From Recognition to Prediction: Analysis of Human Action and Trajectory Prediction in Video

Junwei Liang

CMU-LTI-21-008

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15123
www.lti.cs.cmu.edu

Thesis Committee:

Alexander Hauptmann (Chair)	Carnegie Mellon University
Alan W. Black	Carnegie Mellon University
Kris Kitani	Carnegie Mellon University
Lu Jiang	Google Research

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
In Language and Information Technologies

Keywords: Action Recognition, Trajectory Prediction, Action Prediction, Human Behavioral Analysis, Future Prediction, Machine Perception, Autonomous Driving

Abstract

With the advancement in computer vision deep learning, systems now are able to analyze an unprecedented amount of rich visual information from videos to enable applications such as autonomous driving, socially-aware robot assistant and public safety monitoring. Deciphering human behaviors to predict their future paths/trajectories and what they would do from videos is important in these applications. However, human trajectory prediction still remains a challenging task, as scene semantics and human intent are difficult to model. Many systems do not provide high-level semantic attributes to reason about pedestrian future. This design hinders prediction performance in video data from diverse domains and unseen scenarios. To enable optimal future human behavioral forecasting, it is crucial for the system to be able to detect and analyze human activities as well as scene semantics, passing informative features to the subsequent prediction module for context understanding.

In this thesis, we conduct human action analysis and develop robust algorithms and models for human trajectory prediction in urban traffic scenes. This thesis consists of three parts. The first part analyzes human actions. We aim to develop an efficient object detection and tracking system similar to the perception system used in self-driving, and tackle the action recognition problem under weakly-supervised learning settings. We propose a method to learn viewpoint invariant representations for video action recognition and detection with better generalization. In the second part, we tackle the problem of trajectory forecasting with scene semantic understanding. We study multi-modal future trajectory prediction using scene semantics and exploit 3D simulation for robust learning. Finally, in the third part, we explore using both scene semantics and action analysis for the prediction of human trajectories. We show our model efficacy on a new challenging long-term trajectory prediction benchmark with multi-view camera data in traffic scenes.

Thank you to my academic adviser, Alex, the best teacher in the world, who has guided me both in research and in life for the past six years. Thank you to the committee who kept me on track and co-authors along the way that provided me with great help and advice. Thank you to Lu, who mentored me and led me to everything I know about computer science research. And finally, thank you to Xinxin, my lovely wife, who traveled through Cambodia for a month to get to the U.S., to accompany and support me during the COVID-19 pandemic, and to be the pillar of my life.

Contents

Chapter 1

Introduction

1.1 Motivation of Research

With the advancement in deep learning and computer vision, systems now are able to analyze an unprecedented amount of rich visual information from videos to enable many AI applications. Researchers have achieved great successes in a wide range of computer vision tasks like image classification [70], object detection and instance segmentation [71], object tracking [190, 241] and scene semantic segmentation [32]. Computer vision engineers and scientists are able to put these models into production for applications like image/video content retrieval and in areas like retail and public safety. One of the most exciting applications is autonomous driving, which may revolutionize how people get around places and how freight moves. While many prototypes from companies like Waymo and Baidu Apollo have been built over the years, many challenges remain. One of the main challenges is collision avoidance, which is crucial for self-driving systems co-mingling with humans. This requires systems to anticipate human motions in the future. This important analysis is called future person trajectory prediction. Many existing systems do not provide high-level semantic attribute detection like current human activities and intention recognition to reason about pedestrian future. In this thesis, our goal is to build a robust trajectory prediction system with in-depth semantic context understanding. We utilize joint analysis of human actions and enhanced contextual cues from the environment to achieve intent-aware future trajectory prediction. The future trajectory prediction task has received a lot of attention in the research community [3, 67, 103, 131, 179] (see also this comprehensive survey paper [186]). It is regarded as an fundamental building block in video understanding because forecasting human behaviors is useful in many applications other

than self-driving cars like socially-aware robots [140], advanced surveillance systems, etc.

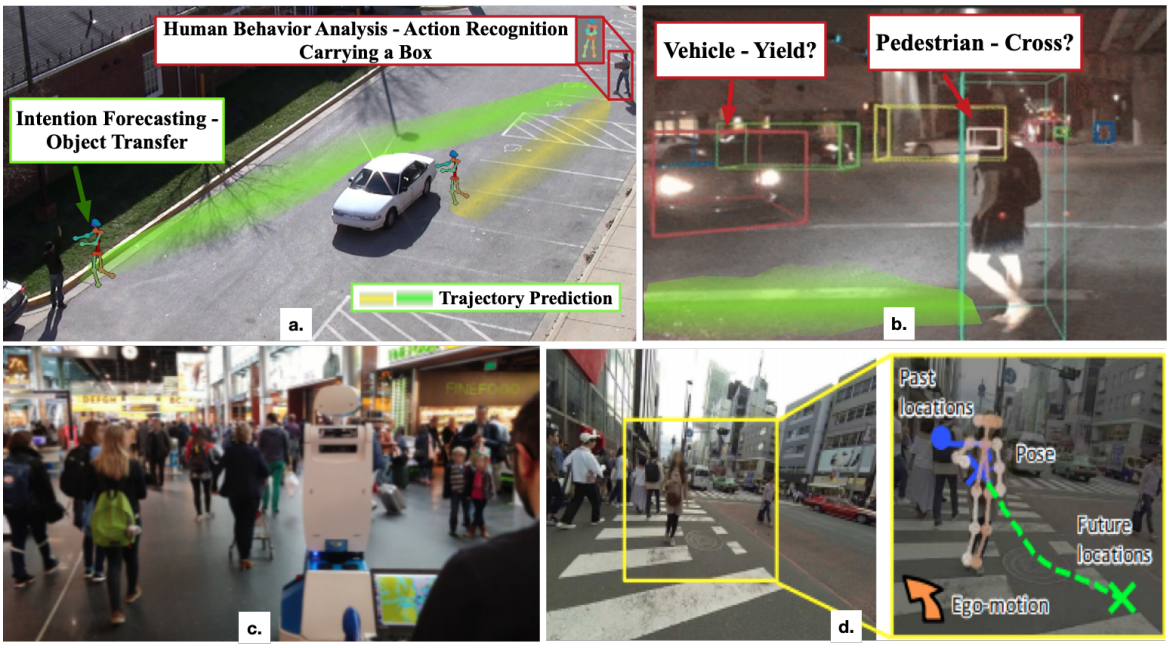

Figure 1.1: Our goal is to jointly analyze scene semantics and human actions to predict their future trajectories. (a.) Joint analysis in stationary 45-degree view cameras. The green and yellow line show two possible future trajectories. By recognizing that the person is carrying a box, the system should be able to forecast the intention based on scene constraints and social interactions (the person at the bottom left is waving at the target person), and predict the correct future trajectory. (b.) Future trajectory prediction in self-driving system. (c.) Socially-aware robots from [186]. (d.) First-person view trajectory prediction [253].

Humans navigate through public spaces often with specific purposes in mind, ranging from simple ones like entering a room to more complicated ones like putting things into a car. Even though their intentions might be determined early, their future trajectories might also be altered by other constraints like social interactions and scene constraints (i.e., external stimuli, as noted in [186]). This poses several challenges for the pedestrian future trajectory and action prediction task as follows:

(1) The scene constraints are complex and they are changing dynamically. In urban environment, the scene can be diverse and unpredictable with multiple actors including vehicles and pedestrians performing different tasks. For self-driving systems, as the cars are moving around, the prediction models have to adapt to the dynamic changes of the scene (as well as taking into account the ego-motions), which makes the forecasting problem even more complex.

We need prediction models that are robust to scene and viewpoint variants.

(2) The future is often uncertain. Given the same historical trajectory, a person may take different paths, depending on their (latent) goals. Consider the example in Fig. 4.1 a., the person (at the top-right corner) might take different paths depending on their intention, e.g., they might take the green path to *transfer object* or the yellow path to *load object into the car*. We have also demonstrated this nature in our proposed Forking Paths dataset in chapter 5. Thus recent work has started focusing on *multi-future trajectory prediction* [24, 117, 121, 146, 217, 219].

(3) Training data is limited for rare scenarios. For self-driving applications, safe operation is a priority. Data from incidents can help us better improve the system but they are often very rare and scene-dependent. Some traffic events are impossible or too dangerous to be acted out by actors for data collection purposes. As large amounts of well-annotated video data for such rare scenarios are hard to get, in this thesis we investigate the usage of 3D simulator and weakly-supervised learning to alleviate this challenge. We also propose a new multi-view trajectory prediction dataset, with rich activity annotations that could enable enhanced contextual cue models.

With the advancement of Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) and graphical models [231], we will investigate how to design efficient and robust models suitable for joint scene semantic and action analysis for future trajectory predictions. To tackle the aforementioned challenges, in Part I, we first explore and develop action analysis perception system that is efficient to capture behavioral cues and semantic attributes of pedestrians. We also look into weakly-supervised learning by utilizing vastly available Internet data in chapter 3 to get long-tail action training data. We aim to build robust models that could produce viewpoint invariant representations in chapter 4. In Part II, we propose multi-modal trajectory prediction model and study domain robustness. For the second challenge, we develop multi-future trajectory prediction model and a new benchmark in chapter 5. For the third challenge, we utilize recent success of learning from 3D simulation [187] and adversarial learning [222] to train a robust model in chapter 6. In Part III, we develop a prediction system that could utilize enhanced contextual cues in the scene and study the more challenging long-term trajectory prediction problem in traffic scenes.

