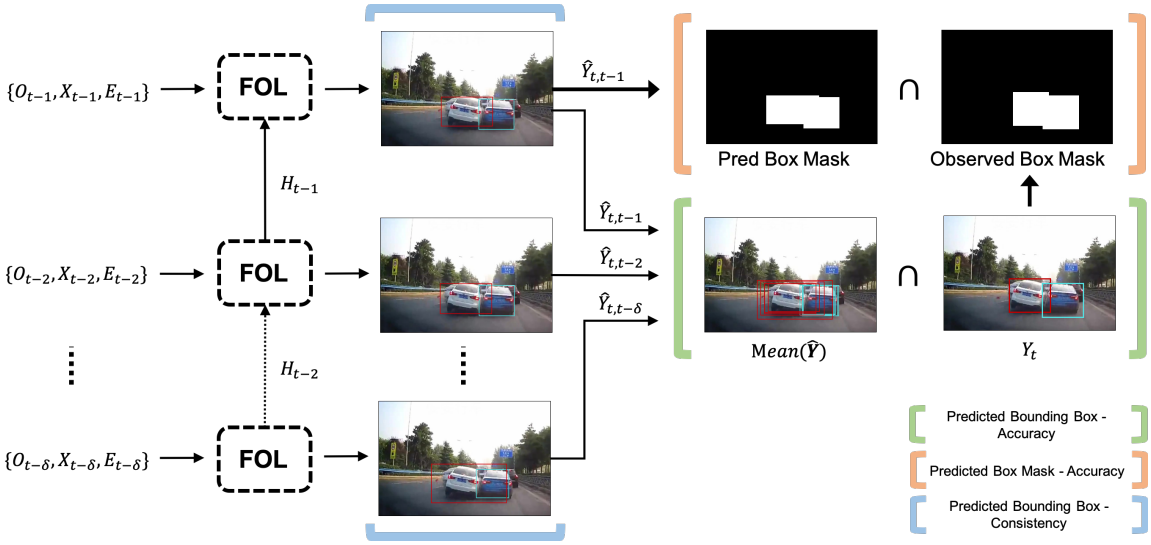


Figure 5.3: Overview of our unsupervised VAD methods. The three brackets correspond to: (1) Predicted bounding box accuracy method (green); (2) Predicted box mask accuracy method (orange); (3) Predicted bounding box consistency method (blue). All methods use multiple previous FOL outputs to compute anomaly scores.

tions. The FOL model predicts bounding boxes of the next δ future frames, i.e., at each time t each object has a bounding box predicted at each time from $t-\delta$ to $t-1$. We first average the positions of the δ bounding boxes, then compute intersection over union (IoU) between the averaged bounding box and the observed box location, where higher IoU means greater agreement between the two boxes. We average computed IoU values over all observed objects, and then compute an aggregate anomaly score $L_{bbox} \in [0, 1]$,

$$L_{bbox} = 1 - \frac{1}{N}\sum_{i=1}^{N} \mathbf{IoU}\left(\left(\frac{1}{\delta}\sum_{j=1}^{\delta} \hat{Y}^i_{t,t-j}\right), Y^i_{t_0}\right), \tag{5.1}$$

where N is the total number of observed objects, and $\hat{Y}^i_{t,t-j}$ is the predicted bounding box from time $t-j$ of object i at time t. This method, which we call FOL-IoU, relies upon accurate object tracking to match predicted and observed bounding boxes.

5.2.4.2 Predicted Box Mask Accuracy

Although tracking algorithms such as Deep-SORT [131] offer reasonable accuracy, it is still possible to lose or mis-track objects. We found that inaccurate tracking particularly happens in severe traffic accidents because of the twist and distortion of object appearances. Moreover, severe ego-motion also results in inaccurate tracking due to sudden changes in object locations. This increases the number of false negatives of the metric proposed above, which simply ignores objects that are not successfully tracked in a given frame. To solve this problem, we first convert all areas within the predicted bounding boxes to binary masks, with areas inside the boxes having value 1 and backgrounds having 0, and do the same with the observed boxes. We then calculate an anomaly score as the IoU between these two binary masks,

$$I^{(u,v)} = \begin{cases} 1, \text{ if pixel } (u,v) \text{ within box } X^i, \forall i, \\ 0, \text{ otherwise,} \end{cases} \tag{5.2}$$

$$L_{mask} = 1 - \mathbf{IoU}\big(\hat{I}_{t,t-1}, I_t\big), \tag{5.3}$$

where $I^{(u,v)}$ is pixel (u, v) on mask I, X^i is the i-th bounding box, $\hat{I}_{t,t-1}$ is the predicted mask from time $t-1$, and I_t is the observed mask at t. In other words, while the bounding box accuracy metric compares bounding boxes on an object-by-object basis, this metric simply compares the bounding boxes of all objects simultaneously. The main idea is that accurate prediction results will still have a relatively large IoU

compared to the ground truth observation. We denote the mask accuracy-based method FOL-Mask.

5.2.4.3 Predicted Bounding Box Consistency

The above methods rely on accurate detection of objects in concurrent frames to compute anomaly scores. However, the detection of anomaly participants is not always accurate due to changes in appearance and mutual occlusions. We hypothesize that visual and motion features related to an anomaly do not only appear once it happens, but are usually accompanied by a salient pre-event. We thus propose another strategy, called FOL-STD, to detect anomalies by computing consistency of future object localization outputs from several previous frames while eliminating the effect of inaccurate detection and tracking.

As discussed in Section 5.2.4.1, for each object in video frame at time t, we can collect δ bounding boxes predicted from time $t-1, t-2, .., t-\delta$. We compute the standard deviation (STD) between all δ predicted bounding boxes to measure their similarity,

$$L_{pred} = \frac{1}{N}\sum_{i=1}^{N} \max_{\{c^x, c^y, w, h\}} \mathbf{STD}(\hat{Y}^{(i)}_{t,t-j}). \tag{5.4}$$

where $\hat{Y}^{(i)}_{t,t-j}$ is the bounding box of the ith object in frame at time t predicted from the frame at time $t-j$, and c^x, c^y, w, h are the center coordinates and the width and height of a bounding box. We compute the maximum STD over the four components of the bounding boxes since different anomalies may be indicated by different effects on the bounding box, e.g., suddenly stopped cross traffic may only have large STD along the horizontal axis. A low STD suggests the object is following normal movement patterns thus the predictions are stable, while a high standard deviation suggestions abnormal motion. For all three methods, we follow [2] to normalize computed anomaly scores for evaluation.

5.2.5 Frame-object Ensemble Anomaly Detection

Our FOL based methods are recognized as object-centric methods by encoding-decoding object information. Frame-level VAD methods focus on appearance while object-centric methods focus more on object motion. We are not aware of any method combining the two. Appearance-only methods may fail with drastic variance in lighting conditions and motion-only methods may fail when trajectory prediction is imperfect. We propose to combine FOL-STD with frame prediction method AnoPred [2],

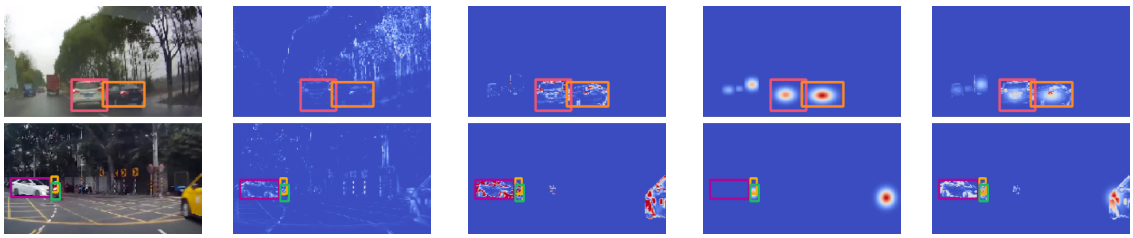
which we call the FOL-Ensemble method. AnoPred predicts one anomaly score per image pixel while our method predicts one anomaly score per object. We first map our object anomaly score to per pixel score by putting a Gaussian function at the center of each object (as introduced in Section 5.3.2). We trained each module of FOL-Ensemble independently and apply average pooling on the computed per pixel scores from two modules to compute final anomaly score. We observed this late fusion is better than fusing hidden features in an early stage and training the two models together, since their hidden features are scaled differently. AnoPred encodes one feature per frame, while FOL-STD has one feature per object.

5.3 A New Evaluation Metric

5.3.1 Critique of Current VAD Evaluation

Most VAD methods compute an anomaly score for each frame, and evaluate by plotting receiver operating characteristic (ROC) curves using temporally concatenated scores and compute an area under curve (AUC) metric. AUC measures how well a VAD method locates an anomaly along the temporal axis but ignores accuracy on spatial axes since averaged anomaly score lacks spatial information. We argue AUC is insufficient to fully evaluate VAD performance.

In computing AUC, a true positive occurs when the model predicts a high anomaly score for a positive frame. Fig. 5.4 shows two positive frames and their corresponding score maps computed by the four benchmarked VAD methods. Although the maps are different, the anomaly scores averaged from these maps are similar, meaning they are treated similarly in AUC evaluation. This results in similar AUCs among all methods, which leads to a conclusion that all perform similarly. However, AnoPred (Fig. 5.4(b)) predicts high scores for trees and other noise, while AnoPred+Mask and FOL-STD (Fig. 5.4(c) and 5.4(d)) predict high scores for unrelated vehicles. Ensemble (Fig. 5.4(e)) alleviates these problems but still has high anomaly scores outside the labeled anomalous regions. Note that score maps of FOL-STD and Ensemble are pseudo-maps introduced in Section 5.3.2.Although these methods yield similar AUCs, VAD methods should be distinguished by their abilities to localize anomalous regions. Anomalous region localization is essential because it improves reaction to anomalies, e.g. collision avoidance, and aids in model explanation, e.g. a model predicts a car-to-car collision because it finds anomalous cars, not trees or noise. This motivates a new spatial-temporal metric to evaluate how well the model detects the temporal and spatial location of the anomaly.



(a) GT image (b) AnoPred (c) AnoPred+Mask (d) TAD+ML (e) Ensemble

Figure 5.4: Anomaly score maps computed by four methods. Ground truth anomalous regions are labeled by bounding boxes. Brighter color indicates higher score.

5.3.2 Spatio-temporal Area Under Curve (STAUC)

We designed a spatial-temporal area under curve (STAUC) metric as a complement to the popular area under curve (AUC) evaluation metric. Although AUC performs well in evaluating per-frame anomaly detection results, it doesn't evaluate the model's ability to spatially localize anomalies. Fig. 5.4 shows how anomaly score is influenced by changes outside the annotated anomalous region. The STAUC computes a true anomaly region ratio (TARR) as a representation of how many anomalous pixels are located in an annotated anomalous region. The TARR is used as a weighting factor when computing AUC, resulting in STAUC. Details can be found in [1].

For each positive frame, we first calculate the true anomalous region rate (TARR), which is a scalar describing how much of the anomaly score is located within the true anomalous region,

$$\text{TARR}_t = \frac{\sum_{i \in m_t} \Delta I(i)}{\sum_{i \in M} \Delta I(i)}, \tag{5.5}$$

where $\Delta I(i)$ is the anomaly score at pixel i, M represents all frame pixels, and m_t is the annotated anomalous frame region (i.e., the union of all annotated bounding boxes). TARR is inspired by anomaly segmentation tasks where the overlap between prediction and annotation is computed [132].

Next, we calculate the spatio-temporal true positive rate (STTPR),

$$\text{STTPR} = \frac{\sum_{t \in TP} \text{TARR}_t}{|P|}, \tag{5.6}$$

where TP represents all true positive predictions and P represents all ground truth positive frames. STTPR is a true positive rate where each true positive is weighted by its TARR. We then use STTPR and the false positive rate to plot an ROC curve (which we call Spatial-Temporal ROC or STROC) and calculate the area under the curve, which gives the STAUC. Note that STAUC$\leq$AUC; the two are equal in the

best case where $\text{TARR}_t = 1\ \forall t$.

Object-centric VAD [46, 107, 44] computes per-object anomaly scores s_k instead of an anomaly score map ΔI. To generalize the STAUC metric to object-centric methods, we first create pseudo-anomaly score maps as illustrated in Fig. 5.4(d). Each object has a 2D Gaussian distribution centered in its bounding box. The pixel score is then computed as the sum of the scores calculated from all boxes it occupies,

$$\Delta I_{pseudo}(i) = \sum_{\forall k, i \in B_k} s_k \, e^{-\frac{|i_x - x_k|^2}{2w_k} - \frac{|i_y - y_k|^2}{2h_k}}, \tag{5.7}$$

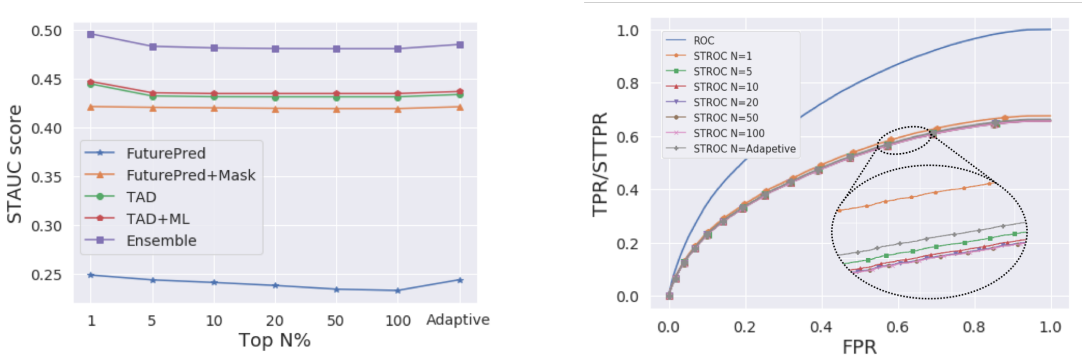
where i_x and i_y are coordinates of pixel i and $[x_k, y_k, w_k, h_k]$ are center location, width, and height of object bounding box B_k. For the Ensemble method, we take the average of ΔI and ΔI_{pseudo} as the anomaly score map in Fig. 5.4(e). This map is used as ΔI in Eq. (5.5) to compute TARR and STAUC.

TARR is not robust to anomalous region size m_t. When $m_t \ll M$, TARR could be small even though all anomaly scores are high in m_t. We thus propose selecting the top $N\%$ of pixels with the largest anomaly scores as candidates, and compute TARR from these candidates instead of all pixels. An extremely small N such as 0.01 may result in a biased candidate set dominated by false or true detections such that $TARR = 0$ or 1. To address this issue, we compute an adaptive N for each frame based on the size of its annotated anomalous region,

$$N_{adaptive} = \frac{\text{number of pixels in anomalous region}}{\text{Total number of pixels}} \times 100. \tag{5.8}$$

The average $N_{adaptive}$ of DoTA is 11.12 with a standard deviation 13.09. The minimum and maximum $N_{adaptive}$ values are 0.005 and 95.8, showing extreme cases where the anomalous object is very small (far away) or large (nearby).

A critical consideration for any new metric is its robustness to hyper parameters. We have tested STAUC with $N = [1, 5, 10, 20, 50, 100, N_{adaptive}]$ for different VAD methods per Fig. 5.5(a), STAUC slightly decreases with N increasing but stabilizes when N is large indicating STAUC is robust. Fig. 5.5(b) shows that STROC curves with different N are close, especially when $N \geq 5$, and their upper bound is the traditional ROC. $N_{adaptive}$ is selected for our benchmarks based on each frame's annotation and its corresponding mid-range STAUC value.



(a) (b)

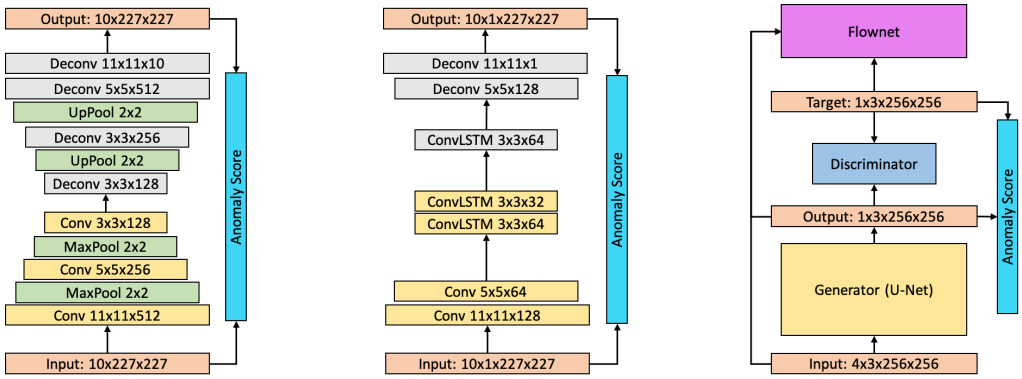
Figure 5.5: (a) STAUC values of different methods using different top $N\%$; (b) ROC curve and STROC curves of the Ensemble method with different top $N\%$.

5.4 Video Anomaly Detection Baselines

We benchmark three frame-level VAD method, ConvAE [100], ConvLSTMAE [102] and AnoPred [2] and their variants as baselines in this paper. We also compare with a K-Nearest Neighbor algorithm using I3D [67] features as extra baseline. Frame-level methods detect anomalies by either reconstructing past frames or predicting future frames and computing the reconstruction or prediction error as the anomaly score.

***K*-Nearest Neighbor Distance.** We segment each video into a bag of short video chunks with 16 frames. Each chunk is labeled as either normal or anomalous based on the annotation of the 8-th frame. We then feed each chunk into an I3D [67] network pre-trained on Kinetics dataset, and extract the outputs of the last fully connected layer as its feature representations. All videos in the HEV-I dataset are used as normal data. The normalized distance of each test video chunk to the centroid of its K nearest normal (K-NN) video chunks are computed as the anomaly score. We show results of $K = 1$ and $K = 5$ in this paper.

ConvAE [100] is a spatio-temporal autoencoder model which encodes temporally stacked images with 2D convolutional encoders and decodes with deconvolutional layers to reconstruct the input (Fig. 5.6(a)). The per-pixel reconstruction error forms an anomaly score map ΔI, and mean squared error (MSE) is computed as a frame-level anomaly score. To further compare the effectiveness of image and motion features, we implemented **ConvAE(gray)** and **ConvAE(flow)** to reconstruct the grayscale image and dense optical flow, respectively. The input to ConvAE(flow) is a stacked historical flow map with size $20 \times 227 \times 227$ acquired from pre-trained FlowNet2 [133].



(a) ConvAE [100] (b) ConvLSTMAE [102] (c) AnoPred [2]

Figure 5.6: Network architecture of benchmarked VAD methods.

We trained both variants using AdaGrad with learning rate 0.01 and batch size 24.

ConvLSTMAE [102] is similar to ConvAE but models spatial and temporal features separately. A 2D CNN encoder first captures spatial information from each frame, then a multi-layer ConvLSTM recurrently encodes temporal features. Another 2D CNN decoder then reconstructs input video clips (Fig. 5.6(b)). We implemented **ConvLSTMAE(gray)** and **ConvLSTMAE(flow)**. We trained both variants using AdaGrad with learning rate 0.01 and batch size 24.

AnoPred [2] is a frame-level VAD method that takes four continuous previous RGB frames as input and applies UNet to predict a future RGB frame (Fig. 5.6(c)). AnoPred boosts prediction accuracy with a multi-task loss incorporating image intensity, optical flow, gradient, and adversarial losses. AnoPred was proposed for surveillance cameras. However, traffic videos are much more dynamic, making future frame prediction difficult. Therefore we also benchmarked a variant of AnoPred to focus on video foregrounds. We used Mask-RCNN [3] pre-trained on Cityscapes [10] to acquire object instance masks for each frame, and apply instance masks to input and target images, resulting in an **AnoPred+Mask** method that only predicts foreground objects and ignores noisy backgrounds such as trees and billboards. In contrast to [100, 102], AnoPred uses Peak Signal to Noise Ratio, as the anomaly score with better results. Both variants are trained based on the original paper.

5.5 Implementation Details

We used the published implementations of ConvAE, ConvLSTMAE, and AnoPred and modified the input layer size to suit gray-scale or optical flow input. All these

models were trained according to the original papers.

We implemented our model in PyTorch [130] and performed experiments on a system with Pascal Nvidia Titan Xp GPU. We use ORB-SLAM 2.0 [134] for ego odometry calculation and compute optical flow using FlowNet 2.0 [133]. In our training data (HEV-I), we used provided camera intrinsic matrix. We used the same matrix for A3D and SA since these videos are collected from different dash cameras and the parameters are unavailable. We also set feature numbers 12000 to have a better performance. We use a 5×5 cropping and bilinear interpolation operator to produce the final flattened feature vector $O_t \sim \mathbb{R}^{50}$. The gated recurrent unit (GRU) [135] is our basic RNN cell. GRU hidden state sizes were set to 128 for all models. We randomly selected 3,275 videos from the DoTA dataset as the training set to train our models. To learn network parameters, we use the RMSprop [136] optimizer with default parameters, learning rate 10^{-4}, and no weight decay. Our models were optimized in an end-to-end manner, and the training process was terminated after 100 epochs using a batch size of 32.