1.2 Applications

Forecasting human behaviors in complex urban environment is useful in many applications such as autonomous driving, public/traffic safety monitoring, socially-aware robots, etc.

1.2.1 Autonomous Driving

Safe operation is a crucial prerequisite for self-driving cars to become ubiquitous in the future. Forecasting human behavior including pedestrians and cyclists is an essential building block to achieve that [99, 131]. In Fig. 4.1 b., it shows an example camera view from the self-driving cars. The human action analysis and prediction system is key to recognizing the pedestrian and whether they are going to continue to walk across the road, making sure that the driving planning system makes the right action for maximum safety.

1.2.2 Socially-aware Robots

Socially-aware smart robot assistants (Fig. 4.1 c.) can be utilized in the future to help humans in performing everyday tasks, such as grocery shopping and delivery. They are required to operate in a safe and socially-acceptable manner, since they would move closely among other people. The ability to analyze and predict human behaviors from first-person camera view (Fig. 4.1 d.) is essential for motion planning of the robots.

1.2.3 Advanced Public/Traffic Safety Monitoring

Human action analysis and prediction system can also be applied to stationary cameras for safety monitoring. With trajectory prediction enabled, traffic safety system can issue warnings to pedestrians and vehicles before potential accidents happen. Meanwhile, by looking into the predicted collided trajectories that do not actually happen, traffic safety researchers can study these cases as near-misses to identify risky intersections or roadways. Furthermore, trajectory prediction can enable analysis and control of crowd flow in populated areas such as malls and airports by combining other crowd dynamics analysis tools like crowd counting [38] (see this news story [1] from the Washington Post how our crowd dynamics analysis has been used in real-world scenario). In terms of public safety monitoring, the forecasting system can also be applied to predicting the path a suspect may flee after an robbery or other crime incidents, by using long-term prediction model that takes into account scene constraints and external knowledge. This would be an important addition to the set of existing public safety tools like gunshot detection [125], shooter localization [129], social media monitoring [81] and event reconstruction [30, 124].

[1] https://www.washingtonpost.com/investigations/interactive/2021/dc-police-records-capitol-riot/

1.3 Thesis Organization

This thesis aims to predict pedestrian future trajectories by jointly analyzing human behaviors and contextual cues from the environment. We first investigate human action analysis, including efficient object detection and viewpoint-invariant human action recognition & detection in videos. This provides important semantic attribute inference and robust representations that could help aid human intention recognition. We then focus on developing models for future trajectory prediction with semantic scene understanding. Finally, we utilize our findings from the first two parts and develop a robust model for challenging long-term trajectory prediction, on a newly annotated multi-view benchmark. A detailed overview of each part is as follows:

Part I Human Action Analysis In this part, we study the problem of human action analysis, including efficient object detection and tracking, and action analysis. For object detection and tracking, our goal is to build an efficient perception system that utilizes robust image object detection model and fast traditional tracking method for extended videos (chapter 2). Our focus in this part is on action recognition and detection. As human actions are diverse, it is impossible to collect training data for all action classes. Therefore we first study weakly-supervised learning with massive video data with weak labels from Internet platforms like YouTube (chapter 3). To enable better generalization of video action recognition models to all kinds of camera views, we then investigate viewpoint-invariant representation learning (chapter 4).

Part II Pedestrian Trajectory Prediction with Scene Semantics In this part, we study how trajectory prediction can benefit from semantic context understanding of the scene. Since the future is uncertain, we first introduce the *Multiverse* model to tackle the multiple-future trajectory prediction problem (chapter 5). To alleviate the limited training data challenge as mentioned in previous section, we propose a machine learning algorithm called *SimAug*, to efficiently learn from 3D simulation data for robust trajectory prediction (chapter 6).

Part III Joint Analysis of Human Actions and Trajectory Prediction In the final part, we aim to build robust prediction model with enhanced contextual cues from scene semantics and human action representations through multi-task learning. We first study jointly predicting short-term pedestrian trajectories and activities on common benchmarks (chapter 7). Since short-term future prediction is not enough to ensure safe operations in autonomous driving applications, we introduce a new, human-annotated long-term trajectory and action prediction

benchmark with multi-view camera data (chapter 8) in urban traffic scenes. Finally, we utilize our findings from the first two parts and develop a robust model for the aforementioned challenging long-term trajectory prediction task (chapter 9).

1.4 Contributions

In this section, we briefly discuss our contributions in each task considered in this thesis.

1.4.1 Part I: Human Action Analysis

As noted before, our goal is to build a robust trajectory prediction system with enhanced semantic context understanding. In order to achieve that, in Part I, we first focus on machine perception, as it is important for prediction models to get enhanced contextual information from the scene and the target agent [186]. We investigate efficient object detection and tracking models in chapter 2. It is not our intention to achieve state-of-the-art on common benchmarks like MSCOCO [136], but to create an efficient framework for video perception that is easy to swap in new models in the future. In chapter 3, we are one of the early works that study how we could better utilize weakly-supervised video data from the Internet using self-paced curriculum learning [91]. In chapter 4, we explore viewpoint-invariant representation for action recognition and detection, which is one of the early works to tackle this problem in video.

1.4.2 Part II: Pedestrian Trajectory Prediction with Scene Semantics

In Part II, we study trajectory prediction models with scene semantic understanding. Following the taxonomy proposed in [186], in chapter 5, we study a pattern-based sequential model that considers scene semantics as contextual cues for multi-future trajectory prediction. In this work, we propose the first human-annotated benchmark for multi-future trajectory prediction. We then explore adversarial learning in chapter 6 to build robust trajectory prediction models for different environment and camera views, which is an important under-explored problem in this research field [186].

1.4.3 Part III: Joint Analysis of Human Actions and Trajectory Prediction

In Part III, we study trajectory prediction models with scene semantic understanding and action analysis. In chapter 7, we develop the first joint action and trajectory prediction model. In this work, we are also among the very early works that look into how **noisy input** of imperfect object detection and tracking would affect trajectory prediction performance. Such robustness experiments have been noted in [186] as very important to understand and measure the effectiveness of the prediction models. In chapter 8, we propose a new human-annotated long-term trajectory prediction benchmark with multi-view video data in urban traffic scenes. In chapter 9, we propose a new model that utilizes both scene understanding and viewpoint-invariant action representation (chapter 4) for the challenging long-term trajectory prediction problem in urban traffic scenes.

1.5 Overall Impact

This thesis has potential impact on crowd dynamics analysis in general, in particular, on several directions of the pedestrian trajectory prediction research, including multi-modal trajectory prediction (chapter 5), learning from 3D simulation (chapter 6), learning from enhanced contextual cues (chapter 7, chapter 8, chapter 9), to name a few. This thesis also has potential impact on the computer vision perception direction, specifically on efficient object detection and tracking (chapter 2), weakly-supvervised learning (chapter 3), and action recognition (chapter 4). Below, we highlight the impact of our work in academia and industry.

Impact in Academia

- Our work has been published at top conferences and journals including CVPR, TPAMI, ECCV, AAAI, IJCAI, etc. As of June 2021, our work has received a total of more than 200 citations from all over the world.
- Our work on using enhanced contextual cues for trajectory prediction (CVPR'19, chapter 7) has received 140+ citations and it is one of the top-cited paper at CVPR 2019 in this research field. Notably, many researchers have extended our work in terms of multi-task learning for trajectory prediction [15, 175], action prediction [28, 108] and egocentric-view trajectory prediction [19, 165, 172]. Our work has also inspired [232, 240] to propose more efficient models that are suitable for deployment and [28, 212, 252] have developed

a new improved version of our work with graph models for single-future trajectory prediction.

- Our recent work on multimodal prediction and simulation dataset have also started to gain momentum in the research community. Our Multiverse model (CVPR'20, chapter 5) has inspired follow-up work to extend the use of grid occupancy maps[23, 60, 101, 161, 175, 189] and spatial-temporal graph attention [9, 14, 232] for multimodal trajectory prediction. Our dataset using 3D simulation (chapter 5 and chapter 6) has enabled [6, 147, 167] to work on this new problem setting for trajectory prediction and [144, 169] have used our dataset for multimodal evaluation.
- Most of our research work has been open-sourced and our Github repositories have a total of 800+ stars and 300+ forks as of June 2021. Our chapter 7 work is #1 Tensorflow-based code on PaperWithCode [2] in trajectory prediction task.

Impact in Industry

- Our weakly-supervised work chapter 3 has won the TRECVID Ad-hoc Video Search Challenge [3], and our efficient object detection and tracking work chapter 2 has won the TRECVID Activities in Extended Videos Challenge. Our pedestrian trajectory prediction work, which is related to traffic safety and public safety, has won the NIST ASAPS challenge [4] with a $30,000 prize.
- Our trajectory prediction work has been implemented for a COVID-19 related project called Social Distancing Early Forecasting System [5] and we have received $6200 research grant for this from Google Cloud.
- Our efficient object detection and tracking code (chapter 2) has been used by many companies, including Paravision, iMotion Germany, Neosperience, etc.

Media Coverage

- Our work on trajectory prediction has been reported by two major Chinese Tech outlet and it received over 30,000 views in a week. Feb 14, 2019.
- Our blog on pedestrian trajectory prediction has been mentioned and referenced by many bloggers [6].

[2] https://paperswithcode.com/task/trajectory-prediction
[3] https://trecvid.nist.gov/
[4] https://www.herox.com/ASAPS1/update/3483
[5] https://github.com/JunweiLiang/social-distancing-prediction
[6] https://medium.com/@junweil/social-distancing-early-forecasting-system-60186baa6

1.6 Terminology

In this paper, we use the term "trajectory" and "path" interchangeably to refer to the ground-level 2D positions of an agent (humans, vehicles, etc.) over a period of time. We refer to short-term and long-term prediction to characterize prediction horizons of 3-5 seconds and 12 seconds ahead, respectively. The "long-term" definition is consistent with recent publications in the field [99, 186, 225]. We also use the term "actions" and "activities" interchangeably to refer to a predefined set of human actions of various duration from public datasets (VIRAT [158], Kinetics [102], etc.). We use "intent", "intention" and "goal" interchangeably to refer to a person's latent, near-future (same time horizon as the prediction) objective or purpose that associates with a physical destination and an action.

Part I

Human Action Analysis

In this part, we study the problem of human action analysis, including efficient object detection and tracking, and action recognition. For object detection and tracking, our goal is to build an efficient perception system that utilizes robust image object detection model and fast traditional tracking method for extended videos (chapter 2). Our focus in this part is on action recognition and detection. As human actions are diverse, it is impossible to collect training data for all action classes. Therefore we first study weakly-supervised learning with massive video data with weak labels from Internet platforms like YouTube (chapter 3). To enable better generalization of video action recognition models to all kinds of camera views, we then investigate viewpoint-invariant representation learning (chapter 4).

Chapter 2

Efficient Object Detection and Tracking in Extended Videos

In this chapter, we study the problem of building an efficient perception system for extended videos, either from surveillance cameras or onboard cameras from self-driving cars. The perception system, which includes object detection and tracking models, provides the basic inputs, i.e., person and object tracklets, to action recognition and future prediction models in later chapters. We study the object detection and tracking models for videos in the Activities in Extended Videos Prize Challenge (ActEV) [8, 138]. With the availability of large-scale video surveillance dataset such as VIRAT [158], ActEV seeks to encourage the development of real-time robust automatic activity detection algorithms in surveillance scenarios. We aim to develop a robust and efficient object detection and tracking model trained on the MSCOCO dataset [136] as the object annotations in the ActEV dataset are quite small compared to MSCOCO. Many image object detection research works [41, 71] have been done on the MSCOCO dataset [136], which includes 80 object classes. Compared to other object detection datasets including PASCAL VOC dataset [53], MSCOCO has better annotations, with many more bounding boxes and classes (The YouTube-BoundBox dataset [176] has more boxes than MSCOCO dataset but it is not exhaustively annotated). We also study Tensorflow [1] and TensorRT to build a perception system that reaches faster-than-real-time relative processing speed with certain hardware requirements. It is not our intention to develop State-of-the-Art (SOTA) object detection model but instead to construct a framework for extended videos that we could easily swap the model

Code and models are released at https://github.com/JunweiLiang/Object_Detection_Tracking

with the latest COCO-trained models (we have implemented Mask-RCNN [71] from 2017 and EfficientDet [215] from 2020.)

2.1 Motivation

Figure 2.1: Example of the ActEV dataset. The videos are annotated with pedestrian and vehicle tracklets with bounding boxes as well as the pedestrian's activities. For example, the two persons at the bottom-left are labeled with "talking".

In recent years, the volume of video data from widely deployed surveillance cameras has grown dramatically. However, camera network operators are overwhelmed with the data to be monitored, and usually cannot afford to view or analyze even a small fraction of their collections. To enable timely response for public safety and traffic safety events, there is thus strong incentive to develop fully-automated methods to identify and localize activities in extended video collections and provide the capability to alert and triage emergent videos. These methods will alleviate the current manual process of monitoring by human operators and scale up with the growth of sensor proliferation in the near future. With the availability of largescale video surveillance dataset such as VIRAT [158], the Activities in Extended Videos Prize Challenge (ActEV) [8] seeks to encourage the development of real-time robust automatic activity detection algorithms in surveillance scenarios. Specifically, an activity is defined to be "one or more people (or vehicle) performing a specified movement or interacting with an object or group of objects". Fig. 2.1 illustrates the "talking", "open_trunk" and "pull" activities. A few recent work applied a two-stage architecture for temporal action localization [43, 248], and demonstrated competitive performance. In particular, R-C3D network [248] closely follows the original Faster R-CNN [178] architecture but in the temporal domain. There are also several

previous works [61, 77] focusing on online action detection or fine-grained action detection untrimmed Internet videos. However,these methods may not generalize to a more challenging spatial-temporal activity detection problem, which is central for extended video analysis in surveillance videos and self-driving cameras. In this chapter, we describe our object detection and tracking models to generate action proposals of our ActEV system.

2.2 Efficient Object Detection and Tracking

The activities of concern in ActEV are summarized in Table 2.1. These activities involve either person or vehicle object, we use this prior knowledge to build the event proposal module starting from the object detection step. The output of this step is person and vehicle bounding box for each frame. The immediate natural next step is to associate detected object across frames, which is tracking. The output of this step is person tracklet and vehicle tracklet.

Type	Activities
Person-Only	Transport_HeavyCarry, Riding, Talking, Activity_carrying, Specialized_talking_phone, Specialized_texting_phone, Entering, Exiting, Closing, Opening
Vehicle-Only	Vehicle_turning_left, Vehicle_turning_right, Vehicle_u_turn
Person-Vehicle	Open_Trunk, Loading, Closing_trunk, Unloading

Table 2.1: Activity categorization on the ActEV dataset.

2.2.1 Object Detection

We utilize Mask-RCNN [71] with feature pyramid network [137] on ResNet [70] as the backbone for object detection, in which RoIAlign [71] is used to extract features for Region-of-Interest. We also experiment with the latest state-of-the-art EfficientDet [215] model. The object detection models are trained on either MSCOCO [136] object detection training set or the VIRAT [158] training set. We apply object detection on every k frame (we find 8 to be a good speed-and-accuracy-trade-off number) in the videos. Full resolution images (1920x1080) or HD resolution (1280x720) images are input to the model to generate object bounding boxes.

Experimental Results. We first experiment with a number of variations of Mask-RCNN and EfficientDet models on the VIRAT dataset, a representative dataset of extended videos, and the

	COCO-mAP	VIRAT-Person Val-AP	VIRAT-Vehicle Val-AP	VIRAT-Bike Val-AP
MaskRCNN, R50-FPN	0.389	0.374	0.943	0.367
MaskRCNN, R101-FPN	0.407	0.378	0.947	0.399
EfficientDet-d2	0.425	0.371	0.949	0.293
EfficientDet-d6	0.513	0.422	0.947	0.355
MaskRCNN, R101-FPN*	-	0.831	0.982	0.588

Table 2.2: Object detection evaluation on the VIRAT dataset. "*" is finetuned on the VIRAT training set.

AVA-Kinetics [119] dataset, a recent action detection dataset based on Internet videos. In these experiments, we mainly evaluate the "Person" and/or "Vehicle" object classes, as they are the most dominant and useful classes for action detection and prediction.

Table 2.2 shows the object detection results on the VIRAT dataset. As we see, for MaskR-CNN, Resnet-50 backbone and Resnet-101 backbone perform similarly on detection persons and vehicles. Further finetuning helps tremendously on the VIRAT dataset. The EfficientDet models are significantly better than MaskRCNN overall on all 80 classes on the MS-COCO dataset. However, on the three objects we focus on in the VIRAT dataset, the improvement is not efficient: especially for EfficientDet-d6, the computation cost is more than 2x compared to MaskRCNN, R101-FPN.

Since our goal is to develop an efficient object detection and tracking framework for extended videos, we experiment with different infrastructure and different code bases. As shown in Table 2.3, we experiment with 3 machines, with different GPUs, CPUs and I/O condition. In Table 2.4, we show the experiments that we have run. We mainly test different versions of Tensorflow as well as TensorRT acceleration on the VIRAT validation set, with about 206k images. We use full resolution (1920x1080) inputs. The experiments are shown in Table 2.5. All these runs produce the same object detection results. As we see, later version of Tensorflow with later version of CUDA improves speed significantly. The frozen graph method in Tensorflow also helps. The TensorRT acceleration does not show significant improvement with Tensorflow version 1.14. In conclusion, it is enough to use frozen graph of Tensorflow models for object detection inferencing. Other ways to increase speed include using lower resolution input images and smaller backbone like Resnet-50, but these methods will lead to a decrease in detection average precision.

Machine Type	
Machine 1	2 GTX 1080 TI, i7, nvme
Machine 2	3 GTX 1080 TI + 1 TITAN X, E5, nvme
Machine 3	4 RTX 2080 TI , i9-9900X, SSD

Table 2.3: Machine types that we use to test object detection model speed.

Run	
1	tf 1.10 (CUDA 9, cudnn 7.1), Variable Model
2	tf 1.13 (CUDA 10.0 cudnn 7.4), Variable Model
3	tf 1.13 (CUDA 10.0 cudnn 7.4), Frozen Graph (.pb)
4	tf 1.13 (CUDA 10.0 cudnn 7.4), Frozen Graph (.pb) ->TensorRT Optimized
5	tf 1.14.0 (CUDA 10.0 cudnn 7.4), Variable Model
6	tf 1.14.0 (CUDA 10.0 cudnn 7.4), Frozen Graph (.pb)
7	tf 1.14.0 (CUDA 10.0 cudnn 7.4), Frozen Graph (.pb) ->TRT FP32
8	tf 1.14.0 (CUDA 10.0 cudnn 7.4), Frozen Graph (.pb) ->TRT FP16

Table 2.4: Experiments that we run to test the model speed on VIRAT.