5.6 Experiments on A3D and SA Dataset

5.6.1 Results on A3D Dataset

Table 5.1: Experimental results on A3D and SA dataset in terms of AUC. AUCs in parenthesis are results of variant metrics as explained in text.

Methods	Input	A3D	SA [19]
K-NN (K = 1)	RGB	48.0	48.2
K-NN (K = 5)	RGB	47.8	48.1
Conv-AE(gray)[100]	Gray	54.7	55.2
Conv-AE(flow)[100]	Flow	54.5	54.4
ConvLSTM-AE(gray)[102]	Gray	51.1	50.2
ConvLSTM-AE(flow)[102]	Flow	52.1	50.7
AnoPred [2]	RGB	57.5	58.4
FOL-IoU	Box+Flow	59.1(57.8)	56.7(55.2)
FOL-Mask	Box+Flow	59.7	57.8
FOL-STD	Box+Flow	64.1(63.0)	58.8(57.1)
FOL-Ensemble	RGB+Box+Flow	**64.7**	**60.2**

Quantitative Results. We evaluated baselines, a state-of-the-art method, and our proposed method on the A3D dataset. As shown in the first column of Table 5.1, our method outperforms the $K-$NN baseline as well as Conv-AE [100], ConvLSTM-AE [102] and the state-of-the-art AnoPred [2] methods. *FOL-IoU* uses the metrics in Eq. (5.1), while the result of a variation where we evaluate minimum (rather than average) IoU over all observed objects is presented in parenthesis. Computing minimum results in not only anomaly detection but also anomalous object localization. However, this minimum metric can perform worse since it is not robust to outliers such as failed prediction of a normal object, which is more frequent in videos with a large number of objects. *FOL-Mask* uses the metrics in Eq. (5.3) and outperforms the above two methods. This method does not rely on accurate tracking, so it handles cases including mis-tracked objects. However, it may mis-label a frame as an anomaly if object detection loses some normal objects. The second last row shows our best method *FOL-STD* which uses the prediction-only metric defined in Eq. (5.4). The result of a variation that computes maximum STD (rather than average) is presented in the parenthesis as well. Similar to the IoU based methods, the maximum STD metric finds the most anomalous object in the frame. By using only prediction, our method is free from unreliable object detection and tracking when an anomaly happens, including the false negatives (in IoU based methods) and the false positives (in Mask based methods) caused by losing objects. However, this method can fail in cases where predicting future locations of an object is difficult, e.g., an object with low resolution, intense ego-motion, or multiple object occlusions due to heavy traffic. At last, we present the result of *FOL-Ensemble* introduced in Section 5.2.5. It can be seen that merging the outputs of two methods from different modalities can further improve the anomaly detection performance. More analysis and visualizations of our methods can be found in Section 5.7.

Qualitative Results. Fig. 5.7 shows two sample results of our best method and the published state-of-the-art on the A3D dataset. For example, in the upper one, predictions of all observed traffic participants are accurate and consistent at the beginning. The ego car is hit at around the 30-th frame by the white car on its left, causing inaccurate and unstable predictions thus generating high anomaly scores. After the crash, the ego car stops and the predictions recover, as presented in the last two images. Fig. 5.7(d) shows a failure case where our method makes false alarms at the beginning due to inconsistent prediction of the very left car occluded by trees. This is because our model takes all objects into consideration equally rather than focusing on important objects. False negatives show that our method is not able to

detect an accident if participants are totally occluded (e.g. the bike) or the motion pattern is accidentally normal from a particular viewpoint (e.g. the middle car).

5.6.2 Results on SA dataset

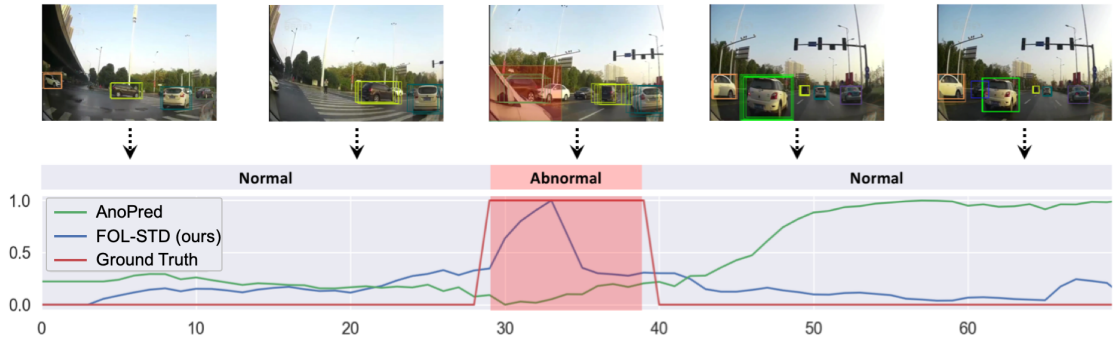
We also compared the performance of our model and baselines on the Street Accident (SA) [19] dataset of on-road accidents in Taiwan. This dataset was collected from dashboard cameras with 720p resolution from the driver's point-of-view. Note that we use SA only for testing, and still train on HEV-I dataset. We follow prior work [19] and report evaluation results with 165 test videos containing different anomalies. The right-most column in Table 5.1 shows the results of different methods on SA. In general, our best method outperforms all baselines and the published state-of-the-art. The SA testing dataset is much smaller than A3D, and we have informally observed that it is biased towards anomalies involving bikes. It also contains videos collected from cyclist head cameras which have irregular camera angles and large vibrations. Figure 5.7(d) shows an example of anomaly detection in the SA dataset.

5.7 Experiments on DoTA Dataset

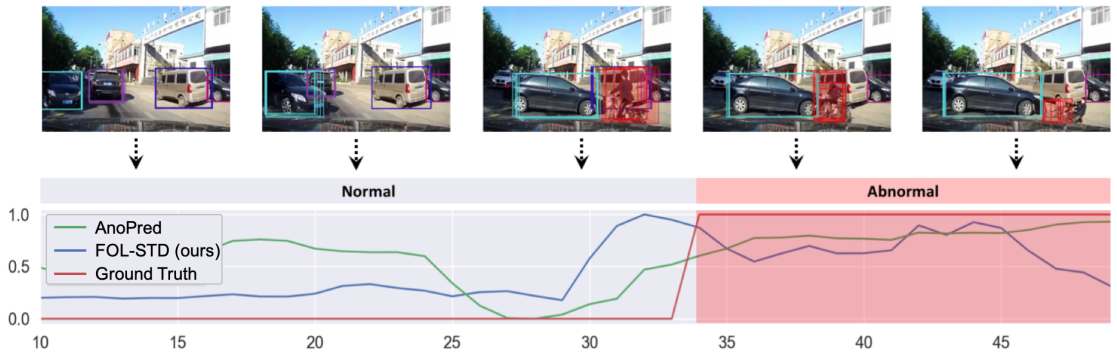
We benchmarked VAD baselines and our methods on the new DoTA dataset. DoTA also provides categorical annotations to suit video action recognition (VAR) and online action detection tasks, thus we provide extra benchmarks for state-of-the-art methods for these two tasks using the DoTA dataset. We randomly partitioned DoTA into 3,275 training and 1,402 test videos and use these splits for all tasks. Unsupervised VAD models must be trained only with non-anomalous data, so we use the precursor frames from each video for training. VAR and online action detection models are fully-supervised and thus are trained using all training data.

5.7.1 Task 1: Video Anomaly Detection (VAD)

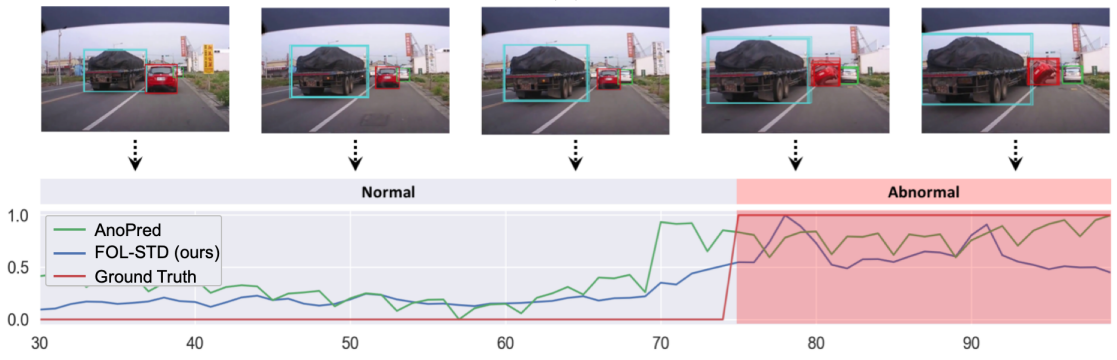
Overall Results. The top four rows of Table 5.2 show performance of ConvAE and ConvLSTMAE with grayscale or optical flow inputs. Generally, using optical flow achieves better AUC, indicating motion is an informative feature for this task. However, all baselines achieve low STAUC, meaning that they cannot localize anomalous regions well. AnoPred achieves 67.5 AUC but only 24.4 STAUC, while AnoPred+mask has 2.7 lower AUC but 17.7 higher STAUC. By applying instance masks, the model focuses on foreground objects to avoid computing high scores for



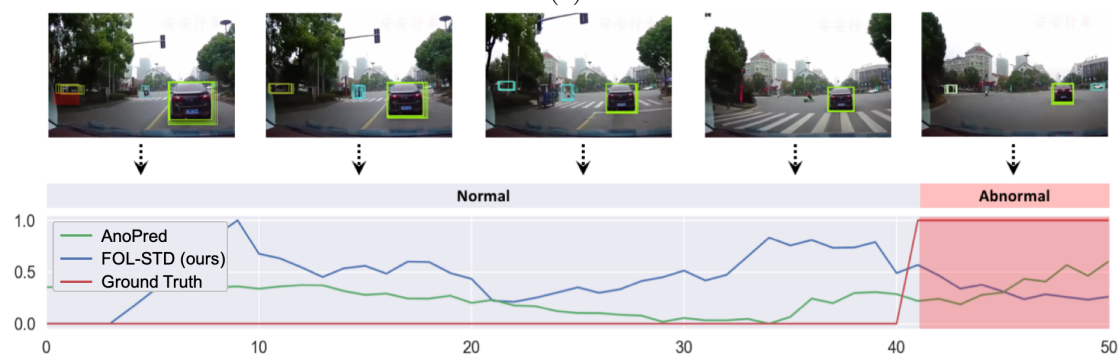
(a)



(b)



(c)



(d)

Figure 5.7: The top three rows are examples of our best method and the AnoPred [2] method on the A3D and SA dataset. The bottom row shows a failure case of our method with false alarms and false negatives.

the background, resulting in slightly lower AUC but much higher STAUC. This supports our hypothesis that *higher AUC does not always imply a better VAD model, while STAUC better captures the ability to localize anomalous regions.*

We also evaluate four variants of our methods: FOL-IoU (prediction accuracy), FOL-Mask (prediction mask accuracy), FOL-STD (prediction consistency) and FOL-Ensemble, where FOL-Ensemble is an ensemble model of FOL-STD and AnoPred+Mask. FOL-Mask outperforms FOL-IoU as it is more robust to inaccurate object tracking. FOL-STD outperforms FOL-IoU and FOL-Mask by a large margin, which shows the effectiveness of our proposed consistency metric over the existing accuracy based metric. Its higher STAUC also shows that FOL-STD is more robust to scenarios where objects are not accurately detected/tracked. FOL-STD outperforms AnoPred on both metrics by specifically focusing on object motion and location, both of which are important indicators of traffic anomalies. The FOL-Ensemble method achieves the best AUC and STAUC among all methods, indicating that combining frame-level appearance and motion features is a direction worth investigating in future VAD research, a conclusion further supported by qualitative results.

Per-class Results. Table 5.3 shows results of AnoPred, AnoPred+Mask, FOL-STD, and FOL-Ensemble broken out according to the type of anomaly. We observe that STAUC (unlike AUC) distinguishes performance by anomaly type, offering guidance as researchers seek to improve their methods. For example, Ensemble has comparable AUCs on OC (on-coming) and VP (vehicle-pedestrian) anomalies (73.4 vs 70.1) but significantly different STAUCs (56.6 vs 35.2), showing that anomalous region localization is harder on VP anomalies. Similar trends exist for the AH* (ahead), LA* (lateral), VP* (vehicle-pedestrian) and VO* (vehicle-obstacle) anoamlies. Second, frame-level and object-centric methods compensate each other in VAD as shown by the Ensemble method's highest AUC and STAUC values in most columns. Third, localizing anomalous regions in non-ego anomalies is more difficult, as STAUCs on ego-involved anomalies are generally higher. One reason is that ego-involved anomalies have better dashcam visibility and larger anomalous regions, making them easier to detect. Table 5.3 also shows the difficulties of detecting different categories, with AH*, VP, VP*, VO* and LA* especially challenging for all methods. We observed that pedestrians in VP and VP* videos become occluded or disappear quickly after an anomaly happens, making it hard to detect the full anomaly event. AH* has a similar issue since sometimes the vehicle ahead is significantly occluded by the vehicle it impacts. VO* is a rarer case in which a vehicle hits obstacles such as bumpers or traffic cones that are typically not detected and are sometimes occluded by the

Table 5.2: Benchmarks of VAD methods on the DoTA dataset.

Method	Input	AUC ↑	STAUC ↑
ConvAE (gray) [100]	Gray	64.3	7.4
ConvAE (flow) [100]	Flow	66.3	7.9
ConvLSTMAE (gray) [102]	Gray	53.8	12.7
ConvLSTMAE (flow) [102]	Flow	62.5	12.2
AnoPred [2]	RGB	67.5	24.4
AnoPred [2] + Mask	Masked RGB	64.8	42.1
FOL-IoU	Box + Flow	61.2	34.6
FOL-Mask	Box + Flow	64.0	35.0
FOL-STD	Box + Flow	**69.7**	**43.7**
FOL-Ensemble	RGB + Box + Flow	**73.0**	**48.5**

anomalous vehicle. Vehicles involved in LA* usually move toward each other slowly until they collide and stop, making the anomaly subtle and thus hard to distinguish.

Qualitative Results. Fig. 5.8(a) shows per-frame anomaly scores and TARRs of three methods on a video where they all achieve high AUCs. AnoPred+Mask has low TARR along the video, indicating failure of correctly localizing anomalous regions. FOL-STD computes high anomaly scores but low TARR in the left example due to inaccurate trajectory prediction for the left car. In the right image, it finds one of the anomalous cars but also marks an unrelated car by mistake. Ensemble combines the benefits of both with anomaly scores for the 20-30th anomaly frames always higher than normal frames. It computes high TARR during the 10-20th anomaly frames as shown in the left score map. The right map shows a failure case combining the failure of AnoPred+Mask and FOL-STD. Although these methods achieve high AUC, their spatial localization is limited according to TARR. Fig. 5.8(b) shows an ego-involved ahead collision (AH). AnoPred+Mask computes a high anomaly score in the early frames by mistake since the prediction of the left car is inaccurate, as shown in the score map. FOL-STD computes a low anomaly score for this frame and therefore the Ensemble method benefits. The right example shows the FOL-STD method correctly computes a high score for the car ahead but also another high score for the bus on the right. The ensemble benefits from AnoPred+Mask so that it focuses more attention on the car ahead instead of the bus.

[1] **ST**: collision with another vehicle that starts, stops, or is stationary; **AH**: ahead collision; **LA**: lateral collision; **OC**: on-coming collision; **TC**: turning or crossing collision; **VP**: vehicle-pedestrian collision; **VO**: vehicle-obstacle collision; **OO**: out-of-control; **UK**: unknown.

Table 5.3: Evaluation metrics of each individual anomaly class (abbreviated as two-letter short names). Ego-involved and non-ego (*) anomalies are shown separately. VO and OO columns are not shown because they do not contain anomalous traffic participants.

Method[1]	ST	AH	LA	OC	TC	VP	ST*	AH*	LA*	OC*	TC*	VP*	VO*	OO*
Individual Anomaly Class AUC:														
AnoPred [2]	69.9	73.6	**75.2**	69.7	73.5	66.3	70.9	62.6	60.1	65.6	65.4	64.9	64.2	57.8
AnoPred [2]+Mask	66.3	72.2	64.2	65.4	65.6	66.6	72.9	63.7	60.6	66.9	65.7	64.0	58.8	59.9
FOL-STD	67.3	77.4	71.1	68.6	69.2	65.1	75.1	66.2	66.8	74.1	72.0	69.7	63.8	69.2
FOL-Ensemble	**73.3**	**81.2**	74.0	**73.4**	**75.1**	**70.1**	**77.5**	**69.8**	**68.1**	**76.7**	**73.9**	**71.2**	**65.2**	**69.6**
Individual Anomaly Class STAUC:														
AnoPred [2]	37.4	31.5	32.8	34.3	33.6	24.9	25.9	15.0	12.5	13.0	20.9	14.0	8.2	8.8
AnoPred [2]+Mask	51.8	51.9	45.1	50.3	47.5	**41.0**	45.3	31.1	33.8	42.5	40.3	25.3	22.9	33.8
FOL-STD	47.4	55.6	46.3	52.2	47.2	26.6	45.1	33.6	38.5	46.9	39.3	25.6	**29.0**	**44.4**
FOL-Ensemble	**54.4**	**60.3**	**53.8**	**56.5**	**54.9**	35.2	**52.4**	**36.4**	**40.8**	**51.9**	**44.7**	**28.6**	28.6	43.5

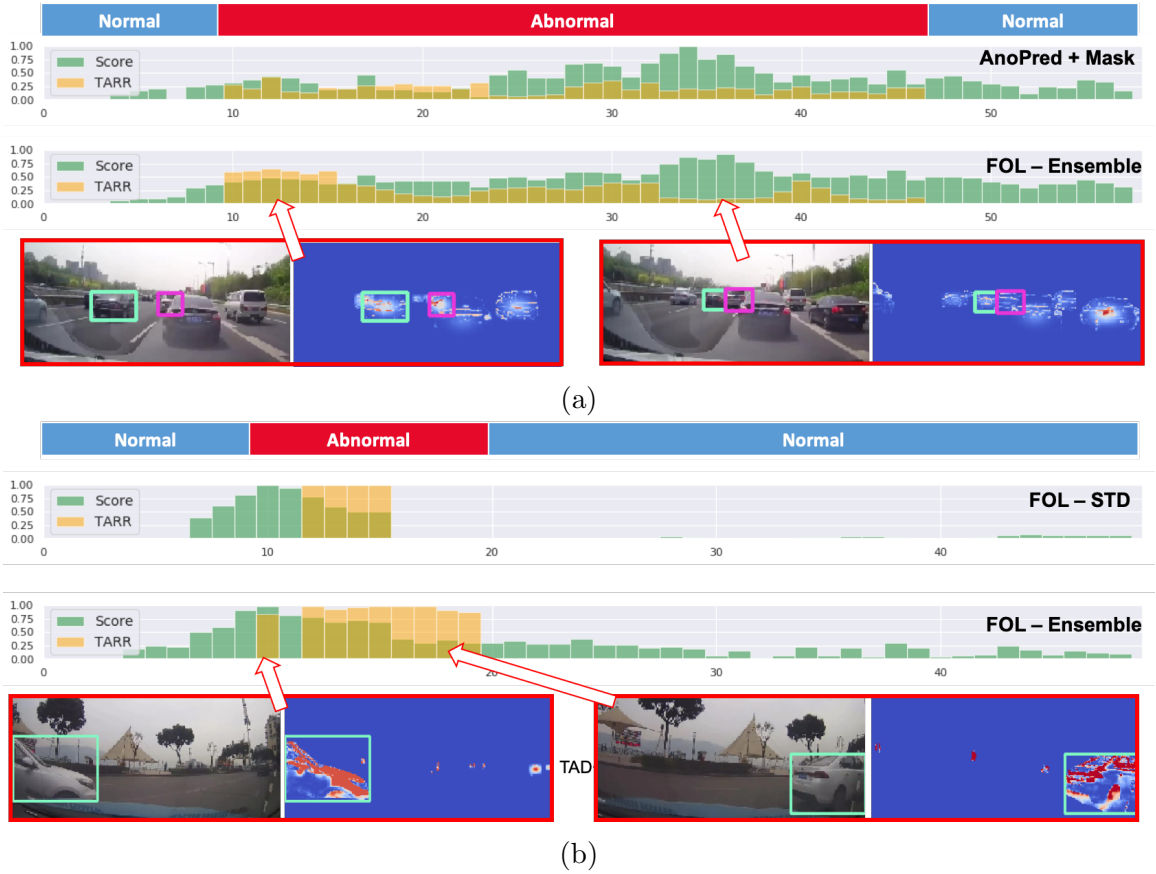


Figure 5.8: Per-frame anomaly scores and TARRs of three methods. Selected RGB frame and score maps are shown. Note that TARR only exists in positive frames.