Run	Machine	# GPU Used	GPU Average Utilization	Per GPU FPS
1	1	2	65.3%	2.5
2	1	2	62.0%	2.72
2	2	4	33.8%	1.76
3	2	4	23.5%	1.95
3	2	2	38.0%	2.87
2	2	1	41.4%	2.78
3	2	1	54.8%	3.56
2	2	1*4	46.2%	2.94
3	2	1*4	52.3%	3.54
5	2	1*4	53.2%	3.03
6	2	1*4	61.7%	3.84
7	2	1*4	61.0%	3.93
8	2	1*4	52.6%	3.89
2	3	1	61.2%	3.57
3	3	1	61.2%	4.74
2	3	1*4	62.6%	3.65
3	3	1*4	65.2%	4.83

Table 2.5: Speed experiments on VIRAT with the MaskRCNN model. "1*4" means 4 inference processes are run concurrently, where one process utilizes one GPU.

	COCO-mAP	AVA-Kinetics Train-Person-AP	AVA-Kinetics Val-Person-AP
MaskRCNN, R101-FPN	0.407	0.664	0.682
EfficientDet-d2	0.425	0.650	0.680
EfficientDet-d6	0.513	0.623	0.658
MaskRCNN, R101-FPN*	-	0.735	0.732

Table 2.6: Person detection evaluation on the AVA-Kinetics dataset. "*" is finetuned on the AVA-Kinetics training set.

Models	Recall (%)	Precision (%)	ID switches	MOTA (%)	MOTAL (%)
KCF	93.5	**97.1**	2519	91.3	90.5
deep SORT	**95.2**	96.5	**909**	**91.7**	**91.8**

Table 2.7: Results of multi-object tracking algorithms in the validation set of VIRAT dataset.

Table 2.6 shows the person detection results on the AVA-Kinetics dataset. As we see, even though the EfficientDet models perform better than MaskRCNN on COCO mean average precision over all 80 object classes, MaskRCNN is better on person detection on the AVA-Kinetics dataset. Further finetuning improves further. In later chapter 4, we will utilize the finetuned model for action detection experiments.

2.2.2 Object Tracking

We compare the performance of deep SORT [239] and kernelized correlation filter (KCF) [73]. As shown in Table 2.7, deep SORT outperforms KCF for all the metrics except precision. As the tracking module is used to generate tracklets which are proposal candidates, we expect a high recall and low ID switches with a comparable precision. The results are reported in Table 2.7. In the final system, we utilize deep SORT [239] to generate tracklets by associating detected objects across frames. We follow a similar track handling and Kalman filtering framework [73]. We use bounding box center positions, aspect ratio, height and their respective velocities in image coordinates as Kalman states. We compute the Mahalanobis distance between predicted Kalman states and newly arrived measurement to incorporate motion information. For each bounding box detection, we use the feature obtained from object detection module as a appearance descriptor (either from the Resnet backbone or the EfficientNet backbone). We compute the cosine distance between tracks and detections in appearance space. To build the association problem, we combine both metrics using a weighted sum. An association is defined admissible if it is within the gating region of both metrics.

2.2.3 Efficient Processing

Our system uses Python Queues to preprocess videos with CPUs while allowing GPUs to run inference on multiple images at the same time. See Figure 2.2. With parallel processing threads, we are able to leverage CPUs and GPUs efficiently.

Experiment. We conduct an experiment on a machine with a GTX 1070 TI GPU, i7-8700K

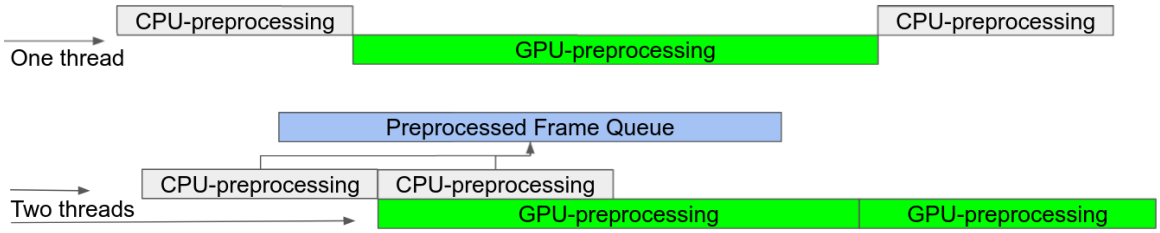

Figure 2.2: Example of the efficient processing pipeline.

CPU and SSDs. We use the FPN-ResNet50 object detection model. We run object detection and tracking on a 5-minutes video of 1280x720 resolution. Results are shown in Table 2.8. "var", "frozen", "partial" means variable, frozen, partial (only output "Person" and "Vehicle" objects) object detection model, respectively. See section 2.2.1 for more details on the different type of Tensorflow models. "m" means running the system with the aforementioned parallel processing method. As we see, the total processing time is reduced from 6 minutes to 2 minutes. In Figure 2.3 and Figure 2.4, we show the CPU and GPU utilization graph for "b=1 frozen,partial" and "b=8 frozen,partial,m" experiments, respectively. As we see, the usage of the system is improved significantly.

RunType	Time	GPU Median Utilization	GPU Average Utilization
b=1 var	06:21	53.00%	54.24%
b=1 frozen	05:06	34.50%	36.30%
b=1 frozen,partial	03:43	57.00%	49.55%
b=8 var	04:27	00.00%	2.35%
b=8 frozen	03:12	62.00%	53.37%
b=8 frozen,partial	03:07	75.50%	62.11%
b=8 var,m	03:29	100.00%	73.45%
b=8 frozen,m	02:14	67.00%	63.69%
b=8 frozen,partial,m	02:08	99.00%	83.83%

Table 2.8: Efficient object detection and tracking experiments.

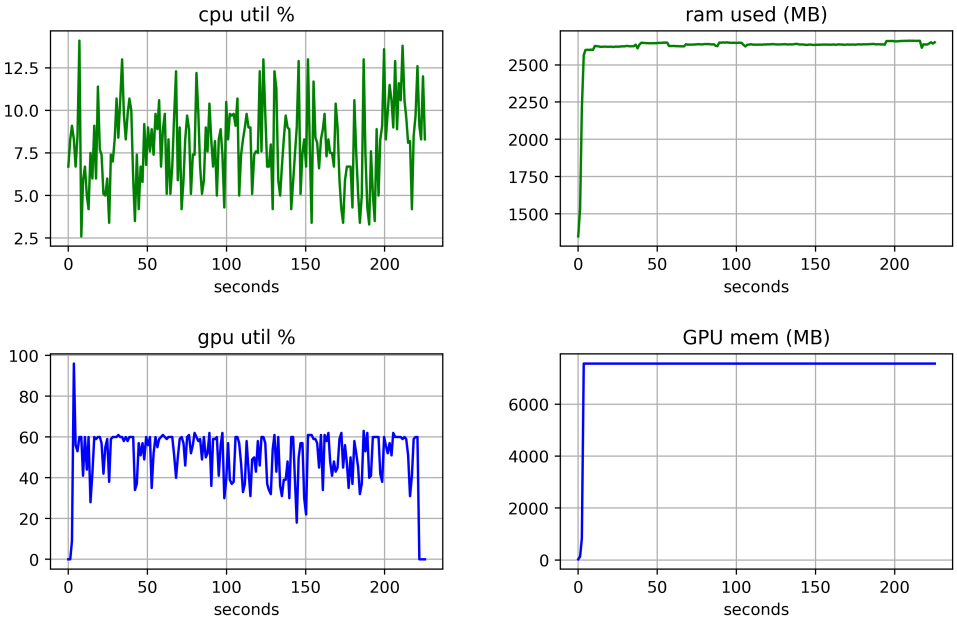

Figure 2.3: CPU and GPU utilization graph for b=1 frozen,partial experiment. See text for details

2.3 Contributions

Our object detection and tracking system is a fundamental component for the rest of the thesis. This system also supports multi-view person/vehicle long-term tracking with re-identification models. This system is used in our submission to the Activities in Extended Videos Prize Challenge (ActEV) [8, 138], where we achieves **best performance** on the leaderboard. This part of the thesis has been open-sourced at https://github.com/JunweiLiang/Object_Detection_Tracking. As of June 2021, this repository receives **240** stars and 82 forks, with a weekly traffic of **200** viewers and a dozen downloads.

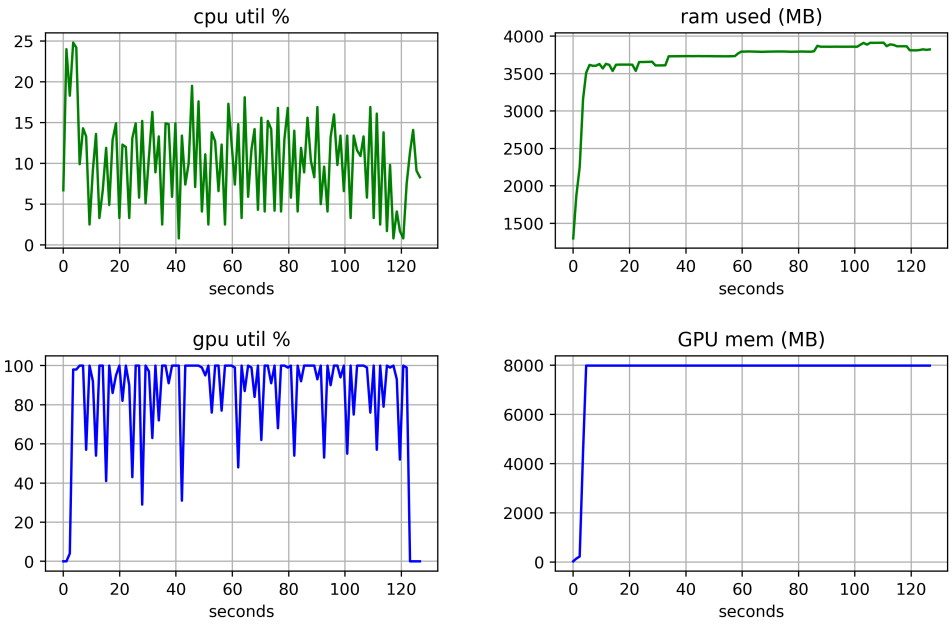

Figure 2.4: CPU and GPU utilization graph for b=8 frozen,partial,m experiment. See text for details

Chapter 3

Weakly-supervised Action Event Recognition

Learning detectors that can recognize concepts, such as people actions in video content is an interesting but challenging problem for human behavioral analysis. However, as human actions are diverse and combination of atomic actions can lead to an exponential amount of action classes, it is often difficult to have sufficient human-annotated training data for action recognition and detection. In this chapter, we present our work on webly-labeled learning with multimodal video data [123, 126, 127] to tackle the challenge of not enough manual annotations for action recognition. In the next chapter (chapter 4), we propose viewpoint-invariant feature representation learning for action recognition and detection that could be generalized to multiple action datasets.

3.1 Motivation

Millions of videos are being uploaded to the Internet every day. These videos capture different aspects of multimedia content about our daily lives. Automatically categorizing videos into concepts, such as people actions, events, etc., is an important topic. Recently many studies have been proposed to tackle the problem of concept learning [2, 46, 92, 100, 122, 133].

Many datasets acquire the clean concept labels via annotators. These datasets include ImageNet [46], TRECVID MED [159] and FCVID [95]. Collecting such datasets requires significant human effort, which is particularly expensive for video data. As a result, the labeled video collection is usually much smaller than the image collection. For example, FCVID [95], only contains about 0.09 million labels on 239 concept classes, much less than the 14 million labels

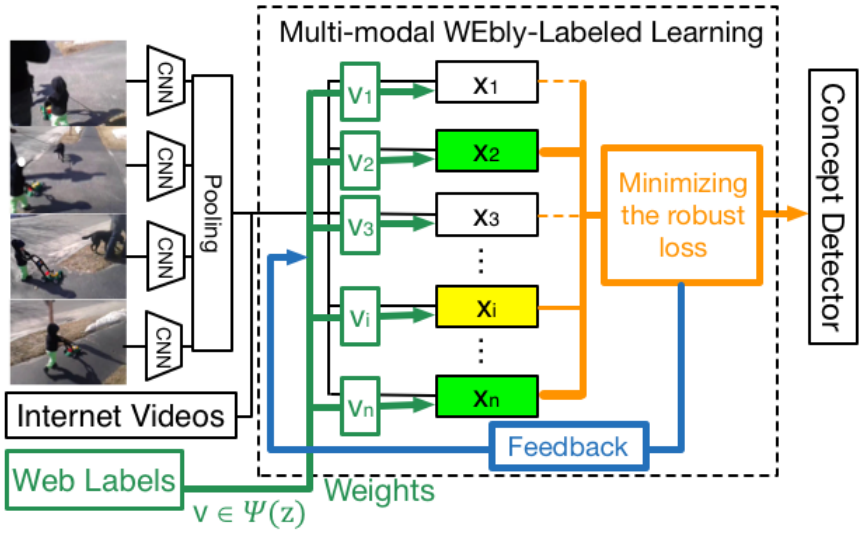

Figure 3.1: Overview of Multi-modal WEbly-Labeled Learning (WELL-MM). The algorithm jointly models the prior knowledge extracted from web labels and the current learned model at each iteration to overcome the noise labels. $\{\mathbf{x}_i\}_{i=1}^n$ are input samples and their current weights are determined by $\{\mathbf{v}_i\}_{i=1}^n$. Colored samples are the samples with nonzero weights at the current iteration. The blue line indicates the feedback from the previous objective function value.

on over 20,000 classes in the image collection ImageNet [46]. On the other hand, state-of-the-art concept models utilize deep neural networks [100], which need more data to train. However, training only on manually labeled clean data seem insufficient for large-scale concept learning.

Images or videos on the web often contain rich contextual information, such as their titles or descriptions. We can infer the label by the metadata. Fig. 3.2 shows an example of a video with inferred concept label "walking a dog". In this chapter, we call the samples with inferred labels *weakly labeled* or *webly labeled*. The webly-labeled data are easy to collect and thus usually orders-of-magnitude larger than manually-labeled data. However, the web labels are very noisy and have both low accuracy and low recall.

Concept learning over weakly labeled data becomes popular as it allows for large-scale learning on big data. However, these methods have only focused on utilizing a single text modality to model the noisy labels [33, 123]. For example, in Fig. 3.2, the textual metadata is useful but also contain lots of noises. In fact, the video is of multiple modalities and our intuition is that the inference obtained from multiple modalities is more reliable than that from a single text modality. For example, we can more confident to say this video is about "walk a

dog" if we spot the text in the title, hear the words "good boy" in the speech, and see a dog in some key frames. To this end, we can leverage the prior knowledge in automatically extracted multi-modal features from the video content such as pre-trained still image detectors, automatic speech recognition and optical character recognition. In some cases when videos have little textual metadata, multi-modal knowledge become the only useful clues in concept learning.

Recent studies on weakly labeled concept learning show promising results. However, since existing approaches only focuses on a single modality, two important questions have yet: 1) what are the important multi-modal prior knowledge, except textual metadata, for modeling noisy labels? 2) how to integrate the multiple modalities into concept learning in a theoretically sound manner?

In this chapter, to utilize multi-modal prior knowledge for concept learning, we propose a learning framework called **Multi-modal WEbly-Labeled Learning (WELL-MM)**. The learning framework is motivated by human learning, in which the learner starts from learning easier aspects of a concept, and then gradually take more complex examples into the learning process[13, 91, 111]. Specifically. WELL-MM learns a concept detector iteratively from first using a few samples with more confident labels, then gradually incorporate more samples with noisier labels. Fig. 3.1 shows the overview of the proposed framework. The algorithm integrates multi-modal prior knowledge, which is derived from the multi-modal video and image features, into the dynamic learning procedure. The idea of curriculum and self-paced learning paradigm has been proved to be efficient to deal with noise and outliers [33, 111]. Our proposed method is the first to generalize the learning paradigm to leverage multi-modal prior knowledge into concept learning. Experimental results show that multi-modal prior knowledge is important in concept learning over noisy data. The proposed WELL-MM outperforms other weakly labeled learning methods on three real-world large-scale datasets, and obtains the state-of-the-art results with recent deep learning models.

The contribution of this chapter is threefold. First, we propose a novel solution to address the problem of weakly labeled data learning through a general framework that considers multi-modal prior knowledge. We show that the proposed WELL-MM not only outperforms state-of-the-art learning methods on noisy labels, but also, notably, achieves comparable results with models trained using manual annotation on one of the video dataset. Second, we provide valuable insights by empirically investigating different multi-modal prior knowledge for modeling noisy labels. Experiments validate that by incorporating multi-modal information, our method is robust against certain levels of noisiness. Finally, the efficacy and the scalability have been demonstrated on three public large-scale benchmarks, which include datasets on both Internet

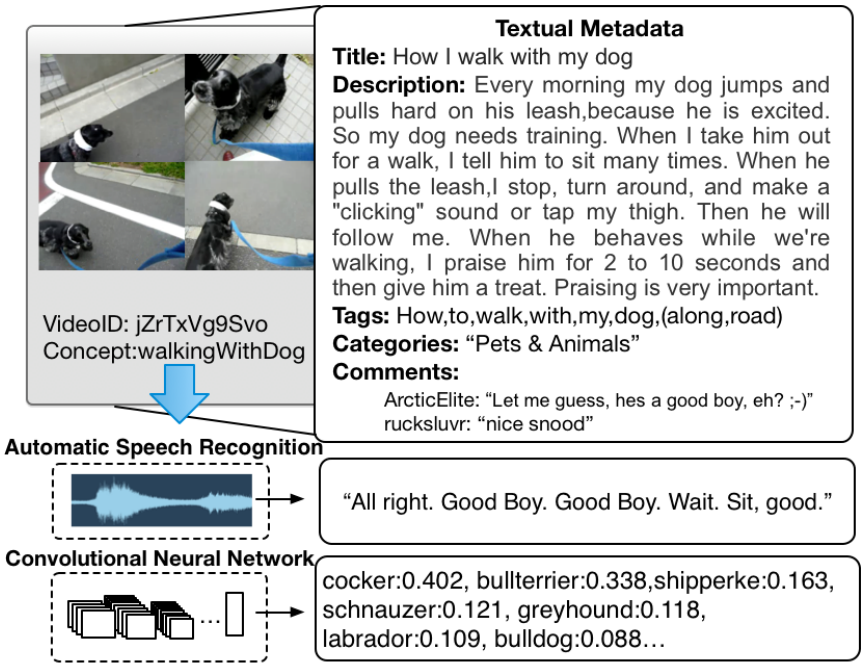

Figure 3.2: Multi-modal prior knowledge from web video.

videos and images. The promising results suggest that detectors trained on sufficient weakly labeled videos may outperform detectors trained on existing manually labeled datasets.