5.7.2 Task 2: Video Action Recognition (VAR)

VAD detects the temporal range of an anomalous event but does not understand the anomaly type. The goal of Video Action Recognition (VAR) is to assign each video

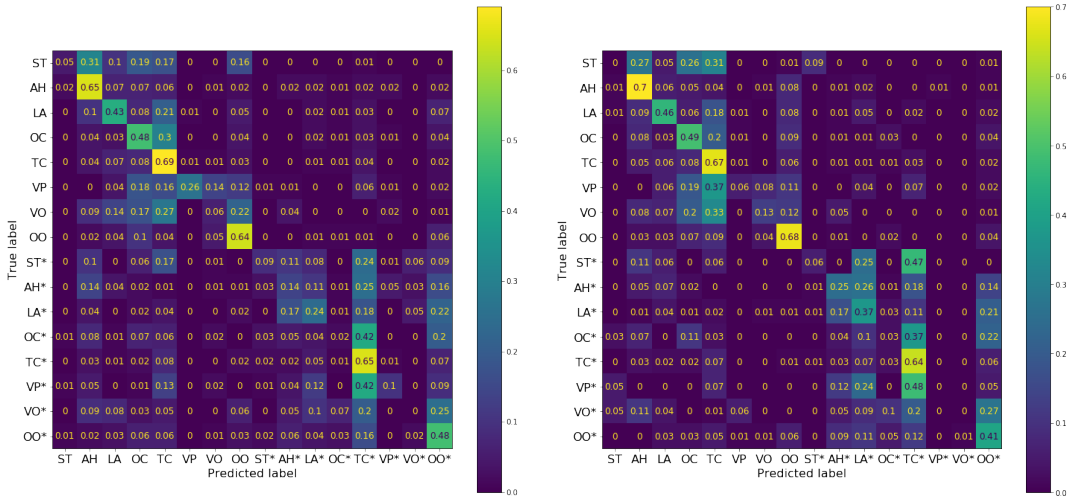
clip to one anomaly category. Taking advantage of the rich categorical annotation of the DoTA dataset, we benchmark seven VAR methods: C3D [66], I3D [67], R3D [68], MC3 [68], R(2+1)D [68], TSN [65] and SlowFast [69]. The previous training/test split is used. Unknown UK(*) anomalies are ignored, yielding 3216 training and 1369 test videos. We trained all models with SGD, learning rate 0.01 and batch size 16 on NVidia TItan XP GPUs. Models are initialized with pre-trained weights from Sports-1M [137] for C3D and Kinetics [138] for the other methods; 0.5 probability random horizontal flip offers data augmentation. For evaluation, we randomly selected ten clips from each test video (as in [69]) except TSN which uses 25 clips per video.

Table 5.4 shows the results. Although newer methods R(2+1)D and SlowFast achieve higher average accuracy, all candidates suffer from low accuracy on DoTA, indicating that traffic anomaly classification is challenging. First, distant anomalies and occluded objects have low visibility and thus are hard to classify. For example, VO (vehicle-obstacle) and VO* are hard to classify due to low visibility and diverse obstacle types (as observed in Section 5.7.1). AH (ahead)* and OC (on-coming)* are also difficult since the front or oncoming vehicles are often occluded. Second, some anomalies are visually similar to others. For example, ST (start/stop/stationary) and ST* are rare and look similar to AH (ahead) and AH* or LA (lateral) and LA* (Fig.3.3) since the only difference is whether the collided vehicle is starting, stopping, or stationary. Third, the anomaly category is usually determined by the frames around anomaly start time, while the later frames do not reveal this category clearly. We have observed 2-4% accuracy improvement when testing models only on the first half of each clip. Additional benchmarks are available in our supplement.

Table 5.4: VAR method per-class and mean top-1 accuracy with the DoTA dataset.

		Anomaly Class																
Method	backbone	ST	AH	LA	OC	TC	VP	VO	OO	ST*	AH*	LA*	OC*	TC*	VP*	VO*	OO*	AVG
TSN	ResNet50	18.2	67.2	**52.9**	**53.8**	**71.0**	0.0	0.0	61.6	0.0	14.7	25.3	6.7	48.1	9.5	0.0	**53.4**	30.2
C3D	VGG16	25.5	61.8	43.9	47.8	57.9	3.3	4.4	52.9	1.2	18.4	36.0	6.7	55.9	8.6	6.0	33.2	29.0
I3D	InceptionV1	10.0	62.4	45.8	45.8	62.2	2.8	6.9	66.6	2.4	**28.1**	24.5	4.7	60.3	9.5	5.0	37.6	29.7
R3D	ResNet18	0.0	56.5	49.6	49.8	66.6	4.4	6.2	47.7	1.8	17.6	32.2	1.0	48.3	**15.2**	6.5	48.0	28.2
MC3D	ResNet18	6.4	62.9	40.1	57.7	64.5	16.7	0.0	61.5	2.4	18.1	20.2	4.0	62.2	4.8	**6.5**	45.6	29.6
R(2+1)D	ResNet18	4.5	64.7	42.8	47.6	68.7	**25.6**	5.6	64.4	**9.4**	14.3	24.3	2.3	**64.7**	9.5	0.0	47.8	**31.0**
SlowFast	ResNet50	0.0	**70.0**	46.0	48.9	67.2	5.6	**13.1**	**68.3**	5.9	24.9	**37.2**	3.3	64.0	0.0	0.0	41.3	**31.0**

Fig. 5.9 show the confusion matrices of R(2+1)D and SlowFast, two of the best models evaluated in our experiments. In addition to Table 5 in the paper, the confusion matrix shows the most confusing categories to help us understand challenging scenarios provided in the DoTA dataset. We make three observations from Fig. 5.9. First, both models have similar confusion matrices, indicating that they perform similarly on DoTA dataset. Second, some categories are confused with other specific



(a) R(2+1)D (b) SlowFast

Figure 5.9: Confusion matrix of two state-of-the-art VAR methods on DoTA.

categories due to their similarities. Among all categories, TC, TC*, OC and OO* are four classes for which many categories are confused. One reason is that there are a large number of samples for these categories in DoTA. Another reason is the similarities among categories. For example OO* is usually an out-of-control vehicle swerving on the road and finally leaving the roadway. Other non-ego anomalies, while having their own features, often result in similar irregular motions, resulting in confusion with OO*. Third, ego-involved categories are usually not confused with non-ego categories. This indicates that although the per-class recognition is difficult, current methods could capably distinguish ego-involved and non-ego anomalies.

5.7.3 Task 3: Online Action Detection

Table 5.5: Online Video Action Detection on our DoTA dataset. "*" indicates non-ego anomaly categories[1].

	Anomaly Category																
Method	ST	AH	LA	OC	TC	VP	VO	OO	ST*	AH*	LA*	OC*	TC*	VP*	VO*	OO*	mAP
FC	**2.5**	13.9	10.6	6.2	16.3	0.8	**1.2**	21.0	0.6	2.9	3.0	0.6	8.0	**1.2**	0.7	7.6	9.9
LSTM	0.6	19.9	15.1	9.2	25.3	2.4	0.6	34.3	0.6	3.8	5.0	1.5	11.0	1.2	0.5	13.3	12.9
Encoder-Decoder	0.5	20.1	15.6	10.4	28.1	**2.9**	0.7	**39.9**	**0.8**	3.7	7.4	2.5	14.7	1.2	0.5	13.2	14.5
TRN	1.0	**22.8**	**20.6**	**15.5**	**30.0**	1.5	0.7	32.3	0.7	**4.0**	**10.2**	**2.9**	**17.0**	1.2	**0.7**	**13.8**	**15.3**

We provide benchmarks for online video action detection on DoTA dataset. Online action detection recognizes the anomaly type by only observing the current and past frames, making it suitable for autonomous driving applications. Since online

action detection does not have a full observation of the whole video sequence, online action detection is considered a more difficult task than is traditional VAR. In this supplementary material, we provide benchmarks of several state-of-the-art online action detection methods on DoTA dataset. We use the same four online methods that have been used in supervised VAD: **FC**, **LSTM**, **Encoder-decoder** and **TRN**. The only difference is that the classifiers are designed to predict only one out of the 16 anomaly categories. We use the same training configurations to train these models. Table 5.5 shows the per-class average precision (AP) and the mean average prediction (mAP).

Quantitative Results. We observe that although TRN, a state-of-the-art method, achieves the highest mAP, all methods suffer from low precision on DoTA. Similar to what we have observed in the paper's VAD and VAR experiments, online action detection is also difficult for ST, ST*, VP, VP*, VO and VO*. AH* an OC* are also difficult due to the highly occluded front of a typical oncoming vehicle. We also observe that ego-involved anomalies are easier to recognize than non-ego anomalies due to their higher visibility.

Qualitative Results. Fig. 5.10 shows some examples of TRN results on our DoTA dataset. The bar plots show the classification confidences of each frame. Cyan colors represent anomalous frames while gray colors represent background (normal) frames. We make the following observations from this experiment: 1) Transition frames between normal and abnormal events are hard to classify. For example class confidences are low at the frames where color changes, i.e., anomaly start and end frames; 2) Subsequent frames after an anomaly begins can be hard to detect. For example confidence significantly decreases at around the 40th frame of first example and the 60th frame of the third example; 3) Visually similar anomalies and gentle anomalies are hard to detect. In the bottom failure case, the confidence of ground truth anomaly class LA* is always low. These frames are either classified as background (normal) or AH* due to the fact that this LA* anomaly is visually similar to a typical AH* anomaly since this collision is relatively gentle.

5.8 Conclusion

In this Chapter, we proposed a novel FOL-based unsupervised video anomaly detection (VAD) method for driving videos. A prediction consistency metric was introduced for computing anomaly scores which is robust to inaccurate object detection and tracking in driving videos. We further introduced an ensemble method to

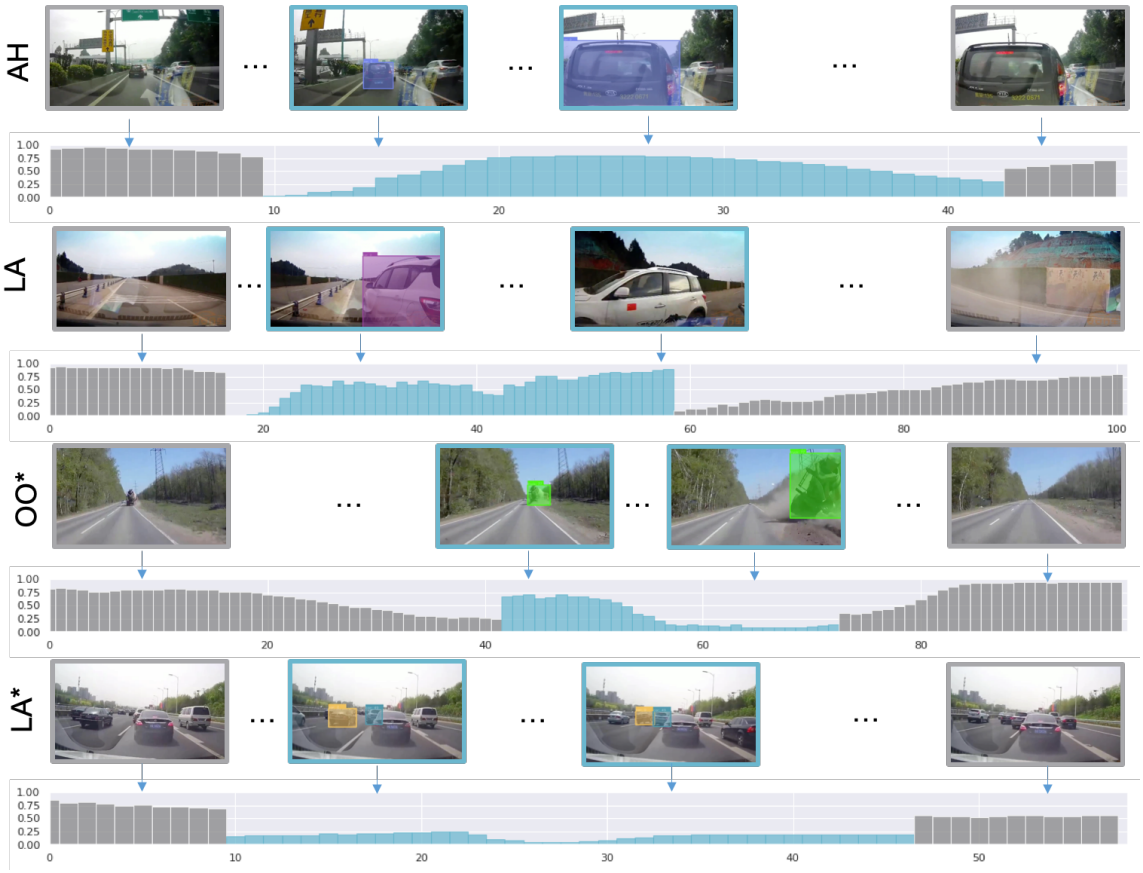


Figure 5.10: Qualitative results of Temporal Recurrent Network (TRN) on our DoTA dataset. The bar plots show classification confidences of each video frame. Gray bars are confidences of "background" (or "normal") classes while cyan bars are confidences of ground truth anomaly classes. The top two rows are two ego-involved anomalies, while the 3rd row is a non-ego out-of-control anomaly. The 4th row is a case where TRN fails to detect a lateral collision.

combine object and frame-level VAD methods to boost performance. We proposed a new spatial-temporal area under curve (STAUC) metric to better evaluate VAD performance. Experimental results show that our method achieves state-of-the-art results on DoTA in terms of both AUC and STAUC. Our DoTA dataset also enables research on video action recognition (VAR) and online action detection in driving scenarios; both of these problems are far from solved according to experimental results. Future work is needed to investigate spatio-temporal localization of anomalies in driving scenarios, early detection of traffic accidents, and validation and verification of autonomous driving systems.

CHAPTER VI

Smart Black Box

This chapter summarizes our work on the Smart Balck Box (SBB), an intelligent event data recorder. The SBB is designed to detect events of interest (EOIs), estimate data values, optimize data buffers, and prioritize high-value data over the long-term. Extensive experiments on traffic trajectory simulations and real-world driving datasets datasets show the efficiency and effectiveness of the SBB.

6.1 Introduction

Autonomous vehicles (AVs) require verification and validation (V&V) to minimize or eliminate the potential for incorrect perceptions, decisions, and actions. The industry has used the Naturalistic Field Operation Test (NFOT) project to collect a large amount of driving data and has conducted Monte Carlo simulations to enable such V&V[22]. However, NFOT data indicate low exposure rates to events of interest (EOIs) [139], suggesting that a large amount of collected data are of minimal to no interest thus could be discarded or logged with a very high compression loss factor.

Emerging AVs with redundant high-bandwidth sensors (e.g. camera, LiDAR) generate as much as 1 GB/second of raw data, a figure that scales to $\sim$ 2160 TB/year given an average driving time per person of 660 hours/year [24]. Effective capture of this raw data fleet-wide over the long-term is therefore challenging, motivating efficient data compression and discard capabilities. Recent advances in deep learning and computer vision have motivated AV research in object detection [140, 141, 3], tracking [142, 143, 131], and semantic scene segmentation [144, 145, 146, 4]. However, these modules might fail when the raw data is recorded with significant compression. Fig. 6.1 illustrates how object detection and semantic segmentation are impacted by application of lossy compression (JPEG). Compression level (1, 0.5, 0.1, 0.01) represents an image quality parameter where 1 means the highest quality. These

images show that with significant compression, computer vision algorithms cannot accurately reproduce results obtained in situ with raw image data even though the human eye can still succeed.

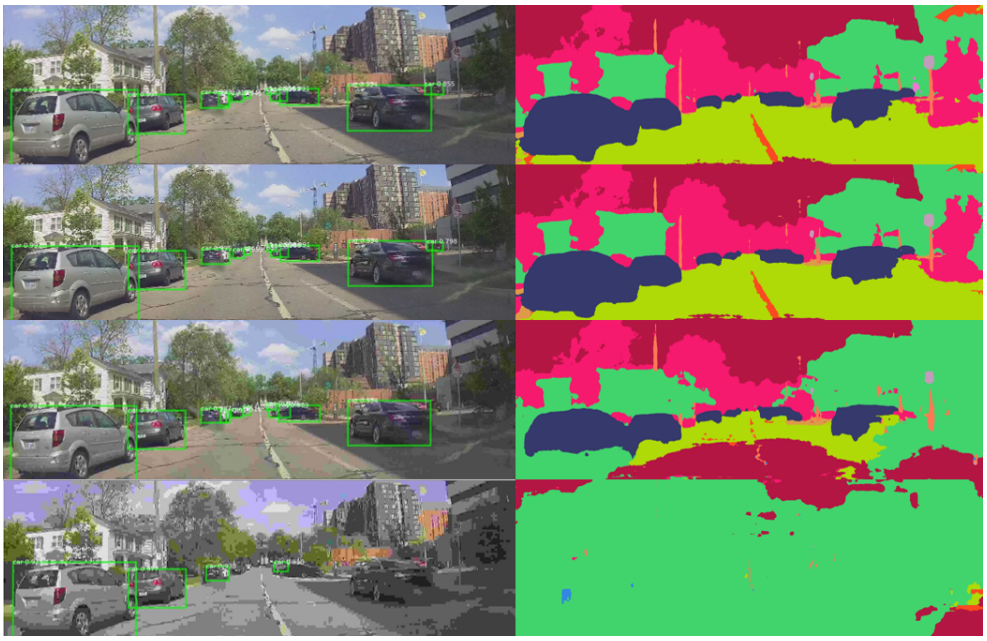


Figure 6.1: Object detection using Mask-RCNN[3] (left); semantic segmentation using DeepLabV2[4] (right). From top to bottom the images are compressed with 1, 0.5, 0.1 and 0.01 quality by a JPEG algorithm.

We design a Smart Black Box (SBB) framework [116, 117] to record high-priority high-bandwidth raw data as a supplement to logging low-bandwidth processed data, e.g., from a Controller Area Network (CAN) bus. The SBB quantifies data value and optimizes the trade-off between stored data value and size. A decision indicates how much a given data frame should be compressed for recording. Buffers containing raw data are compressed and queued so that low-value long-term data are aged out when the SBB approaches its finite storage limit.

Designing the targeted SBB functionality is challenging because there is no standard procedure for quantifying driving data value. Further, no metrics have been established to optimize long-term data collection for on-road vehicles. Additionally, globally optimizing data compression and deletion decisions would require buffering all data over a long-term trajectory. We propose a *data value information metric* based on events rarity and/or video anomaly detection to make locally-optimal compression decisions for each short-term buffer. Compressed data and their value are saved in a long-term priority queue to enable removal of the lowest-value data when finite storage limits are reached. The SBB has been evaluated with both simulated

and real-world data. We first generated a simulated NFOT highway traffic dataset using a simulator [147] that is capable of representing heterogeneous and interactive multi-vehicle traffic scenarios. Four EOIs are studied in this simulation experiment: lead car cut-in (*cutin*), host car hard-braking (*hardbraking*), cut-in conflict ($conflict$) and *crash*. A frame with none of these EOIs is classified as a *normal* frame or event. Values of these events are computed from their likelihood (rarity) among this NFOT dataset, and SBB compression and storage statistics are analyzed. To show the SBB performance in real-world data, we combined our DoTA dataset with the validation videos from BDD100K [6] to create a large-scale dataset containing anomalous events, called BDD100K+DoTA. A video anomaly detection (VAD) method and an online action detection(OAD) method are used to detect and classify the anomalous events in BDD100K+DoTA, and a Bayesian combination algorithm is used to compute the data value. Details of VAD and OAD methods can be found in Chapter V.

This work offers three primary contributions. First, we formulate a value-based data recorder, the SBB, to store high-bandwidth long-term driving data based on data value metrics. This is the first value-based automotive event data recorder to our knowledge. Second, we propose a deterministic Mealy machine (DMM) to track incoming data by value and similarity to enable high-value data and data in a common context to be buffered together. Third, we define a multi-objective constrained optimization strategy to define lossy compression factor for each buffered data frame based on computed data value for the current frame, data value for surrounding frames, and storage cost. We use a simple but effective strategy for managing finite onboard storage to ensure the highest-value data can be saved over the long-term. To demonstrate the impact of lossy compression ratio on SBB recorded data, we test popular deep learning models for object detection and semantic segmentation on first-person view images from a high-fidelity simulator called The Open Racing Car Simulator (TORCS) [148]. We show that images compressed by the SBB have much smaller size for the long run but deep-learning based object detection and semantic segmentation algorithms can still achieve high accuracy on EOIs data.

This chapter is structured as follows. Section 6.2 introduces the SBB framework and presents key definitions. Section 6.3 defines EOIs in two scenarios followed by data value and similarity metrics specifications. Section 6.5 describes the optimization strategy used along with our value-based algorithm for finite long-term data storage management. Section 6.6 and 6.7 presents results from simulation and real-world datasets followed by a brief conclusion in Section 6.8.