3.2 Multi-modal WEbly-Labeled Learning (WELL-MM)

3.2.1 Problem Description

In this chapter, following [228], we consider a concept detector as a classifier and our goal is to train concept detectors from webly-labeled video data without any manually annotated labels. Given a collection of training samples with noisy labels, we do not make any assumption over the underlying noise distribution. Formally, we represent the training set as $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{z}_i, \tilde{\mathbf{y}}_i)\}_{i=1}^{n}$, where $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ are the d-dimensional features of the training set, and $\{\mathbf{z}_1, \cdots, \mathbf{z}_n\}$ represent each sample's corresponding noisy web labels. We assume that the noisy labels are given. The noisy web labels are often automatically inferred using the sample's textual metadata provided by its uploader, or from other modalities such as pre-trained convolutional neural network over still images[27], Automatic Speech Recognition [166], or Optical character recognition [203]. For example, for instance, a video might have a noisy label "cat" as its title and speech both contain the word cat. The $\tilde{\mathbf{y}}_i \subset \mathcal{Y}$ is the inferred concept label set

for the i^{th} observed sample based on its noisy web label, and $\mathcal{Y}$ denotes the full set of target concepts. In our experiment, to simplify the problem, we employ one-versus-all strategy for multi-class classification, and discuss our method in the context of binary classification over the noisy web labels.

3.2.2 Model

Objective Function

In this section, we propose a model called Multi-modal WEbly-Labeled Learning (WELL-MM) to leverage multi-modal prior knowledge for weakly labeled data. Formally, given the training set $\mathcal{D}$ mentioned previously, Let $L(\tilde{y}_i, g(\mathbf{x}_i, \mathbf{w}))$,denote the loss function which calculates the cost between the inferred label $\tilde{y}_i$ and the estimated label given by the decision function $g(\mathbf{x}_i, \mathbf{w})$. Here $\mathbf{w}$ represents the model parameters. Our objective function is to jointly learn the model parameter $\mathbf{w}$ and the latent weight variable $\mathbf{v} = [v_1, \cdots, v_n]^T$ by:

$$\min_{\mathbf{w},\mathbf{v}\in[0,1]^n} \mathbb{E}(\mathbf{w}, \mathbf{v}; \lambda, \Psi) = \sum_{i=1}^{n} v_i L(\tilde{y}_i, g(\mathbf{x}_i, \mathbf{w})) + f(\mathbf{v}; \lambda), \quad \text{subject to } \mathbf{v} \in \Psi \tag{3.1}$$

where the latent weight variable $\mathbf{v} = [v_1, \cdots, v_n]^T$ represents the inferred labels' confidence, and thus reflects the learning sequence of samples. In order to learn concept detectors in noisy data, we utilize the self-paced regularizer f [91] to control the learning process, where f is expect to assign greater weights to samples with confident labels. For simplicity, we consider the linear regularizer Eq. (3.2) proposed in [91]:

$$f(\mathbf{v}; \lambda) = \frac{1}{2}\lambda \sum_{i=1}^{n} (v_i^2 - 2v_i), \tag{3.2}$$

$\lambda \in (0, 1)$ is a hyper-parameter that controls the pace of model training, which resembles the "age" of the model. We set λ to be small at the beginning and only samples of with small loss will be considered in training. As λ grows, more samples with larger loss will be gradually included. As stated in related studies [123, 152], the self-paced in Eq. (3.2) corresponds to a robust loss function. The robust loss in our problem tends to depress samples with noisy labels or outliers and thus may be instrumental in avoiding bad local minima.

In order to utilize the rich contextual information in the noisy data, we embed the multi-modal prior knowledge derived from the web labels $\mathbf{z}$ into a convex curriculum region Ψ for

the latent weight variables. The shape of the region weakly implies the learning sequence, where favored samples have larger expected values. Generally, Ψ can be represented by $\Psi = \{\mathbf{v} \,|\, c(\mathbf{v}, \mathbf{a}) \leq b\}$, where $\mathbf{a} = [a_1, \cdots, a_n]$ is the parameters of the region. In this chapter, we use a linear constraint to form the curriculum region [91]:

$$\Psi = \{\mathbf{v} \,|\, \sum_{i=1}^{n} a_i v_i \leq b\} \tag{3.3}$$

The curriculum region is introduced to leverage the prior knowledge about the noisy labels and, as demonstrated in our experiments, is a crucial factor in weakly labeled data learning. We use multi-modal information to derive the probabilities of samples being positive of a class and if the probabilities are below a threshold (in our experiments it is set at zero) the samples will be consider as negatives. We assign value to a_i in correlated to samples 's probabilities being in the class, and b is set to 1. We use curriculum as a warm start in training, and set μ to zero after the first iteration. Since we empirically observed that curriculum constraints mostly benefit the first few iterations. We will discuss how to derive the multi-modal curriculum in details in the following section.

Eq. (3.1) is difficult to minimize over big data due to the constraints. In this work, we propose to relax the constraints by introducing a Lagrange multiplier μ. The objective function then becomes:

$$\begin{aligned} &\min_{\mathbf{w},\mathbf{v}\in[0,1]^n} \mathbb{E}(\mathbf{w}, \mathbf{v}; \lambda, \mathbf{a}, b, \mu) = \\ &\sum_{i=1}^{n} v_i L(\tilde{y}_i, g(\mathbf{x}_i, \mathbf{w})) + \frac{1}{2}\lambda \sum_{i=1}^{n} (v_i^2 - 2v_i) + \mu(\sum_{i=1}^{n} a_i v_i - b), \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{subject to } \mu \geq 0 \end{aligned} \tag{3.4}$$

The proposed Eq. (3.4) has two benefits over Eq. (3.1). First it enables the large-scale training on noisy data. This is important because as our experiments show that training on noisy data can outperform training on manually labeled data only when the noisy data are orders-of-magnitude larger. Second, it may tolerate the noise introduced in the curriculum region.

Multi-modal Curriculum

In this section we discuss the details on how to construct the curriculum region Ψ. Ψ is a feasible region that embeds the multi-modal prior knowledge extracted from the webly-labeled data as shown in Figure. 3.2. It geometrically corresponds to a convex feasible space for the latent weight variable. Given a set of training samples $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ with corresponding noisy labels

$\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^n$, we want to extract the learning curriculum based on how related the training samples are to the target classes, which is modeled by the probability of the samples being the inferred class label (since we don't have the actual label in webly learning). The training samples with a greater value of probability mean that they are more confident to belong to the true class and should be learned earlier. Similar to Information Retrieval theory [150], here we use random variable $\mathbf{z}$ to represent the noisy web labels, $\mathbf{y}$ to represent the label classes, and the curriculum for a sample is then determined by:

$$\mathbf{P}(\mathbf{z} \mid \mathbf{y}) = \mathbf{P}(\mathbf{y} \mid \mathbf{z})\mathbf{P}(\mathbf{z})/\mathbf{P}(\mathbf{y}) \tag{3.5}$$

Since $\mathbf{P}(\mathbf{y})$ is the same for all samples, it can be regarded as a constant. The prior probability of a web video $\mathbf{P}(\mathbf{z})$ can be implemented with the duration, the view count and comments about the video. In this work we treat the prior as uniform so it can be ignored as well. Therefore, we calculate the curriculum simply based on $\mathbf{P}(\mathbf{y} \mid \mathbf{z})$, the probability of the sample being class $\tilde{y}_i$ given the noisy label. Since we want to incorporate the multi-modal prior information, we calculate the curriculum from:

$$\mathbf{P}(\mathbf{y} \mid \mathbf{z}) \propto \sum_m \theta_m \mathbf{P}(\mathbf{y} \mid \mathbf{z_m}) \tag{3.6}$$

We use random variable z_m to represent the m-th modality of the noisy labels for a sample and θ_m is the predetermined weight for modality m. In this work, other than the textual metadata, we also utilize other modalities such as Automatic Speech Recognition (ASR) [166], Optical Character Recognition (OCR) [203] and basic image detector pre-trained on still images [188] (in this work we use VGG net [201], extract keyframe-level image classification results and average them to get video-level results). Therefore the total number of the modalities is 4. We compare common ways to extract curriculum from web data for concept learning to the proposed novel method that utilizes state-of-the-art topic modeling techniques in natural language processing.

In the following methods (Word Hard Matching and Latent Topic with Word Embedding), we first extract bag-of-words features from different modalities for each video and then match them using specific matching methods to the concept words to get the probabilities in Eq. (3.6).

Word Hard Matching We build curriculum directly using exact word matching or stemmed word matching between the textual metadata of the noisy videos to the targeted concept names. This is the same method as stated in Webly Labeled Learning [123]. Noted that this method only utilizes one modality.

YouTubeTopicAPI The YouTube topic API is utilized to search for videos that are related

to the concept words. The topic API uses textual information of the uploaded videos to obtain related topics of the videos from Freebase. This is the method used in [228].

SearchEngine The curriculum is built using the search result from a text-based search engine [151]. It is similar to related web-search based methods.