6.2 Smart Black Box Design

We proposes a generalized SBB framework to realize efficient data collection from emerging high-bandwidth sensors such as LiDAR and cameras given finite local storage [116, 117]. The SBB minimizes the size of recorded data and maximizes recorded data value by determining how much each data frame should be compressed. The following questions are addressed in this work:

- How are data values quantified?
- What compression factor should be applied to each data frame to trade off data value and data storage size?
- How does the SBB select data to discard given finite storage constraints?
- How do we quantify or evaluate data recording performance, i.e., what metrics should be applied?

Three data storage stages are implemented in the SBB: buffer, long-term storage and cloud database. Buffers are used to temporally cache seconds or minutes of raw data in real-time. Long-term storage relies on a finite onboard storage device capable of recording data collected over days or weeks given normal usage. The cloud database stores and manages data from vehicle fleets over months and years; data is retrieved later for post-processing.

SBB functionality is proposed in Fig. 6.2. At each time step one data frame is collected. Each frame is classified based on event detectors; a scalar in $[0, 1]$ is computed as frame data value. Similarity between a new frame and adjacent buffered frames is also computed as a scalar in $[0, 1]$. A DMM is applied to automatically manage the data buffering process. The inputs to the DMM are the data value, similarity and buffer size. It outputs buffer operation instructions, e.g., writing to buffer and emptying a buffer. The DMM formulation is detailed in Section 6.4. Once the DMM terminates, an optimization problem is solved over the buffered data to determine optimal compression quality for each frame. This process is called local buffer optimization (LBO). A data value filter can be applied to smooth the estimated value. Buffered raw data are compressed and recorded in long-term storage and are sorted based on their values. Once onboard data storage is filled, the lowest-value data are discarded to make room for new high-value data (i.e. prioritized data recording). Given internet access, stored data can be uploaded to a cloud then removed from local

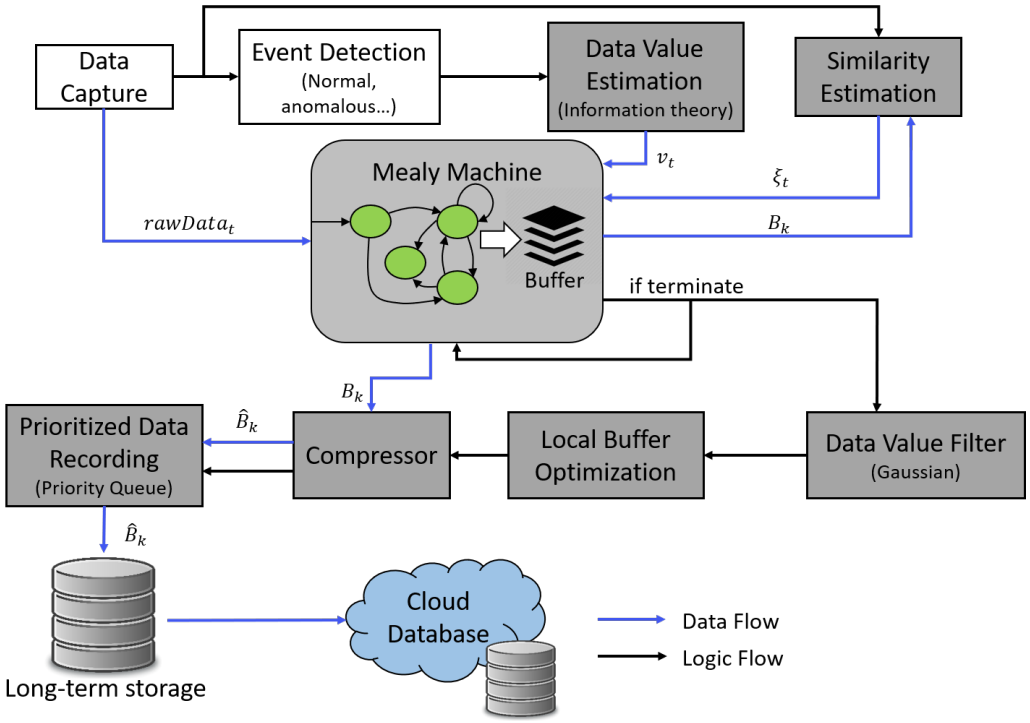


Figure 6.2: Flow chart of the SBB data recording process. Gray blocks represent SBB functions; blue blocks represent data storage and monitoring. Black arrows show the logic flow while blue arrows show the data flow.

storage. Note that data uploading and cloud database management are not studied in this paper to focus attention on compression and discard decision-making.

The SBB offers several advantages. First, DMM buffer tracking enables high-value data and data with a common context to be buffered together. Data value filtering and local buffer optimization (LBO) ensures contextual frames of EOIs are considered. Second, by separating LBO and long-term storage prioritization, the SBB makes locally optimal compression decisions which reduces memory and time complexity relative to long-term (global) optimization. Third, a long-term data storage prioritization scheme enables rapid identification of the lowest-value data to facilitate deletion as needed. Note that conventional database management methods are not applied here since the local storage is designed for data collection with no requirement for high-speed retrieval.

Some key definitions and mathematical notations used in Fig. 6.2 are introduced below:

- **Long-term storage size**: The maximum local storage capacity (e.g. in MegaBytes) that the SBB can utilize, denoted M.

- **Frame**: The sensor data received at each time as well as its value v_t and storage cost (size) c_t, represented as $f_t = (v_t, c_t, rawData_t)$.
- **Decision**: $d_t \in [0, 1]$ is the desired quality to compress $rawData_t$, 0 and 1 denote the lowest and highest data qualities, respectively.
- **Local buffer**: Short-term sequential frames are cached in a local buffer B_k, where k is buffer index. There is $f_t \in B_k$ if a frame f_t is cached in buffer B_k. Buffer length is a scalar $|B_k|$ determined by the DMM.
- **Recorded buffer**: Compressing the local buffer based on LBO output yields recorded buffer $\hat{B}_k$. Each recorded buffer is saved in the database and can be retrieved by two key parameters: the temporal index k and/or the flagged event type of the buffered data. Definition of event types is introduced later.
- **Local buffer decision vector**: Decisions for every frame in B_k form the decision vector D_k.

6.3 Data Value and Similarity Estimation

6.3.1 Data Value Estimation in TORCS simulation

We first introduce SBB data collection in multi-lane highway traffic scenarios simulated in TORCS, where one host vehicle and multiple participant vehicles are present (see Fig. 6.3) This section introduces a simplified data representation and events of interest (EOIs) that can be detected from this traffic scenario. The data value is estimated based on EOIs and then applied in Section 6.4 for buffer definition and data tracking.

The TORCS data reference frame is depicted in Fig. 6.3, where the origin O is the projection of host vehicle centroid on the road right edge. We assume full observability of host and all nearby vehicle locations (x, y) and speeds $(\dot{x})$ from processed sensor data (e.g. CAN Bus, LiDAR, radar, camera). Other physical parameters are ignored and the lane widths and locations are assumed constant for simplicity. In this paper we only consider the closest vehicles in six regions: front left (1), rear left (2), front center (3), rear center (4), front right (5), rear right (6). Also, we compute x_1 to x_n as relative distance to the host car so that $x_0 = 0$ can be ignored, resulting in a 20 dimensional feature vector:

$$X = [y_0, \dot{x}_0, x_1, y_1, \dot{x}_1, ..., x_6, y_6, \dot{x}_6] \tag{6.1}$$

where subscript 0 indicates host vehicle features and 1...6 are the six neighbor cars. For a region where no vehicle exists, we set $x_i = 100\,m$, $\dot{x}_i = 0\,m/s$ and y_i equal to the location of the corresponding lane center line.

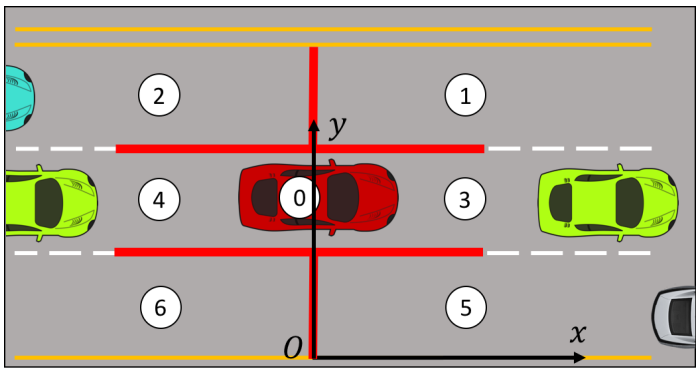


Figure 6.3: Three-lane traffic scenario and reference frame. Circled numbers indicate the host vehicle (in red) and closest vehicles in six surrounding regions, separated by red lines.

We classify each observed data frame as either *normal* or one of the four EOIs: *cutin*, *hardbraking*, *conflict* and *crash*. A *normal* frame is a frame that is not classified as any of the EOIs. Note that we use pre-defined physical metrics for anomaly detection instead of more generalized machine learning based methods [19, 21, 44], since the simulator we used in experiment is able to provide perfect measurements of traffic states.

Cutin : A *cutin* event is recognized when the closest front vehicle, represented by $(x_i, y_i, \dot{x}_i)$, enters the lane of the host vehicle as shown in Fig. 6.4(a). The cut-in range is $R = x_i - x_0$. Let $\dot{y}_i$ be the lateral velocity of the cut-in vehicle, and w_{ln} and w_c be the widths of lane and vehicle, respectively. A cut-in event is defined by

$$\begin{aligned} 0 < y_i - y_0 < \frac{w_{\text{ln}} + w_{\text{c}}}{2} \text{ and } \dot{y}_i < 0 \\ \text{or } -\frac{w_{\text{ln}} + w_{\text{c}}}{2} < y_i - y_0 < 0 \text{ and } \dot{y}_i > 0 \end{aligned} \tag{6.2}$$

Hardbraking : A *hardbraking* event occurs when the deceleration of a car is greater than a hard deceleration threshold $\ddot{x}_{\text{hb}}$ as in Fig. 6.4(b). In this paper we define $\ddot{x}_{\text{hb}} \approx -4.4\,m/s^2$ per [83].

Conflict : A *conflict* event is when the host car is in the proximity zone of the lead car during the cut-in event as shown in Fig.6.4(c). The proximity zone of a lead car

is the rectangle area bounds its geometric contour from 4 feet in front of its front bumper to 30 feet behind its rear bumper [149]. Its length and width are defined as $(l_{\text{pr}}, w_{\text{c}})$.

Crash : A *crash* event occurs when one car collides with another car from any direction. Since car yaw angle is ignored for simplicity, we detect a crash by

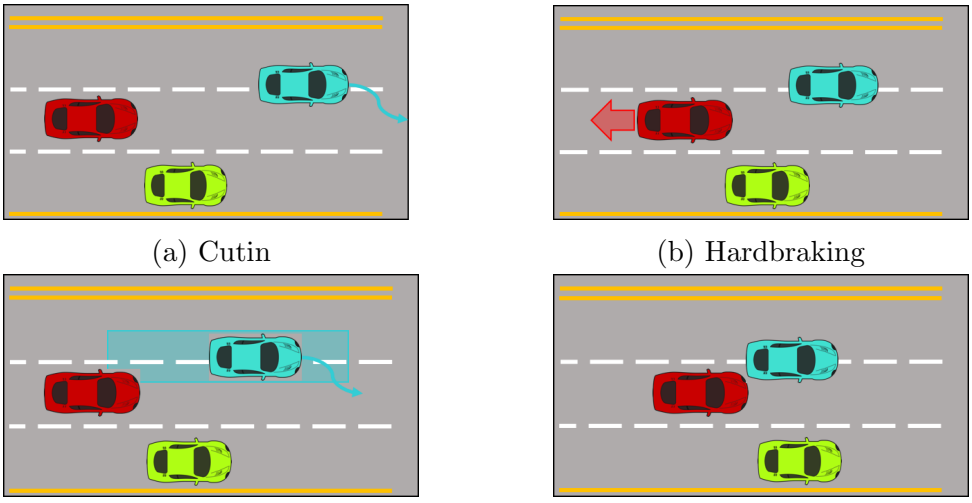
$$|x_i - x_0| \leq l_{\text{c}} \text{ and } |y_i - y_0| \leq w_{\text{c}} \tag{6.3}$$

where l_{c} is vehicle length. Fig. 6.4(d) shows a crash event.

Although multiple EOIs may be detected in a single frame, we mark each frame with the single highest-value EOI to to simplify value assignment. Therefore the *normal* event plus the four EOIs constituted the event space $\mathbb{E}$ for data value assignment, defined as:

$$\mathbb{E} = \{\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4, \varepsilon_5\} = \{normal, cutin, hardbraking, conflict, crash\}$$

The above list of EOIs can be extended and generalized for any data collection task. Advanced detection models can be applied to detect more complicated EOIs. In the following section we present a generalized value metrics computation which can be applied to any EOI as long as an event likelihood probability is provided.

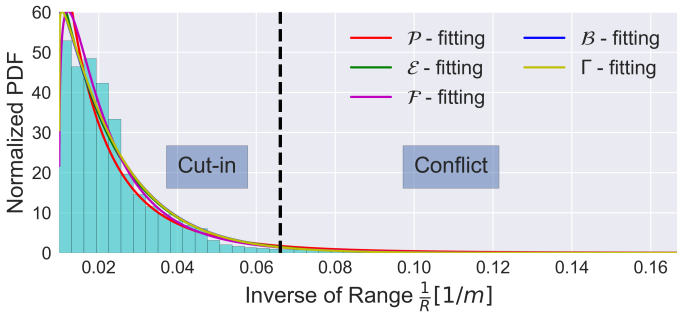


(a) Cutin (b) Hardbraking

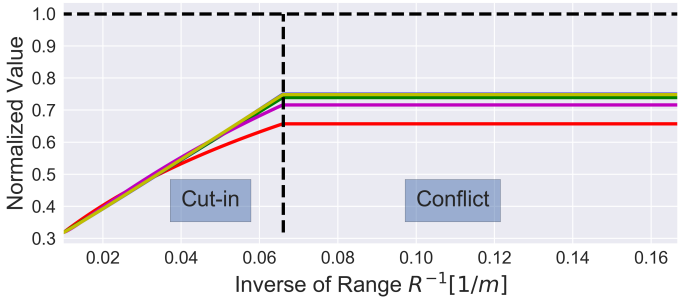
(c) Conflict (d) Crash

Figure 6.4: Four pre-defined EOIs for the ego car (red). The blue rectangle is the proximity zone of the blue car (better in color).

We assume a large naturalistic driving data set which contains all previously defined EOIs has been collected and processed. Given a frame f_t whose corresponding



(a)



(b)

Figure 6.5: Conditional PDF of inverse cut-in range $\Pr(R^{-1}|\varepsilon_2)$ and the calculated data value $v(R_t, \varepsilon_2)$ (normalized)

event type is $\varepsilon_j \in \mathbb{E}$, data value is based on the likelihood (rarity) of ε_j. According to information theory, a lower-probability event carries more information than a higher-probability event, so $v_t = v(\varepsilon_j)$ can be estimated as the information measure of ε_j. We use previously defined EOIs and assume 100% detection confidence for simplicity. Value estimates of different events are given in this section.

Constant value events. We assume a *normal* event has constant low value while *hardbraking*, *conflict* and *crash* events have constant high values. Values of these events $\varepsilon_j \in \{\varepsilon_1, \varepsilon_3, \varepsilon_4, \varepsilon_5\}$ are computed using (6.4) given $\Pr(\varepsilon_j)$ the event likelihood.

$$v(\varepsilon_j) = -\log_2\big(\Pr(\varepsilon_j)\big) \tag{6.4}$$

In this paper we set $v(\varepsilon_5) = 1$ (highest value) and the values of other events are normalized over $[0, 1]$.

Dynamic value event. The value of a *cutin* event is not a constant but a function of cut-in range R. Given a *cutin* event, the conditional probability density function (PDF) of R is represented by $\Pr(R|\varepsilon_2)$. Large R indicates a low-value *cutin*, which is observed in the majority of the dataset. Overly small R (e.g. $15\,m$) and overly large

R (e.g. $100\,m$) are rare in naturalistic driving data, but only small R contains high value. Thus the value of a *cutin* with a measured R_t can be computed from (6.5):

$$v(R_t, \varepsilon_2) = -\log_2 \big(\Pr(R < R_t|\varepsilon_2)\ \Pr(\varepsilon_2) \big) \tag{6.5}$$

where $\Pr(\varepsilon_2)$ is the probability of *cutin* events computed from the dataset. In this work, we use the conditional PDF of R^{-1} instead of R as suggested in [22] to put the small R value in the tail of the distribution. Fig.6.5 shows the fitting result of $\Pr(R^{-1}|\varepsilon_2)$ with Pareto distribution ($\mathcal{P}$), exponential distribution ($\mathcal{E}$), f distribution ($\mathcal{F}$), beta distribution ($\mathcal{B}$), and gamma distribution (Γ) [22, 150]. We use the Bayesian Information Criterion (BIC) for distribution fitting model selection since all candidate models are in the exponential family [151]. The BIC is computed as

$$BIC(\Theta, \mathcal{M}) = k \ln n - 2 \sum_{i=1}^{n} \ln \Pr\left(R_t^{-1}|\varepsilon_2; \Theta, \mathcal{M}\right) \tag{6.6}$$

where n is the number of samples, $\mathcal{M} \in \{\mathcal{P}, \mathcal{E}, \mathcal{F}, \mathcal{B}, \Gamma\}$ is a candidate model, and Θ is the parameter vector with length k that maximizes the likelihood. The model with lowest BIC is selected which for this paper is the $\mathcal{F}$ distribution with $\Theta = [\theta_1, \theta_2]$. The fitted conditional PDF is given in (6.7).

$$\begin{aligned} &\Pr(R^{-1}|\varepsilon_2) = \Pr(R^{-1}|\varepsilon_2; \Theta, \mathcal{F}) \\ &= \frac{1}{\beta(\frac{\theta_1}{2}, \frac{\theta_2}{2})} (\frac{\theta_1}{\theta_1})^{\frac{\theta_2}{2}} R^{1-\frac{\theta_1}{2}} (1 + \frac{\theta_1}{\theta_2} R^{-1})^{-\frac{\theta_1+\theta_2}{2}} \end{aligned} \tag{6.7}$$

where $\beta(a, b) = \frac{(a-1)!(b-1)!}{(a+b-1)!}$. Equation (6.5) can be written as (6.8) using the fitted distribution of R^{-1}.

$$\begin{aligned} &v(R_t, \varepsilon_2) \\ &= -\log_2 \left[\left(1 - \int_0^{R_t^{-1}} \Pr(R^{-1}|\varepsilon_2)\, dR^{-1} \right) \Pr(\varepsilon_2) \right] \end{aligned} \tag{6.8}$$

6.3.2 Data Value Estimation with a Real-world Dataset

This section introduces our method to compute data value in a real-world dataset to enable the DMM module to group buffers and compute optimal compression factors. Similar to [116, 117], we define the value of a data frame as a measure of data novelty. The data value is determined by: 1) The anomaly score estimated by a video

anomaly detection (VAD) module; 2) The anomaly category detected by an online action detection (OAD) module.

6.3.2.1 Video Anomaly Detection (VAD)

As introduced in Chapter V, a VAD algorithm takes observed image frames and predicts an anomaly score for each frame as a description of the degree of abnormality of that frame. Existing VAD algorithms can be categorized as frame-level VAD and object-level VAD. A frame-level VAD algorithm reconstructs or predicts image frames (e.g., in RGB or grayscale) and computes the L2 error of reconstruction or prediction as the anomaly score [100, 102, 2]. An object-level algorithm, on the other hand, predicts object appearance and/or motions and computes the anomaly score based on prediction error [107, 46] or consistency [44, 1].

We trained our FOL-STD algorithm (in Chapter V) using the DoTA (in Section 3.3) dataset to estimate an anomaly score s_i of a frame i and use this score to inform our SBB value estimation. To be specific, we trained the TAD algorithm in [44] using the DoTA dataset and applied it to our data value estimation module.

6.3.2.2 Online Action Detection (OAD)

While the anomaly score from VAD provides information about anomaly probability in a frame, it lacks the knowledge of anomaly category, which is important information for determining data value in long-term driving according to Section 6.3.1. Categorizing anomalous events is essential to SBB design since: 1) It allows the SBB to prioritize high value categories when storage limit is encountered; and 2) It allows the SBB to focus on specific event types based on user requests.

We implement an off-the-shelf OAD algorithm to obtain a confidence score vector o_i for a frame i, which is then combined with anomaly score s_i to estimate data value. In this work, we trained an OAD algorithm called the temporal recurrent network (TRN) [73] using the DoTA dataset [1]. TRN outputs a 17-D vector $o_i = [P_i(c_1), P_i(c_2), \ldots, P_i(c_{17})]$ for each frame, which represents the confidence score for each class as defined in [1]. Since $\sum_{j=1}^{17} P_i(c_j) = 1$ all confidence scores sum to 1.

6.3.2.3 Independent Bayesian Classifier Combination

Both VAD and OAD provide estimates of the probability that a frame is anomalous; VAD gives the anomaly score value s_i, while OAD class probabilities for anomalies can be summed to generate a score $t_i = \sum_{j=1}^{17} o_{i,j}$. By assuming that these

two classifiers are conditionally independent, we can apply the Independent Bayesian Classifier Combination (IBCC)[152] to fuse their outputs into one score.

Let v_i be the ground truth anomaly indicator of frame i, and let $v_i = 1(0)$ indicate an anomalous (normal) frame. We assume v_i is generated from a binomial distribution with class probabilities $\boldsymbol{p} = [p_0, p_1]$, where p_0 and p_1 are probabilities of normal and anomaly. $\boldsymbol{p}$ has Dirichlet prior $\boldsymbol{\nu} = [\nu_0, \nu_1]$. We then digitize the anomaly score $\hat{s}_i$ and the OAD score t_i using a threshold ϵ so that scores greater than ϵ are mapped to 1 and are otherwise 0. We assume that $\hat{s}_i$ and $\hat{t}_i$ are generated from binomial distributions conditioned on the ground truth anomaly status v_i with class probabilities $\boldsymbol{\pi}_k^{(s)} : \pi_{k,l}^{(s)} p(\hat{s}_i = l | v_i = k)$ and $\boldsymbol{\pi}_k^{(t)} : \pi_{k,l}^{(t)} p(\hat{t}_i = l | v_i = k)$, respectively, where $l, k \in \{0, 1\}$. $\boldsymbol{\pi}_k^{(s)}$ and $\boldsymbol{\pi}_k^{(t)}$ also have Dirichlet priors $\boldsymbol{\alpha}_k^{(s)} = [\alpha_{k,0}^{(s)}, \alpha_{k,1}^{(s)}]$ and $\boldsymbol{\alpha}_k^{(t)} = [\alpha_{k,0}^{(t)}, \alpha_{k,1}^{(t)}]$. $\boldsymbol{\pi}^{(s)}$ and $\boldsymbol{\pi}^{(t)}$ are called the confusion matrices for random variables $\hat{s}_i$ and $\hat{t}_i$ respectively. Then, with N frames, we have a joint distribution for the IBCC model [152] given by:

$$
\begin{aligned}
&p(\boldsymbol{p}, \boldsymbol{\pi}^{(s)}, \boldsymbol{\pi}^{(t)}, \boldsymbol{v}, \boldsymbol{s}, \boldsymbol{t} | \boldsymbol{\alpha}^{(s)}, \boldsymbol{\alpha}^{(t)}, \boldsymbol{\nu}) = \\
&\prod_{i=1}^{N} (p_{v_i} \pi_{i,\hat{s}_i}^{(s)} \pi_{v_i,\hat{t}_i}^{(t)}) p(\boldsymbol{v} | \boldsymbol{\nu}) p(\boldsymbol{\pi}^{(s)}, \boldsymbol{\pi}^{(t)} | \boldsymbol{\alpha}^{(s)}, \boldsymbol{\alpha}^{(t)})
\end{aligned}
\tag{6.9}
$$

We use the Variational Bayes IBCC [152] to approximate the unknown variables $\boldsymbol{p}$, $\boldsymbol{\pi}^{(s)}$, $\boldsymbol{\pi}^{(t)}$, $\boldsymbol{v}$. We approximate the posterior distribution over these variables to be:

$$
q(\boldsymbol{p}, \boldsymbol{\pi}^{(s)}, \boldsymbol{\pi}^{(t)}, \boldsymbol{v}) = q(\boldsymbol{v}) q(\boldsymbol{p}) q(\boldsymbol{\pi}^{(s)}) q(\boldsymbol{\pi}^{(t)}), \tag{6.10}
$$

where $q(\cdot)$ represents the posterior probabilities. Variational Bayes iteratively updates $q(\boldsymbol{v})$ and $q(\boldsymbol{p}, \boldsymbol{\pi}^{(s)}, \boldsymbol{\pi}^{(t)})$ until the variables converge. The anomaly posterior $q(\boldsymbol{v})$ is updated by:

$$
q(v_i = k) = E_{\boldsymbol{v}}[v_i = k] = \frac{\rho_{i,k}}{\rho_{i,0} + \rho_{i,1}}, \tag{6.11}
$$

$$
\ln(\rho_{i,k}) = E_{\boldsymbol{p}}\big[\ln(p_k)\big] + \sum_{\hat{s}_i \in \{0,1\}} p(\hat{s}_i) E_{\boldsymbol{\pi_k}}\big[\ln \pi_{k,\hat{s}_i}^{(s)}\big] + \sum_{\hat{t}_i \in 0,1} p(\hat{t}_i) E_{\boldsymbol{\pi_k}}[\ln \pi_{k,\hat{t}_i}^{(t)}]. \tag{6.12}
$$

$\boldsymbol{\nu}$, the Dirichlet prior for $\boldsymbol{p}$, is updated using the current expected number of normal

and anomalous frames. The posterior $q(\boldsymbol{p})$ is then re-computed from the prior:

$$q(\boldsymbol{p}) = \text{Dir}(\boldsymbol{p}|\boldsymbol{\nu'}), \quad \text{where } \nu'_k = \nu_k + \sum_{i=1}^{N} E_{\boldsymbol{v}}[v_i = k]. \tag{6.13}$$

The first term for the $q(\boldsymbol{v})$ update in Eq. (6.12) is computed as:

$$E_{\boldsymbol{p}}[\ln(p_k)] = \psi(\nu'_k) - \psi(\nu'_0 + \nu'_1), \tag{6.14}$$

where ψ is a Digamma function.

$\boldsymbol{\alpha}^{(s)}$ and $\boldsymbol{\alpha}^{(t)}$, the Dirichlet priors for $q(\boldsymbol{\pi^{(s)}})$ and $q(\boldsymbol{\pi^{(t)}})$, are updated similarly using the expected anomaly values. Here, $q(\boldsymbol{\pi^{(s)}})$ and $q(\boldsymbol{\pi^{(t)}})$ are updated from the original $\boldsymbol{\pi^{(s)}}$ and $\boldsymbol{\pi^{(t)}}$:

$$q(\boldsymbol{\pi}_k^{(s)}) = \text{Dir}(\boldsymbol{\pi}_k^{(s)}|\boldsymbol{\alpha}'^{(s)}), \text{where } \alpha'^{(s)}_{k,l} = \alpha^{(s)}_{k,l} + \sum_{i=1}^{N} p(\hat{s}_i = l) E_{\boldsymbol{v}}[v_i = k], \tag{6.15}$$

$$q(\boldsymbol{\pi}_k^{(t)}) = \text{Dir}(\boldsymbol{\pi}_k^{(t)}|\boldsymbol{\alpha}'^{(t)}), \text{where } \alpha'^{(t)}_{k,l} = \alpha^{(t)}_{k,l} + \sum_{i=1}^{N} p(\hat{t}_i = l) E_{\boldsymbol{v}}[v_i = k]. \tag{6.16}$$

The second and third terms for the $q(\boldsymbol{v})$ update in Eq. (6.12) are computed as:

$$E[\ln(\pi^{(s)}_{k,l})] = \psi(\alpha'^{(s)}_{k,l}) - \psi(\alpha'^{(s)}_{k,0} + \alpha'^{(s)}_{k,1}), \tag{6.17}$$

$$E[\ln(\pi^{(t)}_{k,l})] = \psi(\alpha'^{(t)}_{k,l}) - \psi(\alpha'^{(t)}_{k,0} + \alpha'^{(t)}_{k,1}). \tag{6.18}$$

Finally, we update $q(\boldsymbol{v})$ using $E_{\boldsymbol{p}}[\ln(p_k)]$, $E[\ln(\pi^{(s)}_{k,l})]$, and $E[\ln(\pi^{(t)}_{k,l})]$ as in Eq. (6.11) and Eq. (6.12). After convergence, we have $E_{\boldsymbol{v}}[v_i = 1]$ for each frame i as the IBCC estimated data value v_i.

Value Scaling. Some applications may want to bias the data value in favor of or against specific anomaly classes. We use the OAD output to scale the value according to user-defined biases. For each anomaly class j except for $j = 1$, the normal class, the user may define a bias $b_j \in [-1, 1]$. A bias of $+1$ indicates a heavy bias towards class j, while a bias of -1 indicates a heavy bias against class j. A bias of 0 indicates no bias towards or against class j. Then, for frame i, we define scale factor $k_i = 1 + \sum_{j=2}^{17} b_j o_{i,j}$. Using this scale factor, we compute the scaled value $v_i = \min\{k_i v_i, 1\}$ for IBCC value or $v_i = \min\{k_i s_i, 1\}$ for anomaly score value.

6.3.3 Data Similarity Metrics

We compute a similarity metric $\xi_t(f_t, B_k)$ to represent the similarity of a new incoming data frame f_t with the current buffer B_k. A high similarity between B_k and f_t indicates the two frames may belong to the same driving scenario so that f_t might be appended to B_k for completeness. This similarity metric together with the data value metric are input to a SBB deterministic Mealy machine (DMM) to determine whether to buffer f_t together with B_k or not, as introduced below in Section 6.4.

Consider the current buffered data time series B_k with sequential feature vectors $[X_1, X_2, ..., X_{|B_k|}]$ and a new single frame f_t with feature vector X_t. The difference between B_k and f_t, $\Delta(f_t, B_k)$, is defined as the standardized Euclidean distance between X_t and previous feature vectors in (6.19). Note that all X are normalized to $[0, 1]$.

$$\Delta(f_t, B_k) = \sqrt{\sum_{j=1}^{N} \frac{(X_t^{(j)} - \mu_j)^2}{\sigma_j^2}} \tag{6.19}$$

where $X_t^{(j)}$ is the jth element of feature vector X_t, N is the total number of features, and μ_j and σ_j are the mean and standard deviation of the jth feature in buffer B_k, respectively. The similarity score $\xi \in (0, 1]$ is computed in (6.20). The higher the ξ value, the more similar f_t is to B.

$$\xi(f_t, B_k) = e^{-\Delta(f_t, B_k)} \tag{6.20}$$

6.4 Online Data Buffering

The SBB must decide when to start buffering data and when to stop and send the data to the LBO module, i.e., the start and end points of data segments. We refer to this decision as *buffer tracking*. The buffer tracking process is modeled as a deterministic Mealy machine (DMM) [153] as shown in Fig.6.6. The DMM is defined as a 6-tuple $(S, S_0, \Sigma, \Lambda, T, G)$; each element is introduced below.

- States: $S = \{s_i\}_{i=1}^{4} = \{$*active, buffering, waiting, terminate*$\}$.
- Start state: $S_0 =$ *active*.
- Input: $\Sigma = \{e_i\}_{i=1}^{6}$, per Table 6.1.
- Output: $\Lambda = \{a_i\}_{i=1}^{7}$; actions a_i correspond to buffer decisions per Table 6.2.

- Transitions $T : S \times \Sigma \rightarrow S$ per Fig. 6.6.
- Output function $G : S \rightarrow \Lambda$: mapping from states to outputs, per Table 6.2.

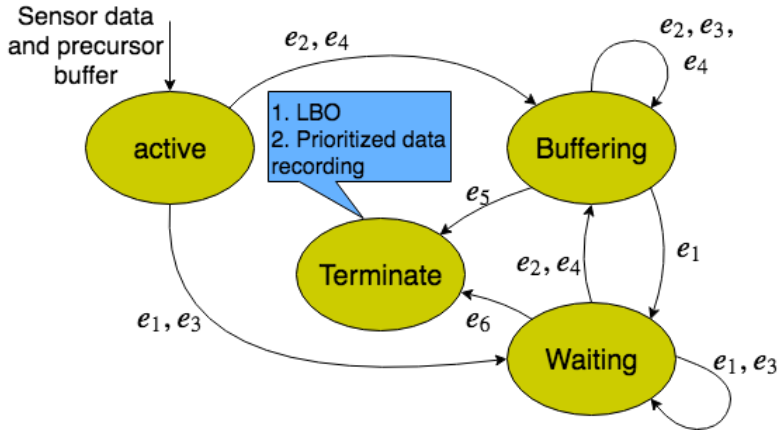


Figure 6.6: DMM for data buffer tracking decisions. The blue box highlights SBB actions executed when each buffer tracking DMM execution sequence terminates.

6.4.1 Mealy Machine States

Active: The DMM is initialized in the active state with a precursor buffer B_{pre} containing contextual data frames. Given input, the DMM transfers to the buffering or waiting state. Four inputs are possible for this state: $\{e_1, e_2, e_3, e_4\}$.

Buffering: In the buffering state, a new data frame is stored in "major" buffer B_{maj}. Data in B_{maj} will eventually be used for LBO when the DMM terminates. The DMM can transit to buffering, waiting or terminate states from the buffering state according to received inputs. All input are possible except e_6.

Waiting: In this state, a new data frame is stored in a "wait" buffer B_{wait}. B_{wait} will be emptied when the DMM transits to buffering state or terminates. The machine can transit to the waiting, buffering or terminate state from waiting state based on inputs. All input are possible except e_5.

Terminate: The DMM terminates once transits to the terminate state and the resulted B_{maj} is sent to the following modules. A new DMM execution cycle will be initialized to track the next buffer.

6.4.2 Input alphabet:

The input alphabet is generated based on data value, data similarity, major buffer size, and waiting time. Data value is estimated from (6.4) and (6.8). We set a

threshold so that a frame f_t with value $v_t > v(\varepsilon_1)$ indicates an EOI. Data similarity $\xi(f_t, B_k)$ is computed in (6.20). Note that B_k is the major buffer B_{maj} if it is not empty; otherwise, B_k is the wait buffer B_{wait}. We set threshold ξ_0 so that $\xi(f_t, B_k) > \xi_0$ indicates that frame f_t and buffered data B_k are from similar driving scenarios. The DMM thus appends f_t to B_k unless the buffer size limit is reached.

The major buffer size (number of frames) is represented as $|B_{maj}|$. We set a threshold so that the DMM state transits to *terminate* when it reaches the largest allowed size T_{maj} (event e_5). The waiting time is represented by the size of wait buffer $|B_{wait}|$. When the DMM state is *buffering*, B_{wait} is empty so that waiting time is 0. Each time the state transits to *waiting*, waiting time will be incremented. We set a threshold so that the DMM state transitions to *terminate* when it has been waiting for more than T_{wait} frames (event e_6). The elements in input alphabet Σ are defined in Table 6.1.

Table 6.1: DMM input alphabet, ξ and v are estimated similarity and value metrics.

Σ	**Description**
e_1	$\xi \leq \xi_0$ and $v \leq v(\varepsilon_1)$ and $\lvert B_{wait}\rvert < T_{wait}$ and $\lvert B_{maj}\rvert < T_{maj}$
e_2	$\xi \leq \xi_0$ and $v > v(\varepsilon_1)$ and $\lvert B_{wait}\rvert < T_{wait}$ and $\lvert B_{maj}\rvert < T_{maj}$
e_3	$\xi > \xi_0$ and $v \leq v(\varepsilon_1)$ and $\lvert B_{wait}\rvert < T_{wait}$ and $\lvert B_{maj}\rvert < T_{maj}$
e_4	$\xi > \xi_0$ and $v > v(\varepsilon_1)$ and $\lvert B_{wait}\rvert < T_{wait}$ and $\lvert B_{maj}\rvert < T_{maj}$
e_5	$\lvert B_{maj}\rvert \geq T_{maj}$
e_6	$\lvert B_{wait}\rvert \geq T_{wait}$

6.4.3 Output alphabet

The output alphabet corresponds to buffer operations or actions given the current state and the input. Buffer operations include writing data to a buffer, writing data from one buffer to another, and emptying a buffer (see Table 6.2). Typically a buffer will be emptied when its data is written to another buffer.

In the proposed DMM, a_1 to a_5 are outputs assigned during the buffer tracking process; these actions simply write to a buffer or empty a buffer. When the DMM terminates, either a_6 or a_7 is applied (Fig. 6.7). If the DMM transfers from the buffering state to the terminate state, a_6 executes. The last L frames of B_{maj} are used as B_{pre} for the next buffer tracking cycle to provide contextual information. The previous frames are sent to LBO for data compression decision making. If the DMM

Table 6.2: DMM output alphabet

S	Σ	Λ	Description
s_1	e_2/e_4	a_1	Write from B_{pre} to B_{maj}, empty B_{pre}
	e_1/e_3	a_2	Write from B_{pre} to B_{wait}, empty B_{pre}
s_2	e_1	a_5	Write new frame to B_{wait}.
	$e_2/e_3/e_4$	a_3	Write new frame to B_{maj}
	e_5	a_6	Write last L frames of B_{maj} to B_{pre}, the rest of B_{maj} is sent to LBO for long-term storage.
s_3	e_1/e_3	a_5	Write new frame to B_{wait}.
	e_2/e_4	a_4	Write frames of B_{wait} and the new frame to B_{maj}, empty B_{wait}
	e_6	a_7	Write last L frames of B_{wait} to B_{pre}, and the rest of B_{wait} to B_{maj}. B_{maj} is sent to LBO for long-term storage. Empty B_{wait}.

transfers from the waiting state to the terminate state, a_7 executes. B_{wait} is then divided into two partitions; the first (earliest) partition is appended to B_{maj} while the most recent (latest) partition is used as B_{pre} for the next DMM execution cycle.

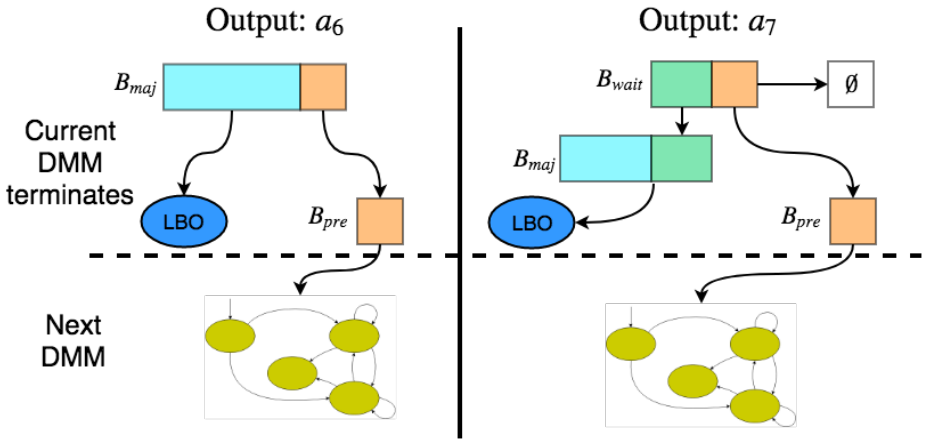


Figure 6.7: Buffer operation after DMM terminates.

It is possible for B_{maj} to overflow when populating it from B_{wait}, so the size of B_{pre} is computed by:

$$|B_{pre}| = \max(L, |B_{maj}| + |B_{wait}| - T_{maj}) \tag{6.21}$$

where L is a user-specified minimum size of B_{pre}.

6.5 Local Buffer Optimization and Long-term Storage Management

This section specifies the LBO problem and proposes a method of decoupling LBO to facilitate real-time execution. LBO determines optimal compression quality of each frame in a buffer. A long-term storage management strategy is then introduced to deal with finite storage limits.

6.5.1 LBO Formulation

LBO is applied to each data buffer obtained from the DMM to determine the optimal compression quality for each frame of the buffer. This paper formulates LBO as a nonlinear programming (NLP) problem over design vector $D = [d_1, ..., d_{|B_k|}]$. The objective function is based on three metrics: 1) Minimize total data storage cost; 2) Maximize total data value; 3) Maximize data recording decision continuity. These metrics are defined below.

Storage cost (size). For data buffer B_k, the total storage cost given decision vector D is computed as

$$C_{B_k}(D) = \sum_{i=1}^{|B_k|} c_i \phi(d_i) \leq \sum_{i=1}^{|B_k|} c_i \tag{6.22}$$

where c_i is the storage cost of the ith frame in B_k, $\phi(d_i)$ is the mapping from compression quality to compression ratio, also called the quality-ratio curve. $\phi(d_i)$ monotonically increases over $d_i \in [0, 1]$. The form of $\phi(d_i)$ depends on the compression algorithm and the data type. We use $\phi(\cdot)$ of the JPEG compressor in (6.23),

$$\phi(d) = -a_1 \log_2(1 - a_2\, d) + a_3 \tag{6.23}$$

where $[a_1, a_2, a_3]$ is the parameter vector fit by compressing real-world driving videos. Readers are guided to [116] for further details.

Data value term. The total data value of B_k given D is computed in (6.24):

$$V_{B_k}(D) = \sum_{i=1}^{|B_k|} v_i d_i \leq \sum_{i=1}^{|B_k|} v_i \tag{6.24}$$

Data value is presumed proportional to data compression quality in this work.

Decision continuity term. The decision continuity metric discourages abrupt changes in data frame decision value and is computed as the total change over all adjacent decisions in D.

$$W_{B_k}(D) = \sum_{i=2}^{|B_k|} (d_i - d_{i-1})^2 \tag{6.25}$$

This continuity term encourages storage of low-value frames when they are in proximity to high-value frames. Coupling is introduced between any two consecutive decisions to smooth the compression quality curve.