Ours We build the curriculum based on the latent topic we learned from the noisy label. We incorporate Latent Dirichlet Allocation (LDA) [16] to determine how each noisy labeled video is related to each target concept. The intuition is that each web video consists of mixtures of topics (concepts), and each topic is characterized by a distribution of words. We impose asymmetric priors over the word distribution so that each learned topic will be seeded with particular words in our target concept. For example, a topic will be seeded with words "make, phone, cases" for the target concept "MakingPhoneCases". we use the online variational inference algorithm from [75]. We then match noisy labels from each modality $\mathbf{z_{im}}$ to the latent topic word distribution using word embedding soft matching [153]. The word embedding is pre-trained using Google News data.

Our method can utilize information from different modalities while common methods like search engine only consider textual information. We compare the performance of different ways of curriculum design by training detectors directly in Section 4.

3.2.3 Algorithm

As proven in recent studies [111, 152], Eq. (3.1) is a biconvex optimization problem. We utilize the alternative convex search algorithm (ACS) [12] to optimize Eq. (3.1) following [91, 111]. Algorithm 1 takes the input of the training set, an instantiated self-paced regularizer and the curriculum constraint function; it outputs an optimal model parameter $\mathbf{w}$. it derives the curriculum region from multi-modal noisy labels $\mathbf{Z} \in \mathbb{R}^{m \times n}$ and forms the curriculum constraint function. Then, it initializes the latent weight variables in the feasible region. In the while loop, the algorithm alternates between two steps until it finally converges: In step 4 given the most recent $\mathbf{v}^*$, the algorithm learns the optimal model parameters; In step 5, we fix the $\mathbf{w}^*$ and the algorithm learns the optimal weights $\mathbf{v}^*$ for each sample. Starting in the beginning, the model grows from learning with easy (less noisy) samples with a small model "age". The model "age" is gradually increased so that the model can incorporate more noisy samples in the training and become more robust over time. Step 4 can be implemented by existing off-the-shelf supervised learning methods such as the Support Vector Machine or back propagation. Gradient-based methods can be used to solve the convex optimization problem in Step 5. According to [65],

the alternative search in Algorithm 1 converges as the objective function is monotonically decreasing and is bounded from below.

Algorithm 1 Multi-modal WEbly-Labeled Learning

input : Input dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{Z}, \tilde{\mathbf{Y}}\}$, self-paced function f and a curriculum constraint function c

output: Model parameter $\mathbf{w}$

Derive curriculum region from $\mathbf{Z} \in \mathbb{R}^{m \times n}$ into $\mathbf{a}, b$

Initialize $\mathbf{v}^*$, λ in the curriculum region

while *not converged* **do**

  Update $\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathbb{E}(\mathbf{w}, \mathbf{v}^*; \lambda, \mathbf{a}, b)$

  Update $\mathbf{v}^* = \arg\min_{\mathbf{v}} \mathbb{E}(\mathbf{w}^*, \mathbf{v}; \lambda, \mathbf{a}, b)$

  if λ *is small* **then** increase λ by the step size;

end

return $\mathbf{w}^*$

3.3 Experimental Results

In this section, we evaluate our method WELL-MM for learning video detectors on noisy labeled data. We first conduct our method on noisy learning in image domain. The efficacy of our methods are mainly verified on two major public benchmarks: FCVID and YFCC100M, where FCVID is by far one of the biggest manually annotated video dataset [95], and the YFCC100M dataset is the largest multimedia benchmark [220].

3.3.1 Experimental Setup

Datasets, Features and Evaluation Metrics Previous studies on noisy learning in image domain have been focusing on noise estimation [209, 243]. We compare our method with them on the synthesized noisy dataset CIFAR-10 generated using code from [243]. We report accuracy on each setting along with the results reported in papers [209, 243] experimented on the same dataset.

Fudan-columbia Video Dataset (FCVID) contains 91,223 YouTube videos (4,232 hours) from 239 categories. It covers a wide range of concepts like activities, objects, scenes, sports, DIY, etc.

Detailed descriptions of the benchmark can be found in [95]. Each video is manually labeled to one or more categories. In our experiments, we do not use the manual labels in training, but instead we automatically generate the web labels according to the concept name appearance in the video metadata. The manual labels are used only in testing to evaluate our and the baseline methods. Following [95], the standard train/test split is used. The second set is YFCC100M [220] which contains about 800,000 videos on Yahoo! Flickr with metadata such as the title, tags, the uploader, etc. There are no manual labels on this set and we automatically generate the curriculum from the metadata in a similar way. Since there are no annotations, we train the concept detectors on the most 101 frequent latent topics found in the metadata. There are totally 47,397 webly labeled videos on the 101 concepts for training.

On FCVID, as the manual labels are available, the performance is evaluated in terms of the precision of the top 5 and 10 ranked videos (P@5 and P@10) and mean Average Precision (mAP) of 239 concepts. On YFCC100M, since there are no manual labels, for evaluation, we apply the detectors to a third public video collection called TRECVID MED which includes 32,000 Internet videos [159]. We apply the detectors trained on YFCC100M to the TRECVID videos and manually annotate the top 10 detected videos returned by each method for 101 concepts.

Implementation Details We build our method on top of a pre-trained convolutional neural network as the low-level features (VGG network [201], except in the image experiment we use AlexNet [109] as in [243]). We extract the key-frame level features and create a video feature by the average pooling. The same features are used across different methods on each dataset. The concept detectors are trained based on a hinge loss cost function by SVM. Algorithm 1 is used to train the concept models iteratively and the λ stops increasing after 100 iterations. At each iteration, we apply a dropout of 0.5 when sampling negative samples. We automatically generate curriculum labels based on the video metadata, ASR, OCR and VGG net 1,000 classification results using latent topic modeling with word embedding matching as shown in Section 3.

Baselines in video domain experiment The proposed method is compared against the following five baseline methods which cover both the classical and the recent representative learning algorithms on webly-labeled data. *BatchTrain* trains a single SVM model using all samples in the multi-modal curriculum built with our method as described in section 3.2.2. *Self-Paced Learning (SPL)* is a classical method where the curriculum is generated by the learner itself [111]. *BabyLearning* is a recent method that simulates baby learning by starting with few training samples and fine-tuning using more weakly labeled videos crawled from the search engine [133]. *GoogleHNM* is a hard negative mining method proposed by Google [228]. It

utilizes hard negative mining to train a second order mixture of experts model according to the video's YouTube topics. *FastImage* [69] is a video retrieval method that utilizes web images from search engine to match to the video with re-ranking. *WELL-MM* is the proposed method. The hyper-parameters of all methods including the baseline methods are tuned on the same validation set. On FCVID, the set is a standard development set with manual labels randomly selected from 10% of the training set (No training was done using ground truth labels) whereas on YFCC100M it is also a 10% proportion of noisy training set.

3.3.2 Experiments on FCVID

Curriculum Comparison As discussed in Section 3.2.2, we compare different ways to build curriculum for noisy label learning. Here we also compare their effectiveness by training concept detectors directly using the curriculum labels. The batch train model is used for all generated curriculum labels. In Table 3.1 we show the batch trained models' precision at 5, 10 and mean average precision on the test set of FCVID. For WELL-MM, we extract curriculum from different modalities as shown in Section 3.2.2, and combine them using linear weights. The weights are hyper-parameters that are tuned on the validation set, and the optimal weights for textual metadata, ASR, image classification and OCR results are 1.0, 0.5, 0.5 and 0.05, respectively. This attempt to combining curriculum from different modalities serves as a pilot study. However, experiments show that such simple linear weighting is already effective with WELL-MM. Further research in this direction is left for future work. We also compare WELL-MM with using only latent topic modeling and word embedding soft matching. Results show that the curriculum generated by combining latent topic modeling and word embedding using multi-modal prior knowledge is the most accurate, which indicates our claim of exploiting multi-modal information is beneficial.

Baseline Comparison Table 3.2 compares the precision and mAP of different methods where the best results are highlighted. As we see, the proposed WELL-MM significantly outperforms all baseline methods, with statistically significant difference at p-level of 0.05. Comparing SPL with BatchTrain, it shows that the self-paced learning model over-fits to the noise without prior knowledge and performs worse than the simple BatchTrain model. Comparing WELL-MM with SPL and BatchTrain, the effect of incorporating multi-modal curriculum makes a significant difference in terms of performance, which suggests the importance of prior knowledge and preventing over-fitting in webly learning. The promising experimental results substantiate the efficacy of the proposed method.

Table 3.1: Comparison of different curriculum using the BatchTrain learning method.