Objective function. Based on Eqs. (6.22)-(6.25) we define objective function:

$$\begin{aligned} \min_{D} \quad & \eta\, C_{B_k}(D) - \zeta\, V_{B_k}(D) + W_{B_k}(D) \\ & \text{subject to } d_i \in [0,1], \quad \forall\, d_i \in D \end{aligned} \tag{6.26}$$

Above, $\eta, \zeta \geq 0$ are weighting parameters that can be varied to examine solution sensitivity or represent user preferences. The optimization problem (6.26) may not be easy to solve because the dimension $|B_k|$ is typically large. In what follows we introduce a simple but effective value filtering method so that the continuity term $W_{B_k}(D)$ can be dropped. This results in a decoupled LBO problem where a unique minimizer exists and can be analytically solved.

6.5.2 Decoupled LBO

Estimating data value over sequential frames generates discrete-time value sequence $\{v_1, v_2, ..., v_t, ...\}$. This value sequence can have impulsive and step behaviors due to transient data events. The formulated LBO solves this problem by encouraging decision continuity $W_{B_k}(D)$ in (6.25). However, this term introduces coupling to LBO which results in a high-dimensional NLP. As an alternative, we apply a data value filtering pre-processing step which suggests the $W_{B_t}(D)$ in (6.26) can be eliminated. Data filtering is based on: 1) Assigning contextual frames of a high-value event high data value; 2) Preserving (not filtering out) impulsive events or short-term durational events of high value. The sequential data frame value signal is therefore filtered by Gaussian functions as shown in (6.27).

$$\hat{v}_t = \begin{cases} \max(v_t, v_{T_0} e^{-(\frac{t-T_0}{\sigma_f})^2}) & \text{if } t < T_0 \\ \max(v_t, v_{T_1} e^{-(\frac{t-T_1}{\sigma_f})^2}) & \text{if } t > T_1 \\ v_t & \text{otherwise} \end{cases} \tag{6.27}$$

where T_0 and T_1 are event start and end time, respectively, and σ_f is data value deviation that controls Gaussian curve width.

The proposed data value filtering scheme can serve to decouple consideration of data continuity from LBO computation. We can drop the continuity term in (6.26) to decouple the overall solution into a series of one-dimensional frame optimization problems per (6.28). With this strategy, LBO can optimize the data value specification for each frame independent of all other frames. Given constant $\frac{\zeta}{\eta}$, each LBO decision is determined by a data frame's value and size.

$$\begin{aligned} &\min_{d_i} c_i \phi(d_i) - \frac{\zeta}{\eta} \hat{v}_i \, d_i, \\ &\text{subject to } d_i \in [0, 1]. \end{aligned} \tag{6.28}$$

It has been shown in [116] that the decoupled LBO performs similar to coupled LBO with much less computation time.

6.5.3 Prioritized Data Recording in Long-term Storage

Once SBB storage limit M is reached and an optimal decision vector for a new buffer is computed, either the old buffer(s) or the new buffer must be discarded. We propose storing buffers over a long-term as a priority queue (heap) so that those with lower values are discarded preferentially. The heap is constructed based on buffer value, and the lower-value buffers are discarded until heap size is less than M. Storage limit M can be set to a value smaller than maximum available physical storage to assure data buffers can be successfully captured.

The total value of the kth buffer is calculated as:

$$V^*_{B_k} = (1 + \lambda)^k \max_i (v_i \cdot d_i) \tag{6.29}$$

where v_i is the ith data frame value, and $1 + \lambda$ with $0 < \lambda \ll 1$ is an aging factor that amplifies the value of the newest data buffer.

Each recorded buffer $\hat{B}_k$ contains the compressed data for all $f_t \in B_k$. The buffer

storage cost is C_{B_k} as in (6.22) and the buffer value is $V^*_{B_k}$. When a buffer is to be removed, data included in the buffer and their indices can be rapidly located and pruned; note that pruned indices will not be reused in other buffers. A binary $min - heap$ queue [154] is constructed to store buffers based on $V^*_{B_k}$. Algorithm 2 describes the prioritization sequence.

Algorithm 2: Priority Queue Logic

Input : New buffer B_k, heap HQ, heap size C_{HQ}, maximum available storage M.

Output: Updated heap HQ

HQ.push(B_k)
$C_{HQ} = C_{HQ} + C_{B_k}$
while $C_{HQ} > M$ **do**
    $\widehat{B}$ = HQ.pop() `// pop the buffer with the smallest` $V_{\widehat{B}}$
    $C_{HQ} = C_{HQ} - C_{\widehat{B}}$
end

6.6 Experiments in TORCS Simulation

This section presents a case study using the pre-defined EOIs from Section 6.3 and long-term traffic data generated from a simulator. Two case studies are presented analyzing coupled and decoupled LBO parameter selection, respectively. A comparison between prioritized data recording and FIFO recording with different storage limitations is also provided. Last, we examine the reproducibility of deep learning results on images compressed by the SBB to demonstrate its utility.

6.6.1 Simulation Environment

A simulator is used to generate three-lane highway traffic trajectories for this case study. The simulator is developed based on a game theoretic traffic model and is capable of representing heterogeneous and interactive multi-vehicle traffic scenarios. More details of the simulator can be found in [147]. We feed data generated from the simulator into our SBB. This simulator is utilized because it covers a large range of traffic scenarios over a short period of running time. Note that the proposed approach can be applied to other datasets also. In this case study, we define one host car and 15 participant cars so that EOIs are not too frequent or rare. The frame rate is $10Hz$, and the trajectory length is 600 seconds.

Training data. We generated $10,000$ Monte Carlo trajectories with randomly initialized car locations and velocities to estimate the data value metrics. Each trajectory terminates when the set length is reached or when the host car crashes with a participant car. Driving data of the host car and all participant cars are collected. The accumulated driving time of all cars is about $15,041$ hours and the total mileage is about 0.68 million kilometers.

From the 10,000 MC trajectories, there are $29,953,405$ cut-in events captured among all cars as well as $23,000,206$ hard braking events and $1,024,611$ conflicts. Since the simulation is restarted if one host car crash is detected, we compute probability of crash using only host car crashes; $4,799$ crashes are obtained. Likelihood of events are computed from (6.30) and listed in Table 6.3. For crash probability, the denominator is the number of frames for the host car only.

$$\Pr(\varepsilon_j) = \frac{\#\text{ of frames with } \varepsilon_j \text{ detected}}{\text{total } \#\text{ of MC trajectory frames}} \tag{6.30}$$

Table 6.3: Probability and estimated value metrics of normal frames and EOIs. $cutin_1$ and $cutin_2$ have ranges $R = 100\,m$ and $30\,m$, respectively.

Events	$normal$	$cutin_1$	$cutin_2$	$hardbraking$	$conflict$
Prob.	0.92	0.045	0.010	0.035	0.0015
Value	0.009	0.34	0.53	0.37	0.72

Test data. We simulated a single long-term test trajectory to evaluate SBB performance. This long-term trajectory consisted of $115,615$ frames (around 3 hours 12 minutes) and terminated with a crash. Statistics on test data are shown in the first and second rows of Table 6.4. We evaluated the SBB with coupled and decoupled LBO over this trajectory, then compared prioritized data recording with a conventional FIFO queuing model. We also presented the reproducibility of object detection and semantic segmentation results on SBB compressed data.

Metrics. We define three metrics, average value per frame ($aVPF$), average memory per frame ($aMPF$) and value per memory (VPM), to guide parameter selection for LBO:

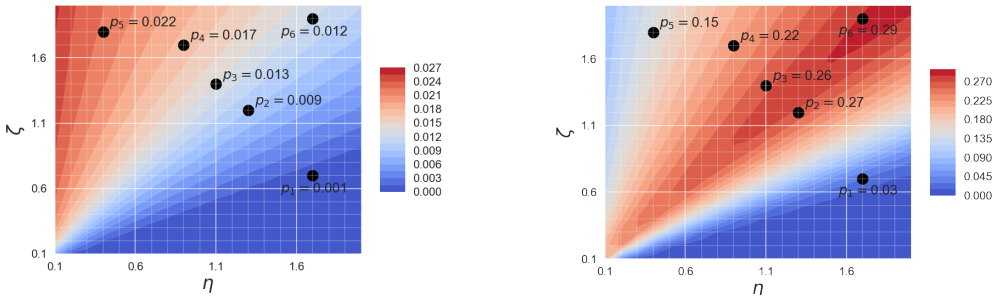
$$aVPF = \frac{\sum_i \hat{v}_i \, d_i}{N} \tag{6.31a}$$

$$aMPF = \frac{\sum_i c_i \, \phi(d_i)}{N} \tag{6.31b}$$

$$VPM = \frac{aVPF}{aMPF} \tag{6.31c}$$

6.6.2 Case Study 1: Coupled LBO and Parameter Selection

We first applied the proposed SBB with coupled LBO to the testing data. The sensitivity of coupled LBO to weighting parameter η and ζ is investigated. Figs. 6.8(a) to 6.8(b) show the contours of $aVPF$, $aMPF$ and VPM with $\eta \in \{0.1, 0.2, ..., 2.0\}$ and $\zeta \in \{0.1, 0.2, ..., 2.0\}$. Other parameters include: $T_{maj} = 600$, $T_{wait} = 30$, $L = 20$, $\sigma_f = 10$, $\xi_0 = 0.5$.



(a) Average value per frame ($aVPF$) (b) Value per memory (VPM)

Figure 6.8: Sensitivity analysis of the coupled LBO method.

Generally, the SBB performance is best when selecting parameters that result in high VPM value. However, the VPM can be large as long as the $aMPF$ is small enough, in which case all data are highly compressed. Therefore, users might also have a minimum expectation regarding $aMPF$ (or $aVPF$) of recorded SBB data. Thus, instead of simply selecting (η, ζ) corresponding to the highest VPM, the trade-off between VPM and $aVPF$ must be considered. Six example parameter selections (p_1 to p_6) are shown in Fig. 6.8 and their corresponding $aVPF$ and VPM are presented. It can be seen that at point P_6 there is a high $VPM = 0.29$ but a low $aVPF = 0.012$. Such a selection of (η, ζ) resultrs in high recording efficiency but the data can be too compressed to provide sufficient information. We select $\eta = 0.9$ and $\zeta = 1.7$ for the following experiment since it results in relatively high VPM and $aVPF$.

Computational time required to solve the coupled LBO depends on data buffer

length and frame values. The buffer length obtained by the DMM is from 3 seconds to 60 seconds with average 7.5 seconds, while the LBO solving time varies from 0.003 seconds to 24.6 seconds with average 0.9 seconds. In conclusion, solving a coupled LBO is time-consuming with a large buffer containing highly variable data values, motivating application of decoupled LBO in a deployed SBB. In this study we ran sequential least squares programming (SLSQP) solver provided by a Scipy (Python) optimization package on a machine with 16GB RAM and an Intel Xeon(R) CPU E3-1240 v5 @ 3.50GHZ*8.

6.6.3 Case Study 2: Decoupled LBO and Parameter Selection

The continuity term in the decoupled LBO is dropped, resulting in (6.32). Parameters $[a_1, a_2, a_3]$ are obtained from quality-ratio curve fitting in [116].

$$\begin{aligned} F(d_i) &= \phi(d_i) - \frac{\zeta}{\eta}\,\hat{v}_i\,d_i \\ &= -a_1 \log_2(1 - a_2 d_i) + a_3 - \frac{\zeta}{\eta}\,\hat{v}\,d_i \end{aligned} \tag{6.32}$$

The resulting function is convex and derivable in $d_i \in [0, 1]$. Therefore a unique minimum solution can be analytically computed by finding the zero-derivative solution. The optimal decision is then given by:

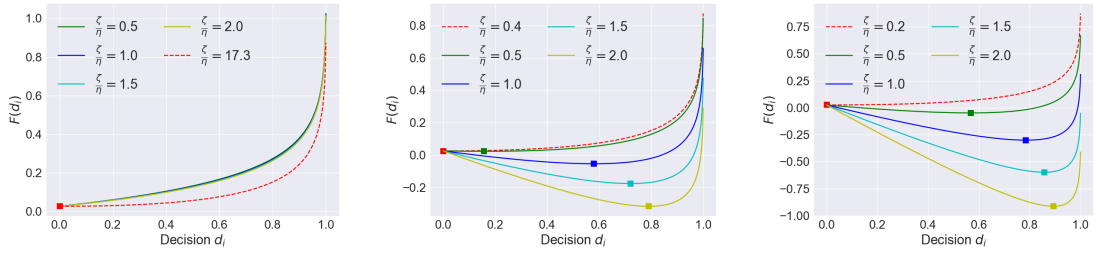
$$d_i^* = \max(0,\, -\frac{a_1}{\log_2 2\hat{v}_i}(\frac{\zeta}{\eta})^{-1} + \frac{1}{a_2}) \tag{6.33}$$

Fig. 6.9 shows the objective functions of four constant-value events with different $\frac{\zeta}{\eta}$. The *cutin* event is not shown since its value is a function of observed *cutin* range. This ratio must be selected in an interval such that all EOIs can be recorded with a high quality while *normal* frames are compressed with low quality.

Given the fitted parameters, if

$$\frac{\mathrm{d}F(d_i)}{\mathrm{d}d_i}\Big|_{d_i=0} \geq 0 \tag{6.34}$$

then $F(d_i)$ is monotonically increasing and the minimum is $d_i^* = 0$. We define boundary parameter $\frac{\zeta}{\eta} = \frac{a_1 a_2}{(\log 2\hat{v}_i)}$ by solving the equality condition in (6.34). There is one boundary parameter for each EOI. Corresponding objective functions are shown by red dotted curves in Fig. 6.9. Parameter selection must have $\frac{\zeta}{\eta} > \frac{a_1 a_2}{(\log 2\hat{v})}$ for all EOIs.



(a) normal (b) hard braking (c) conflict

Figure 6.9: Objective function of decoupled LBO for each event. Square blocks indicate optimal solutions. Each red dotted curve indicates the boundary value of parameter for such an event.

In this case study, to guarantee that the *hardbraking* event is recorded at high quality, we select $\frac{\zeta}{\eta} > 0.4$. If $\frac{\zeta}{\eta} < 0.2$, none of these EOIs are recorded. To avoid recording normal data with high quality, $\frac{\zeta}{\eta} < 17.3$ should be enforced.

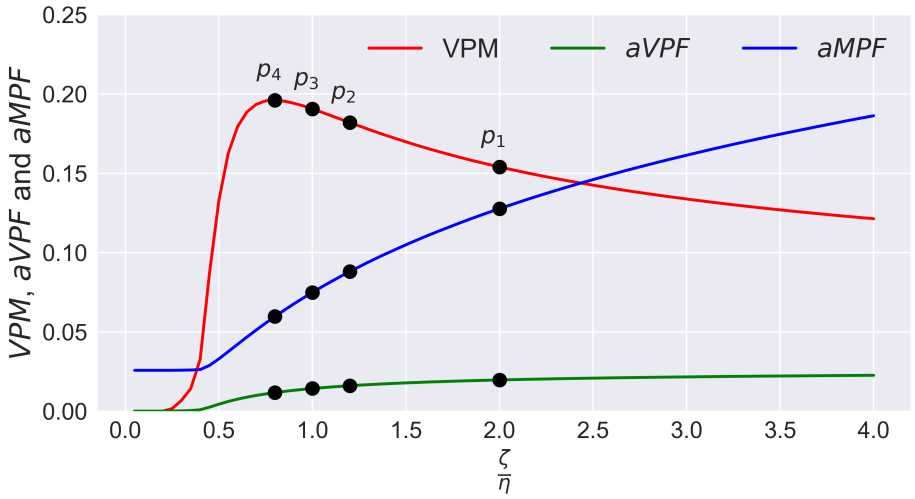


Figure 6.10: Value per frame with different weighting parameter ratios for the decoupled LBO method. Black points are examples of different parameter selections, similar to Fig. 6.8.

In Fig. 6.10, VPM decreases significantly when $\frac{\zeta}{\eta} > 0.7$. However, $aVPF$ and $aMPF$ values are extremely small when $\frac{\zeta}{\eta} < 0.7$, indicating that most data are highly compressed. This is consistent with observations from the coupled LBO parameter selection study. Therefore we recommend $\frac{\zeta}{\eta} > 0.7$ but not too large to realize a reasonable trade-off between VPM and $aVPF$.

6.6.4 Results and Analysis

Below we present SBB results using the coupled LBO method with parameters selected per Section 6.6.2.

6.6.4.1 SBB data recording statistics

Statistics of SBB data recording on the test data are presented in Table 6.4. The third and fourth rows show the average and standard deviation of SBB compression decision (quality) of each event. It can be seen that the qualities on EOIs are much higher and more stable compared to the quality of *normal* frames, indicating that the SBB places high emphasis on all EOI frames but treats *normal* frames differently based on how close each frame is to an EOI. Generally, the SBB maintains high quality for EOIs while compressing normal data frames to save memory.

Table 6.4: Raw test data and SBB data compression statistics.

		normal	*cutin*	*hard braking*	*conflict*
Raw data	**# of frames**	106230	2955	5865	541
	Size (MB)	19639.16	547.18	1082.15	101.45
SBB data	**Avg.** d_i	0.44	0.76	0.79	0.88
	Std. d_i	0.36	0.019	0.012	0.005
	Size (MB)	1511.30	93.02	151.63	22.70

The last row of Table 6.4 summarizes SBB memory requirements for each event type to support comparison of raw data and SBB storage requirements over normal and EOI datasets. Most of the recorded driving data frames are *normal* since the SBB records all frames unless finite storage is reached and *normal* frames are dominant in the test data set. Some *normal* frames are also contextual frames of EOIs which have higher value according to the filter and therefore are compressed with high quality. Table 6.5 indicates the percentage (pct.), quantity, and storage cost of contextual frames as a fraction of all normal frames acquired from simulations. The contextual frames here are defined as frames within a specified range (in number of frames) from an EOI.

For example, we found that 20.27% of the *normal* frames are located within ± 5 frames of an EOI, and their storage cost is about 43.29% of the storage cost of all *normal* frames. This indicates that the memory is efficiently utilized to record

Table 6.5: Statistics on recorded normal frames. The context range is in number of frames with frame rate $10Hz$.

Context range	± 5	± 10	± 15	± 20
Quantity pct.	20.27%	34.99%	46.19%	54.80%
Storage cost pct.	43.29%	66.87%	79.46%	83.09%

contextual frames. 83.09% of the storage cost of *normal* frames are within ± 20 frames of an EOI.

6.6.4.2 Prioritized data recording with long-term storage limits

We apply different storage limitations (M) to the SBB. Recorded event counts are summarized in Table 6.6. With $M = 1500\,MB$, more *cutin* (~ 200), *hardbraking*(~ 500), and $conflict(\sim 90)$ events are missed by the FIFO model compare to the prioritized model, while the SBB reserves more storage for EOIs by discarding more *normal* frames. With $M = 500\,MB$, the SBB model saves 40% of the *cutin* events and all $conflict$ events, while the FIFO model records $10,072$ more *normal* frames and missed 79% of the $conflict$ frames. Note that less *hardbraking* frames are recorded by the SBB with $M = 500\,MB$ because their values are determined to be lower than for some *cutin*s. These differences show the prioritized data recording scheme is able to record valuable data happening in the early phase of the trajectory that would be discarded by a conventional FIFO (circular buffer) data recorder.

Table 6.6: Prioritized (SBB) and FIFO data recording comparison with storage limit M. Both schemes use LBO to guide data buffer JPEG compression.

M		*normal*	*cutin*	*hardbraking*	$conflict$
$1500MB$	**Ours**	74271	2725	5547	541
	FIFO	90953	2530	5036	456
$500MB$	**Ours**	21135	1182	1599	541
	FIFO	31207	738	1719	113

6.6.5 Performance of SBB Data on Deep Learning Model

To evaluate SBB compression with respect to AV percept reproducibility, we applied object detection and semantic segmentation models on data recorded with SBB

compression and two comparative baselines. For object detection, we apply Mask-RCNN [3] pre-trained on the Common Objects in Context (COCO) dataset. Bounding box average precision AP^{bb} [3] is computed as the evaluation metric. For semantic segmentation tasks, we use DeepLabV2 [4] pre-trained on the CrowdFlower dataset. We evaluate the performance using pixel intersection over union (IOU) as shown in (6.35).

$$PIOU = \frac{\sum_{i=0,j=0}^{W,H} \mathbb{1}_{(\hat{p}_{i,j}, p_{i,j})}}{W \cdot H} \tag{6.35}$$

where $\hat{p}_{i,j}$ and $p_{i,j}$ are predicted and ground truth classes of pixel (i, j), respectively, and $\mathbb{1}_{i,j}$ is an indicator function returning 1 if $\hat{p}_{i,j} = p_{i,j}$.

The **SBB** compresses images using the method from Section 6.6.4 resulting in 1778.91 MB of stored data; **Baseline1** compresses all images with 0.1 quality resulting in 1433.6 MB of data; **Baseline2** compresses all images with 0.5 quality, resulting in 2355.2 MB of data.

Image data is obtained from the TORCS simulator as described above. The simulator [147] used to generate traffic data has been integrated with TORCS to assure experimental data is consistent [155]. TORCS-generated images are referred to as "raw data" in this study and SBB compressed images are "SBB data". Fig. 6.11 shows an example result with different compression qualities, consistent with Fig. 6.1.

Figure 6.11: Object detection (left) and semantic segmentation (right) on a TORCS image compressed with 1, 0.5, 0.1 and 0.01 quality (from top to bottom).

The metric values of Mask-RCNN and DeeplabV2 on different data are presented in Table 6.7. We assume object detection and semantic segmentation results on raw data are ground truth and report the metrics with SBB and two baseline datasets. We separately show the results on *normal* frames and frames contain three EOIs and

ignore crash frames since no compression is applied with SBB. Both Mask-RCNN and DeeplabV2 achieve better performance on EOI frames in SBB data compared to baseline data, indicating that the EOIs saved by the SBB is more reproducible in these two specific tasks. The performance on *normal* images recorded by the SBB is worse than on the images from baselines, consistent with the SBB design goal to highly compress normal data to save memory for EOIs. The SBB is able to record more reproducible but less memory-consuming data compared to compressing data with a preset constant compression ratio.

Another interesting observation is that DeeplabV2 performance is more robust to JPEG compression compared to Mask-RCNN. One possible reason is that with JPEG compression, the object feature used for detection is destroyed while the whole image feature is better maintained, which makes it harder to distinguish an object from background.

Table 6.7: Mask-RCNN AP^{bb} and DeeplabV2 $PIOU$ on two baseline datasets and SBB data. Greater numbers indicate better performance.

	Data	*normal*	*cutin*	*hard braking*	*conflict*
Mask RCNN	Baseline1	0.32	0.32	0.38	0.28
	Baseline2	0.68	0.68	0.74	0.67
	SBB	0.47	0.74	0.80	0.80
DeeplabV2	Baseline1	0.78	0.80	0.79	0.79
	Baseline2	0.88	0.90	0.90	0.91
	SBB	0.58	0.92	0.92	0.95

6.7 Experiments on BDD100K+DoTA Datasets

We conducted SBB data collection experiments on a large-scale real-world video dataset and present results in this section. We summarize storage requirements of SBB compressed data to showcase its preservation of anomalous data. We then compare our SBB prioritized data recording with a first-in-first-out (FIFO) queue data recording strategy [117]. Finally, we compare the performance of two different value estimates: anomaly score value and IBCC combined value.

6.7.1 BDD100K+DoTA Datasets

The SBB is designed for high-bandwidth data collection in long-term driving where the onboard storage is limited. Therefore, SBB performance evaluation requires a dataset that contains a large quantity of high-quality video data with embedded events of interest (EOIs). To our best knowledge, there is no single dataset that satisfies all these requirements. The BDD100K dataset [6] is one of the largest high-quality driving video datasets with $100,000$ video clips, equal to $\sim 1,100$ driving hours. The DoTA dataset [1] is the largest and newest high-quality video dataset for traffic anomalies with $4,677$ anomalous video clips. We combined the $10,000$ validation videos in the BDD100K dataset and randomly interspersed 500 anomalous video clips from the DoTA dataset, resulting in a large-scale testing video with $\sim 4,000,000$ frames at 10 FPS. By combining these two datasets, we obtained a $>$ 100-hour high-quality driving video with the vast majority ($\sim 99.5\%$) of the frames as normal but still with a large number of EOIs the SBB might recognize and record.

6.7.2 Results

SBB Data Compression. SBB data compression statistics with no memory limit are presented in Table 6.8. It can be seen that the storage cost of normal frames is significantly reduced (703.84 GB to 127.92 GB, 82%) by the SBB. This leads to a 108% increase in the ratio of anomalous data storage to normal data storage. Both the average (**avg.**) and median (**med.**) compression factor decisions of the SBB are higher for the anomalous frames, indicating that the SBB is able to identify and preserve anomalous frames over normal ones. Figure 6.12 displays normal frames which were highly compressed by the SBB along with preserved anomalous frames. Figure 6.13 shows two failure cases where anomalous frames were mistakenly compressed. Both of these failures showcase a lack of robustness against cases where anomalous objects are occluded.

The decision difference between normal and anomaly frames with these real-world datasets is not as significant compared to the simulation experiment in Section 6.6 due to the fact that the EOI detection in simulation was 100% accurate while VAD on real-world data is far from perfect per Section 5.7.1. Moreover, the median decision for a normal frame is significantly lower than the mean, indicating that there are outlier normal frames with unusually high value scores. The standard deviation (**std.**) of anomalous frames is significantly larger than that in simulation experiments (0.36 vs ~ 0.02), showing how inaccurate VAD and OAD reduces SBB efficiency on real-world

Table 6.8: Raw and SBB compressed data statistics on the BDD100K+DoTA dataset.

		Normal	Anomaly	Anomaly Ratio
Raw Data	# of frames	3,967,977	16,768	
	size (GB)	703.84	1.76	0.25%
SBB w/ VAD+OAD	size (GB)	127.92	0.67	0.52%
	avg. d_i	0.51	0.58	
	med. d_i	0.55	0.65	
	std. d_i	0.24	0.26	
SBB w/ ground truth labels	size (GB)	18.04	1.19	6.60%
	avg. d_i	0.00	0.92	
	med. d_i	0.00	0.92	
	std. d_i	0.03	0.00	

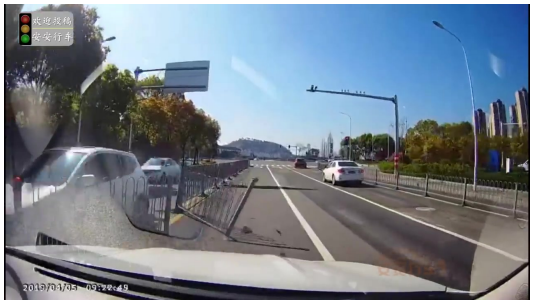
data. The limitations of VAD and OAD are further shown by evaluating performance of the SBB given ground-truth labels as VAD and OAD scores. The anomalous-to-normal storage ratio increases by 2640%, driven by the substantial differences in decisions between normal and anomalous frames. This upper-bound performance of the SBB indicates that as anomaly detection techniques continue to improve, the performance of the SBB will improve as well.

Table 6.9: Comparison of Prioritized Recording and FIFO.

M		Normal	Anomaly
25 GB	Priority	583,552	5,365
	FIFO	804,387	3,568
12.5 GB	Priority	308,459	3,739
	FIFO	427,466	1,903

Priority Queue vs. FIFO. Table 6.9 compares frames recorded with the SBB prioritized recording system against frames recorded with a FIFO queue at memory limits of $M = 12.5$ GB and 25 GB. These values represent a non-trivial amount of data to upload assuming only sporadic internet access is available. In both experimental scenarios, prioritized recording saved fewer normal frames and more anomalous frames than with the FIFO strategy. Notably, the prioritized recording using 12.5 GB saved more anomalous frames than the FIFO queue at 25 GB. The prioritization strategy of the SBB removes $\sim 93\%$ of the normal frames while still recording $\sim 20\%$ anomalous frames at $M = 12.5$ GB. Compared with the FIFO queue, the SBB saves $\sim 25\%$ fewer normal frames and $\sim$ 50-100% more anomalous frames.

(a) A normal driving frame



(b) A vehicle-object collision (VO) event

(c) Precursor of the TC event

(d) An ego turning collision (TC) event

Figure 6.12: Compressed normal frames (left) and preserved anomalies (right).

(a) A non-ego out-of-control (OO*) event where a windshield wiper is partially blocking the anomaly.

(b) A non-ego road crossing collision (TC*) where one vehicle is partially occluded.

Figure 6.13: Compressed anomaly failure cases.

Value Estimation Method Comparison. Table 6.10 compares decision statistics for hybrid value estimation and IBCC value estimation. We note that the VAD-only method generates the largest decision difference in normal and anomalous frames. We suspect this is a result of OAD's inability to consistently differentiate between anomalous and normal frames per Section 5.7.3. IBCC-based value estimation results in a relatively low difference in decisions. However, IBCC does lead to lower standard deviation in the decision indicating that it reduces the outlying anomaly scores which results in more stable decisions. This is because IBCC makes use of prior distributions shared between frames for each hidden variable to establish a base expectation for the anomaly status. On the other hand, the hybrid method takes only the current observations into account, meaning each frame is considered completely independent of every other frame. Low decision standard deviation is especially valuable when memory is limited. With high standard deviation, many normal frames will be assigned high priority, while many anomalous frames will be assigned low priority. Thus, when memory capacity is reached, anomalous data mistakenly given low priority may be discarded.

For applications which value general EOIs, VAD-only value estimation ($\alpha = 1$, $\beta = 0$) has the greatest ability to distinguish normal and anomalous data. However, users interested in specific EOIs may opt to use hybrid value in order to incorporate the EOI classification offered by OAD. In terms of hybrid value parameters, Table 6.10 shows that lower weights result in higher decision differences. However, in situations where retaining high data quality is critical, higher α and β values may be used to achieve higher overall decision quality. Additionally, the higher decision differences as α increases shown in Figure 6.14 indicate once again that VAD contributes more to the differentiation of normal and anomalous frames than does OAD. Finally, the low-variance decision-making of IBCC is useful in memory-limited systems when the retention of most anomalous frames at lower quality is more important than the retention of fewer anomalous frames at higher quality.

6.8 Conclusion

This chapter has presented a Smart Black Box (SBB) architecture that makes compression and storage prioritization decisions with a two-stage process. The SBB first caches raw data in a short-term buffer and determines compression factor based on data value and size. Data value is computed based on its novelty and the presence of temporally-proximal high-value data frames. Short-term buffers are managed

Table 6.10: Compression quality decisions for hybrid and IBCC value estimation.

Value Estimation	α	β		Normal	Anomaly
VAD Only	1.0	0.0	**avg.** d_i	0.27	0.39
			med. d_i	0.14	0.36
			std. d_i	0.25	0.31
OAD Only	0.0	1.0	**avg.** d_i	0.10	0.14
			med. d_i	0.0	0.07
			std. d_i	0.15	0.17
Hybrid	0.9	0.1	**avg.** d_i	0.26	0.37
			med. d_i	0.14	0.36
			std. d_i	0.29	0.33
	0.5	0.5	**avg.** d_i	0.20	0.30
			med. d_i	0.10	0.27
			std. d_i	0.24	0.28
	0.1	0.9	**avg.** d_i	0.12	0.19
			med. d_i	0.02	0.16
			std. d_i	0.16	0.19
IBCC	N/A	N/A	**avg.** d_i	0.30	0.34
			med. d_i	0.28	0.32
			std. d_i	0.09	0.13

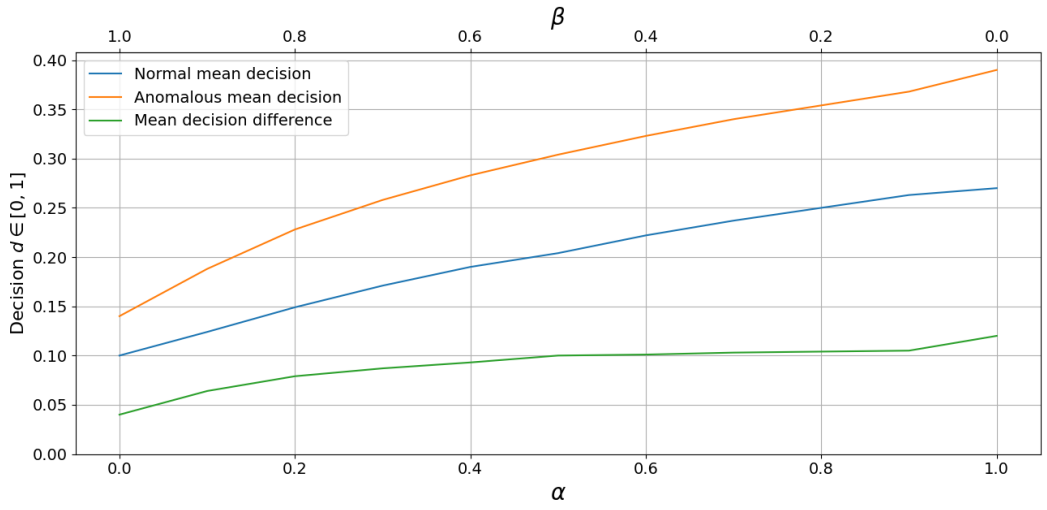


Figure 6.14: Mean normal and anomalous decisions for $\alpha + \beta = 1$.

by a deterministic Mealy machine (DMM) so that high-value data or similar data are buffered together. For long-term data collection given finite onboard storage, the SBB discards the lowest-value data regardless of age. A simulation case study generates driving trajectories and first-person view images containing four prede-

fined EOIs. We show that the local buffer optimization strategy enables the SBB to record reproducible but less memory-consuming data. Experiments on a large (> 100 hour) BBD100K+DoTA driving video shows how the SBB transfers from simulation to real-world. By combining video anomaly detection (VAD) and online action detection (OAD) scores, the SBB detects real-world events and records valuable data. Experiment results also show that SBB efficiency is limited by the accuracy of VAD and OAD algorithms.

Future work could extend the SBB in several aspects. For example, multiple clusters can be applied to record different EOIs so that new EOIs can be compared with existing clusters to avoid recording redundant data. Another potential extension is to combine static metrics with real-time (dynamic) metrics including available storage, observed driving and traffic risks, and observed frequency (novelty) of each event type.

CHAPTER VII

Conclusion and Future Work

7.1 Conclusion

This dissertation studied three problems: 1) Object trajectory and behavior prediction in driving videos; 2) Anomaly detection in driving videos based on research in 1); and 3) a Smart Black Box for driving video data collection based on research in 2). We improved state of the art (SOTA) in traffic object modeling and used these methods to improve SOTA in traffic anomaly detection. We utilized knowledge of traffic objects and anomalies to conduct efficient data collection experiments based on our understanding of traffic scenes.

We introduced the HEV-I, A3D and DoTA datasets created for future object localization (trajectory prediction) and driving video anomaly detection. A specific advantage of the DoTA dataset is that it provides not only temporal annotations for anomalous events, but also specifies event spatial locations in terms of bounding boxes and a semantic description of the anomaly category. DoTA emphasizes the importance of explainability in video anomaly detection: 1) Does the model actually look at the anomalous region on the video? and 2) Does the model understand what type of anomaly is happening in the video? These two questions are not addressed in previous research so DoTA is a strong supplement to fill this gap.

Our multi-stream network for ego-centric future object localization (FOL) incorporated optical flow and ego-motion information. FOL is one of the first efforts to address the ego-centric vehicle bounding box prediction problem. Rich optical flow information from images and ego-motion of the moving car-mounted camera is used to predict bounding boxes in a "long-term" future (*e.g.*, 1 second). Our multi-modal bi-directional trajectory predictor (BiTraP) based on goal estimation mitigates accumulated prediction error. BiTraP is based on the idea that an object's trajectory is significantly influenced by its goal position and how it is approaching that goal.

BiTraP uses a bi-directional trajectory decoder for goal estimation and completes the trajectory from start to goal and from goal to start. Without using extra information from the environment or neighbor objects, BiTraP achieves SOTA results on eight published datasets, 2 first-person view and 6 bird's-eye view. BiTraP is the first work that studies how different latent space distributions impact a multi-modal trajectory predictor, which inspires future work to carefully select/design the latent space of a multi-modal trajectory predictor.

For pedestrian behavior prediction we modeled intent and semantic action in a multi-task learning network to boost detection accuracy for each. Distinct from general video action recognition or online action detection tasks, pedestrian behavior prediction is a much less studied area with many concepts not yet carefully defined. This dissertation hypothesizes that intent describes underlying future action and models pedestrian behavior as a combination of a pedestrian's intent and action. Our multi-task network encodes observed data to predict future action which in turn improves estimates of present action and intent. Experiments with published datasets show the effectiveness of our method, especially in scenarios where pedestrian action evolves over time.

Our video anomaly detection (VAD) work extends our FOL research. Because previous frame-level VAD methods perform poorly with onboard camera data we introduced an FOL-based VAD method to focus on movable foreground objects and ignore a potentially noisy moving background. Errors in FOL caused by inaccurate object detection and tracking motivated our definition of a prediction consistency based anomaly metric. We close the gap between previous frame-level VAD methods and our object-level method by introducing an Ensemble method that combines outputs from each. Our Ensemble method achieves SOTA results on three published datasets including our A3D and DoTA datasets.

Long-term driving data collection is essential to develop and evaluate AV perception algorithms. We proposed a Smart Black Box (SBB) as an intelligent event data recorder for high-bandwidth data collection in long-term driving scenarios with limited onboard storage. Our anomaly detection methods guide the SBB to prioritize highly anomalous events over normal driving data. The SBB manages small data buffers in real-time and compresses data buffers using a compression factor optimized over expected data value and cost. To evaluate SBB performance, we designed a simulated highway traffic dataset with TORCS and real-world experiments with BDD100K and DoTA datasets. Given hundreds of GBs of video frames and limited onboard storage, the proposed SBB compresses or discards most normal frames

to save space for anomalous event data including context annotations. Collected anomaly data maintains high-quality and can be used in further research to validate and verify AV behaviors.

7.2 Future Work

Direct follow-on research and long-term future work is summarized in this section. First, object interaction has been assumed an important factor impacting each traffic participant's decisions. However, previous research using interaction modeling in trajectory prediction shows little improvement in accuracy, and results lack explainability. Future work might design explainable interaction modeling methods to improve trajectory and behavior prediction accuracy. Also, extracting useful information from environment data is a key direction worth further study. For example, relevant infrastructure features in an image, e.g., stop light illumination, impact traffic participants. Designing a model that recognizes object-environment relations is important to understand object trajectories and behaviors.

Second, existing traffic video anomaly detection is still far from being sufficiently accurate for real-world deployment. We introduced the DoTA dataset to help improve models to recognize anomalies in terms of spatial, temporal, and categorical accuracy. However, addressing all three problems in concert is still difficult, especially for unsupervised methods. Therefore, it is critical to conduct additional research to close the gap between supervised and unsupervised VAD methods to find efficient methods with high explainability in spatial, temporal and categorical domains.

Finally, the SBB design implemented in this dissertation only considers pre-defined events of interest (EOIs) during data collection. In future work data streams might be organized in multiple clusters to distinguish different EOIs, to facilitate identification of new EOIs in comparison with existing EOI clusters, and to avoid recording redundant data. SBB functionality can also be improved by combining static metrics with real-time (dynamic) metrics including available storage, observed driving and traffic risks, and observed frequency (novelty) of each event type.

Despite the significant advances recently made in trajectory prediction and anomaly detection, the AV community still faces an important an open question: What will be required for people to feel confident about perception systems that rely on machine learning, and how can these systems be safely deployed in future autonomous vehicle (AV) systems? Emerging work in human-robot systems and artificial intelligence ethics has investigated these broad questions with grounding in fields including

psychology, precedence, and policy. In the AV industry, validation & verification over long-term driving uses miles per disengagement (MPD) as a primary metric to justify the reliability of a vehicle. However, metrics like MPD do not explain the logic behind every decision made by the AV systems, not to mention logic applied by subsystems such as machine learning based perception modules. Two metrics are emerging as critical for a trust-worthy machine learning (ML) based AV system: transferability and explainability. A transferable ML method does not overfit to a preferred dataset but can be generalized to different application environments, *e.g.*, from a car to a truck and from one country to another. An explainable ML method not only predicts the correct answer but also explains the logic behind the prediction, *e.g.*, a pedestrian is not going to cross the road because there is a red light. Keeping this confidence metric in mind motivates several future work trajectories for this dissertation.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Y. Yao, X. Wang, M. Xu, Z. Pu, E. M. Atkins, and D. J. Crandall, "When, where, and what? a new dataset for anomaly detection in driving videos," *arXiv preprint arXiv:2004.03044*, 2020.

[2] W. Liu, W. Luo, D. Lian, and S. Gao, "Future frame prediction for anomaly detection–a new baseline," in *CVPR*, 2018.

[3] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *ICCV*, 2017.

[4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *arXiv:1606.00915*, 2016.

[5] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *IJRR*, 2013.

[6] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving video database with scalable annotation tooling," *arXiv preprint arXiv:1805.04687*, 2018.

[7] Y. Yao, M. Xu, C. Choi, D. J. Crandall, E. M. Atkins, and B. Dariush, "Egocentric vision-based future vehicle localization for intelligent driving assistance systems," in *ICRA*, 2019.

[8] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," *arXiv preprint arXiv:1903.11027*, 2019.

[9] S. Malla, B. Dariush, and C. Choi, "Titan: Future forecast using action priors," *arXiv preprint arXiv:2003.13886*, 2020.

[10] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *CVPR*, 2016.

[11] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *CVPR*, 2012.

[12] A. Rasouli, I. Kotseruba, T. Kunic, and J. K. Tsotsos, "Pie: A large-scale dataset and models for pedestrian intention estimation and trajectory prediction," in *ICCV*, 2019.

[13] A. Rasouli, I. Kotseruba, and J. K. Tsotsos, "Agreeing to cross: How drivers and pedestrians communicate," in *IV*, 2017.

[14] K. Mangalam, E. Adeli, K.-H. Lee, A. Gaidon, and J. C. Niebles, "Disentangling human dynamics for pedestrian locomotion forecasting with noisy supervision," in *WACV*, 2020.

[15] B. Liu, E. Adeli, Z. Cao, K.-H. Lee, A. Shenoi, A. Gaidon, and J. C. Niebles, "Spatiotemporal relationship reasoning for pedestrian intent prediction," *RAL*, 2020.

[16] R. Ervin, J. Sayer, D. LeBlanc, S. Bogard, M. Mefford, M. Hagan, Z. Bareket, and C. Winkler, "Automotive collision avoidance system field operational test report: methodology and results," NHTSA, Tech. Rep., 2005.

[17] H. C. Gabler, C. E. Hampton, and J. Hinch, "Crash severity: a comparison of event data recorder measurements with accident reconstruction estimates," SAE Technical Paper, Tech. Rep., 2004.

[18] T. van der Heiden, N. S. Nagaraja, C. Weiss, and E. Gavves, "Safecritic: Collision-aware trajectory prediction," *arXiv preprint arXiv:1910.06673*, 2019.

[19] F.-H. Chan, Y.-T. Chen, Y. Xiang, and M. Sun, "Anticipating accidents in dashcam videos," in *ACCV*, 2016.

[20] T. Suzuki, H. Kataoka, Y. Aoki, and Y. Satoh, "Anticipating traffic accidents with adaptive loss and large-scale incident db," in *CVPR*, 2018.

[21] R. Herzig, E. Levi, H. Xu, H. Gao, E. Brosh, X. Wang, A. Globerson, and T. Darrell, "Spatio-temporal action graph networks," in *CVPRW*, 2019.

[22] D. Zhao, H. Lam, H. Peng, S. Bao, D. J. LeBlanc, K. Nobukawa, and C. S. Pan, "Accelerated evaluation of automated vehicles safety in lane-change scenarios based on importance sampling techniques," *IEEE Trans. on Intel. Transp. Sys.*, vol. 18, no. 3, pp. 595–607, 2017.

[23] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transportation Research Part A: Policy and Practice*, 2016.

[24] T. Coughlin, "The memory of cars," *Forbes*, July 20th, 2016. [Online]. Available: https://www.forbes.com/sites/tomcoughlin/2016/07/20/the-memory-of-cars/#3170f5c41b33

[25] W. Liu, W. Luo, Z. Li, P. Zhao, and S. Gao, "Margin learning embedded prediction for video anomaly detection with a few anomalies," in *IJCAI*, 2019.

[26] A. Rasouli, I. Kotseruba, and J. K. Tsotsos, "Understanding pedestrian behavior in complex traffic scenes," *IEEE Transactions on Intelligent Vehicles*, 2017.

[27] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *ICCV*, 2009.

[28] L. Leal-Taixé, M. Fenzi, A. Kuznetsova, B. Rosenhahn, and S. Savarese, "Learning an image-based motion context for multiple people tracking," in *CVPR*, 2014.

[29] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[30] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2.

[31] N. Lee and K. M. Kitani, "Predicting wide receiver trajectories in american football," in *WACV*, 2016.

[32] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *CVPR*, 2016.

[33] J. Liang, L. Jiang, J. C. Niebles, A. G. Hauptmann, and L. Fei-Fei, "Peeking into the future: Predicting future person activities and locations in videos," in *CVPR*, 2019.

[34] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction," in *CVPRW*, 2018.

[35] A. Sadeghian, F. Legros, M. Voisin, R. Vesel, A. Alahi, and S. Savarese, "CarNet: Clairvoyant attentive recurrent network," in *ECCV*, 2018.

[36] C. Choi and B. Dariush, "Looking to relations for future trajectory forecast," in *ICCV*, 2019, pp. 921–930.

[37] Y. Li, Z. Ye, and J. M. Rehg, "Delving into egocentric actions," in *CVPR*, 2015.

[38] M. Ma, H. Fan, and K. M. Kitani, "Going deeper into first-person activity recognition," in *CVPR*, 2016.

[39] G. Bertasius, A. Chan, and J. Shi, "Egocentric basketball motion planning from a single first-person image," in *CVPR*, 2018.

[40] C. Fan, J. Lee, M. Xu, K. K. Singh, Y. J. Lee, D. J. Crandall, and M. S. Ryoo, "Identifying first-person camera wearers in third-person videos," *arXiv:1704.06340*, 2017.

[41] M. Xu, C. Fan, Y. Wang, M. S. Ryoo, and D. J. Crandall, "Joint person segmentation and identification in synchronized first-and third-person videos," *arXiv:1803.11217*, 2018.

[42] A. Bhattacharyya, M. Fritz, and B. Schiele, "Long-term on-board prediction of people in traffic scenes under uncertainty," in *CVPR*, 2018.

[43] T. Yagi, K. Mangalam, R. Yonetani, and Y. Sato, "Future person localization in first-person videos," in *CVPR*, 2018.

[44] Y. Yao, M. Xu, Y. Wang, D. J. Crandall, and E. M. Atkins, "Unsupervised traffic accident detection in first-person videos," in *IROS*, 2019.

[45] X. Du, R. Vasudevan, and M. Johnson-Roberson, "Bio-lstm: a biomechanically inspired recurrent neural network for 3-d pedestrian pose and gait prediction," *RAL*, 2019.

[46] R. Morais, V. Le, T. Tran, B. Saha, M. Mansour, and S. Venkatesh, "Learning regularity in skeleton trajectories for anomaly detection in videos," in *CVPR*, 2019.

[47] S. Sivaraman and M. M. Trivedi, "Dynamic probabilistic drivability maps for lane change and merge driver assistance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2063–2073, 2014.

[48] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Multi-agent generative trajectory forecasting with heterogeneous data for control," in *ECCV*, 2020.

[49] Y. Yao, E. Atkins, M. Johnson-Roberson, R. Vasudevan, and X. Du, "Bitrap: Bi-directional pedestrian trajectory prediction with multi-modal goal estimation," *arXiv preprint arXiv:2007.14558*, 2020.

[50] M. Hudnell, T. Price, and J.-M. Frahm, "Robust aleatoric modeling for future vehicle localization," in *CVPRW*, 2019.

[51] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially acceptable trajectories with generative adversarial networks," in *CVPR*, 2018.

[52] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, H. Rezatofighi, and S. Savarese, "Sophie: An attentive gan for predicting paths compliant to social and physical constraints," in *CVPR*, 2019.

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

[54] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, "Desire: Distant future prediction in dynamic scenes with interacting agents," in *CVPR*, 2017.

[55] N. Deo and M. M. Trivedi, "Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms," in *IV*, 2018.

[56] B. Ivanovic, E. Schmerling, K. Leung, and M. Pavone, "Generative modeling of multimodal multi-human behavior," in *IROS*, 2018.

[57] C. Choi, A. Patil, and S. Malla, "Drogon: A causal reasoning framework for future trajectory forecast," *arXiv preprint arXiv:1908.00024*, 2019.

[58] B. Ivanovic and M. Pavone, "The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs," in *ICCV*, 2019.

[59] E. Rehder and H. Kloeden, "Goal-directed pedestrian prediction," in *ICCVW*, 2015.

[60] E. Rehder, F. Wirth, M. Lauer, and C. Stiller, "Pedestrian prediction by planning using deep neural networks," in *ICRA*, 2018.

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

[62] N. Deo and M. M. Trivedi, "Trajectory forecasts in unknown environments conditioned on grid-based plans," *arXiv preprint arXiv:2001.00735*, 2020.

[63] K. Mangalam, H. Girase, S. Agarwal, K.-H. Lee, E. Adeli, J. Malik, and A. Gaidon, "It is not the journey but the destination: Endpoint conditioned trajectory prediction," *arXiv preprint arXiv:2004.02025*, 2020.

[64] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *NeurIPS*, 2014.

[65] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *ECCV*, 2016.

[66] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *ICCV*, 2015.

[67] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *CVPR*, 2017.

[68] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *CVPR*, 2018.

[69] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," in *ICCV*, 2019.

[70] R. De Geest, E. Gavves, A. Ghodrati, Z. Li, C. Snoek, and T. Tuytelaars, "Online action detection," in *ECCV*, 2016.

[71] J. Gao, Z. Yang, and R. Nevatia, "Red: Reinforced encoder-decoder networks for action anticipation," *BMVC*, 2017.

[72] Z. Shou, J. Pan, J. Chan, K. Miyazawa, H. Mansour, A. Vetro, X. Giro-i Nieto, and S.-F. Chang, "Online detection of action start in untrimmed, streaming videos," in *ECCV*, 2018.

[73] M. Xu, M. Gao, Y.-T. Chen, L. S. Davis, and D. J. Crandall, "Temporal recurrent networks for online action detection," in *ICCV*, 2019.

[74] M. Gao, M. Xu, L. S. Davis, R. Socher, and C. Xiong, "StartNet: Online detection of action start in untrimmed videos," in *ICCV*, 2019.

[75] A. Rasouli, I. Kotseruba, and J. K. Tsotsos, "Are they going to cross? a benchmark dataset and baseline for pedestrian crosswalk behavior," in *ICCVW*, 2017.

[76] S. A. Bouhsain, S. Saadatnejad, and A. Alahi, "Pedestrian intention prediction: A multi-task perspective," *arXiv preprint arXiv:2010.10270*, 2020.

[77] Z. Fang and A. M. López, "Is the pedestrian going to cross? answering by 2d pose estimation," in *IV*, 2018.

[78] F. Piccoli, R. Balakrishnan, M. J. Perez, M. Sachdeo, C. Nunez, M. Tang, K. Andreasson, K. Bjurek, R. D. Raj, E. Davidsson *et al.*, "Fussi-net: Fusion of spatio-temporal skeletons for intention prediction network," *arXiv preprint arXiv:2005.07796*, 2020.

[79] P. Wei, Y. Liu, T. Shu, N. Zheng, and S.-C. Zhu, "Where and why are they looking? jointly inferring human attention and intentions in complex tasks," in *CVPR*, 2018.

[80] D. Xie, T. Shu, S. Todorovic, and S.-C. Zhu, "Learning and inferring "dark matter" and predicting human intents and trajectories in videos," *TPAMI*, 2017.

[81] F. Schneemann and P. Heinemann, "Context-based detection of pedestrian crossing intention for autonomous driving in urban environments," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 2243–2248.

[82] H. Zhang, Y. Liu, C. Wang, R. Fu, Q. Sun, and Z. Li, "Research on a pedestrian crossing intention recognition model based on natural observation data," *Sensors*, vol. 20, no. 6, p. 1776, 2020.

[83] K. Takeda, C. Miyajima, T. Suzuki, P. Angkititrakul, K. Kurumida, Y. Kuroyanagi, H. Ishikawa, R. Terashima, T. Wakita, M. Oikawa, and Y. Komada, "Self-coaching system based on recorded driving data: Learning from one's experiences," *IEEE Trans. on Intel. Transp. Sys.*, vol. 13, no. 4, pp. 1821–1831, 2012.

[84] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly detection in automobile control network data with long short-term memory networks," in *Intl. Conf. on Data Science and Advanced Analytics*. IEEE, 2016, pp. 130–139.

[85] H. Liu, T. Taniguchi, Y. Tanaka, K. Takenaka, and T. Bando, "Visualization of driving behavior based on hidden feature extraction by using deep learning," *IEEE Trans. on Intel. Transp. Sys.*, vol. 18, no. 9, pp. 2477–2489, 2017.

[86] C. Miyajima and K. Takeda, "Driver-behavior modeling using on-road driving data: A new application for behavior signal processing," *IEEE Signal Processing Magazine*, vol. 33, no. 6, pp. 14–21, nov 2016.

[87] N. Li, T. Misu, and A. Miranda, "Driver behavior event detection for manual annotation by clustering of the driver physiological signals," in *9th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 2583–2588.

[88] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1773–1795, dec 2013.

[89] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *ROBOMECH Journal*, vol. 1, no. 1, jul 2014.

[90] M. S. Shirazi and B. Morris, "Observing behaviors at intersections: A review of recent studies & developments," in *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, jun 2015.

[91] O. P. Popoola and K. Wang, "Video-based abnormal human behavior recognition—a review," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 865–878, nov 2012.

[92] A. T. Schulz and R. Stiefelhagen, "Pedestrian intention recognition using latent-dynamic conditional random fields," in *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2015.

[93] S. Yi, H. Li, and X. Wang, "Pedestrian behavior understanding and prediction with deep neural networks," in *ECCV*, 2016.

[94] Z. Li, I. V. Kolmanovsky, U. V. Kalabić, E. M. Atkins, J. Lu, and D. P. Filev, "Optimal state estimation for systems driven by jump-diffusion process with application to road anomaly detection," *IEEE Transactions on Control Systems Technology*, 2016.

[95] E. Lampiri, "Sensor anomaly detection and recovery in a nonlinear autonomous ground vehicle model," in *2017 11th Asian Control Conference (ASCC)*. IEEE, dec 2017.

[96] W. Li, V. Mahadevan, and N. Vasconcelos, "Anomaly detection and localization in crowded scenes," *TPAMI*, 2013.

[97] C. Lu, J. Shi, and J. Jia, "Abnormal event detection at 150 fps in matlab," in *ICCV*, 2013.

[98] W. Sultani, C. Chen, and M. Shah, "Real-world anomaly detection in surveillance videos," in *CVPR*, 2018.

[99] J. Fang, D. Yan, J. Qiao, and J. Xue, "Dada: A large-scale benchmark and model for driver attention prediction in accidental scenarios," *arXiv:1912.12148*, 2019.

[100] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury, and L. S. Davis, "Learning temporal regularity in video sequences," in *CVPR*, 2016.

[101] J. R. Medel and A. Savakis, "Anomaly detection in video using predictive convolutional long short-term memory networks," *arXiv:1612.00390*, 2016.

[102] Y. S. Chong and Y. H. Tay, "Abnormal event detection in videos using spatiotemporal autoencoder," in *ISNN*, 2017.

[103] W. Luo, W. Liu, and S. Gao, "Remembering history with convolutional lstm for anomaly detection," in *ICME*, 2017.

[104] ——, "A revisit of sparse coding based anomaly detection in stacked rnn framework," in *ICCV*, 2017.

[105] D. Gong, L. Liu, V. Le, B. Saha, M. R. Mansour, S. Venkatesh, and A. v. d. Hengel, "Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection," in *ICCV*, 2019.

[106] J. Wang and A. Cherian, "Gods: Generalized one-class discriminative subspaces for anomaly detection," in *ICCV*, 2019.

[107] R. T. Ionescu, F. S. Khan, M.-I. Georgescu, and L. Shao, "Object-centric autoencoders and dummy anomalies for abnormal event detection in video," in *CVPR*, 2019.

[108] M. P. DaSilva, "Analysis of event data recorder data for vehicle safety improvement," NHTSA, Tech. Rep. DOT HS 810 935, 2008.

[109] S. Narayanasamy, G. Pokam, and B. Calder, "BugNet: Continuously recording program execution for deterministic replay debugging," in *32nd Intl. Symp. on Computer Arch. (ISCA)*. IEEE, 2005.

[110] A. Perez, M. I. Garcia, M. Nieto, J. L. Pedraza, S. Rodriguez, and J. Zamorano, "Argos: An advanced in-vehicle data recorder on a massively sensorized vehicle for car driver behavior experimentation," *IEEE Trans. on Intel. Transp. Sys.*, vol. 11, no. 2, pp. 463–473, 2010.

[111] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, and J. Heidemann, "An evaluation of multi-resolution storage for sensor networks," in *Proc. 1st Intl. Conf. on Emb. Net. Sens. Sys.* ACM, 2003.

[112] Y. Diao, D. Ganesan, G. Mathur, and P. J. Shenoy, "Rethinking data management for storage-centric sensor networks." in *CIDR*, vol. 7, 2007, pp. 22–31.

[113] W. Li, V. Mahadevan, and N. Vasconcelos, "Anomaly detection and localization in crowded scenes," *TPAMI*, 2014.

[114] J. Bakker, H. Jeppsson, L. Hannawald, F. Spitzhüttl, A. Longton, and E. Tomasch, "Iglad-international harmonized in-depth accident data," in *ESV*, 2017.

[115] N. Li, Y. Yao, I. Kolmanovsky, E. Atkins, and A. Girard, "Game-theoretic modeling of multi-vehicle interactions at uncontrolled intersections," *arXiv preprint arXiv:1904.05423*, 2019.

[116] Y. Yao and E. Atkins, "The smart black box: A value-driven automotive event data recorder," in *ITSC*, 2018.

[117] ——, "The smart black box: A value-driven high-bandwidth automotive event data recorder," *T-ITS*, 2020.

[118] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[119] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik, "Recurrent network models for human dynamics," in *ICCV*, 2015.

[120] H. O. Jacobs, O. K. Hughes, M. Johnson-Roberson, and R. Vasudevan, "Real-time certified probabilistic pedestrian forecasting," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2064–2071, 2017.

[121] C. Anderson, X. Du, R. Vasudevan, and M. Johnson-Roberson, "Stochastic sampling simulation for pedestrian trajectory prediction," *arXiv preprint arXiv:1903.01860*, 2019.

[122] J. Bütepage, H. Kjellström, and D. Kragic, "Anticipating many futures: Online human motion prediction and generation for human-robot interaction," in *ICRA*, 2018.

[123] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *NIPS*, 2015.

[124] A. Bhattacharyya, B. Schiele, and M. Fritz, "Accurate and diverse sampling of sequences based on a "best of many" sample objective," in *CVPR*, 2018.

[125] B. D. Lucas, T. Kanade *et al.*, "An iterative image registration technique with an application to stereo vision," *DARPA Image Understanding Workshop*, 1981.

[126] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv:1406.1078*, 2014.

[127] L. A. Thiede and P. P. Brahma, "Analyzing the variety loss in the context of probabilistic trajectory prediction," in *ICCV*, 2019.

[128] J. Gallier and J. H. Gallier, *Curves and surfaces in geometric modeling: theory and algorithms*. Morgan Kaufmann, 2000.

[129] A. Simonelli, S. R. Bulo, L. Porzi, M. López-Antequera, and P. Kontschieder, "Disentangling monocular 3d object detection," in *CVPR*, 2019.

[130] http://pytorch.org/.

[131] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *ICIP*, 2017.

[132] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, "Mvtec ad–a comprehensive real-world dataset for unsupervised anomaly detection," in *CVPR*, 2019.

[133] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *CVPR*, 2017.

[134] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *T-RO*, 2017.

[135] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Gated feedback recurrent neural networks," in *ICML*, 2015.

[136] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning, lecture 6a: Overview of mini–batch gradient descent."

[137] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *CVPR*, 2014.

[138] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev *et al.*, "The kinetics human action video dataset," *arXiv:1705.06950*, 2017.

[139] NHTSA, "Traffic safety facts 2015," *NHTSA*, 2015.

[140] R. Girshick, "Fast r-cnn," in *ICCV*, 2015.

[141] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[142] W. Choi, "Near-online multi-target tracking with aggregated local flow descriptor," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3029–3037.

[143] Y. Xiang, A. Alahi, and S. Savarese, "Learning to track: Online multi-object tracking by decision making," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[144] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.

[145] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[146] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *arXiv preprint arXiv:1511.00561*, 2015.

[147] N. Li, D. W. Oyler, M. Zhang, Y. Yildiz, I. Kolmanovsky, and A. R. Girard, "Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems," *T-CST*, 2017.

[148] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner, "TORCS, The Open Racing Car Simulator," http://www.torcs.org, 2014.

[149] S. E. Lee, E. C. Olsen, and W. W. Wierwille, "A comprehensive examination of naturalistic lane-changes," NHTSA, Tech. Rep. DOT HS 809 702, 2004.

[150] W. Wang, J. Xi, and D. Zhao, "Driving style analysis using primitive driving patterns with bayesian nonparametric approaches," *arXiv preprint arXiv:1708.08986*, 2017.

[151] E. Wit, E. van den Heuvel, and J.-W. Romeijn, "'all models are wrong...': an introduction to model uncertainty," *Statistica Neerlandica*, vol. 66, no. 3, pp. 217–236, jul 2012.

[152] E. Simpson, S. Reece, A. Penta, and S. Ramchurn, "Using a bayesian model to combine lda features with crowdsourced responses," in *TREC*, 2012.

[153] C. G. Cassandras and S. Lafortune, Eds., *Introduction to Discrete Event Systems*. Springer US, 2008. [Online]. Available: https://doi.org/10.1007/978-0-387-68612-7

[154] T. H. Cormen, "Introduction to algorithms, 3rd edition (mit press)," in *Introduction to Algorithms, 3rd Edition*. The MIT Press, jul 2009, ch. 6, pp. 162–166.

[155] G. Su, N. Li, Y. Yildiz, A. Girard, and I. Kolmanovsky, "A traffic simulation model with interactive drivers and high-fidelity car dynamics," *IFAC-PapersOnLine*, vol. 51, no. 34, pp. 384–389, 2019.