Method	P@5	P@10	mAP
WordHardMatching	0.782	0.763	0.469
YouTubeTopicAPI	0.587	0.563	0.315
SearchEngine	0.723	0.713	0.413
WordEmbedding	0.790	0.774	0.462
LatentTopic	0.731	0.716	0.409
WELL-MM	**0.838**	**0.820**	**0.486**

Table 3.2: Baseline comparison on FCVID

Method	P@5	P@10	mAP
BatchTrain	0.838	0.820	0.486
FastImage [69]	-	-	0.284
SPL [112]	0.793	0.754	0.414
GoogleHNM [228]	0.781	0.757	0.472
BabyLearning [133]	0.834	0.817	0.496
WELL-MM	**0.918**	**0.906**	**0.615**

Robustness to Noise Comparison In this comparison we manually control the noise level of the curriculum in order to systematically verify how our methods would perform with respect to the noise level within the web data. To this end, we randomly select video samples with ground truth labels for each concept, so that the noise level of the curriculum labels are set at 20%, 40%, 60%, 80% and we fix the recall of all the labels. We then train WELL-MM using such curriculum and test them on the FCVID testing set. We also compare WELL-MM to three other methods with the same curriculum, among them *GoogleHNM* is a recent method to train video concept detector with large-scale data. We exclude *BabyLearning*, which relies on the returned results by the search engine, since in this experiment the curriculum is fixed. As shown in Table 3.3, as the noise level of the curriculum grows, WELL-MM maintains its performance while other methods drop significantly. Specifically, when the noise level of curriculum increased from 60% to 80%, other methods' mAP drops 46.5% on average while WELL-MM's mAP only drops 19.1% relatively. It shows that WELL-MM is robust against different level of noise, which shows great potential in larger scale webly-labeled learning as the dataset gets

bigger, the noisier it may become.

Table 3.3: WELL-MM performance with curriculum consisting of multiple artificial noise levels.

Method \ Noise Level	20%	40%	60%	80%
BatchTrain	0.592	0.538	0.463	0.232
SPL	0.586	0.515	0.396	0.184
GoogleHNM	0.602	0.552	0.477	0.304
WELL-MM	**0.673**	**0.646**	**0.613**	**0.496**

Noisy Dataset Size Comparison To investigate the potential of concept learning on webly-labeled video data, we apply the methods on different sizes of subsets of the data. Specifically, we randomly split the FCVID training set into several subsets of 200, 500, 1,000, and 2,000 hours of videos, and train the models on each subset without using manual annotations. The models are then tested on the same test set. Table 3.4 lists the average results of each type of subsets. As we see, the accuracy of WELL-MM on webly-labeled data increases along with the growth of the size of noisy data while other webly learning methods' performance tend to be saturated.

Comparing to the methods trained using ground truth, In Table 3.4, WELL-MM trained using the whole dataset (2000 hours) outperforms Static CNN (trained using manual labels) using around 1400 hours of data and rDNN-F (trained using manual labels with three features) trained using around 450h of data. And since the incremental performance increase of WELL-MM is close to linear, we conclude that with sufficient webly-labeled videos (which are not hard to obtain) WELL-MM will be able to outperform the rDNN-F trained using 2000h of data, which is currently the largest manual labeled dataset.

3.3.3 Experiments on CIFAR-10

Following [243], we generate synthesized noisy training data with a noise level of 30%, 40% and 50% on CIFAR-10 dataset. The models are trained on noisy data and tested on clean data. Classification Accuracy is reported. Our method doesn't assume any kind of noise distribution, while Noisy-CNN [209] assumes the noise distribution depends on classes and Massive-Learning [243] assumes it also depends on the image content. We show the experimental results in Table 3.5. The results show that WELL-MM outperforms the other methods at all noise levels. More interestingly, as the noise level rises from 30% to 50%, the performance of Massive-

Table 3.4: MAP comparison of models trained using web labels and ground-truth labels on different subsets of FCVID. The methods marked by * are trained using human annotated labels.

Method \ Dataset Size	200h	500h	1000h	2000h
BatchTrain	0.364	0.422	0.452	0.486
SPL [112]	0.327	0.379	0.403	0.414
GoogleHNM [228]	0.361	0.421	0.451	0.472
BabyLearning [133]	0.390	0.447	0.481	0.496
WELL-MM	0.487	0.554	0.595	0.615
Static CNN[95]*	0.485	0.561	0.604	0.638
rDNN-F[95]*	0.550	0.620	0.650	0.754

Table 3.5: Experimental results on CIFAR-10

Methods \ Noise Level	30%	40%	50%
Noisy-CNN [209]	0.697	0.667	0.634
Massive-Learning [243]	0.698	0.668	0.630
WELL-MM	**0.709**	**0.700**	**0.682**

Learning [243] drops about 9.8%, while WELL-MM only drops 3.8%. It shows that WELL-MM can also effectively learn robust concept detectors in image domain.

3.3.4 Experiments on YFCC100M

In the experiments on YFCC100M, we train 101 concept detectors on YFCC100M and test them on the TRECVID MED dataset which includes 32,000 Internet videos. Since there are no manual labels, to evaluate the performance, we manually annotate the top 10 videos in the test set and report their precisions in Table 3.6. The MED evaluation is done by four annotators and the final results are averaged from all annotations. The Fleiss' Kappa value for these four annotators is 0.64. A similar pattern can be observed where the comparisons substantiate the rationality of the proposed webly learning framework. Besides, the promising results on the largest multimedia set YFCC100M verify the scalability of the proposed method.

Table 3.6: Baseline comparison on YFCC100M

Method	P@3	P@5	P@10
BatchTrain	0.535	0.513	0.487
SPL [112]	0.485	0.463	0.454
GoogleHNM [228]	0.541	0.525	0.500
BabyLearning [133]	0.548	0.519	0.466
WELL-MM	**0.667**	**0.663**	**0.649**

Figure 3.3: Illustration of representative videos selected by WELL-MM at different iterations

3.3.5 Qualitative Analysis

In this section we show training examples of WELL-MM. In Fig. 3.3, we demonstrate the positive samples that WELL select at different stage of training the concept "baseball" and "birthday". For the concept "baseball", at the early stage (1/93, 25/93), WELL-MM selects easier and clearer samples such as the ones with camera directly pointing at the playground, while at the later stage (75/93, 93/93) WELL-MM starts to train with harder samples with different lighting conditions and untypical samples for the concept. For the concept "birthday", as we see, at later stage of the training, complex samples for birthday event like a video with two girl singing birthday song (75/84) and a video of celebrating birthday during hiking (84/84) are included in the training, while at the early stage, only typical "birthday" videos with birthday cake and candles are included.

3.4 Related Work

Curriculum and Self-paced Learning: Recently a learning paradigm called *curriculum learning* (CL) was proposed by Bengio et al., in which a model is learned by gradually incorporating

from easy to complex samples in training so as to increase the entropy of training samples [13]. A curriculum determines a sequence of training samples and is often derived by predetermined heuristics in particular problems. For example, Chen et al. designed a curriculum where images with clean backgrounds are learned before the images with noisy backgrounds [33], i.e. their method first builds a feature representation by a Convolutional Neural Network (CNN) on images with clean background and then they fine tune the models on images with noisy background. In [205], the authors approached grammar induction, where the curriculum is derived in terms of the length of a sentence. Because the number of possible solutions grows exponentially with the length of the sentence, and short sentences are easier and thus should be learned earlier.

The heuristic knowledge in a problem often proves to be useful. However, the curriculum design may lead to inconsistency between the fixed curriculum and the dynamically learned models. That is, the curriculum is predetermined a prior and cannot be adjusted accordingly, taking into account the feedback about the learner. To alleviate the issue of CL, Kumar et al. designed a learning paradigm, called *self-paced learning* (SPL) [111]. SPL embeds curriculum design as a regularizer into the learning objective. Compared with CL, SPL exhibits two advantages: first, it jointly optimizes the learning objective with the curriculum, and thus the curriculum and the learned model are consistent under the same optimization problem; second, the learning is controlled by a regularizer which is independent of the loss function in specific problems. This theory has been successfully applied to various applications, such as action/event detection [90], domain adaption [216], tracking [214] and segmentation [112], reranking [89], etc.

Learning Detectors in Web Data: Many recent studies have been proposed to utilize a large amount of noisy data from the Internet. For example, [154] proposed a Never-Ending Language Learning (NELL) paradigm and built adaptive learners that make use of the web data by learning different types of knowledge and beliefs continuously. In the image domain, existing methods try to tackle the problem of constructing qualified training sets based on the search results of text or image search engines [34, 49, 133, 243]. For example, NEIL [34] followed the idea of NELL and learned from web images to form a large collection of concept detectors iteratively via a semi-supervised fashion. By combining the classifiers and the inter-concept relationships it learned, NEIL can be used for scene classification and object detection task. [133] presented a weakly-supervised method called Baby Learning for object detection from a few training images and videos. They first embed the prior knowledge into a pre-trained CNN. When given very few samples for a new concept, a simple detector is constructed to discover much more train-

ing instances from the online weakly labeled videos. As more training samples are selected, the concept detector keeps refining until a mature detector is formed. Another recent work in image domain [33] proposed a webly supervised learning of Convolutional Neural Network. They utilized easy images from search engine like Google to bootstrap a first-stage network and then used noisier images from photo-sharing websites like Flickr to train an enhanced model.

In video domain, only a few studies [51, 69, 228] have been proposed for noisy data learning since training robust video concept detectors is more challenging than the problem in the image domain. [51] tackled visual event detection problem by using SVM based domain adaptation method in web video data. [69] described a fast automatic video retrieval method using web images. Given a targeted concept, compact representations of web images obtained from search engines like Google, Flickr are calculated and matched to compact features of videos. Such method can be utilized without any pre-defined concepts. [228] discussed a method that exploits the YouTube API to train large-scale video concept detectors on YouTube. The method utilized a calibration process and hard negative mining to train a second order mixture of experts model in order to discover correlations within the labels.

3.5 Summary

In this chapter, we propose a novel method called WELL-MM for webly labeled video data learning. WELL-MM extracts multi-modal informative knowledge from noisy weakly labeled video data from the web through a general framework and achieves the best performance only using webly-labeled data on two major video datasets. The comprehensive experimental results demonstrate that WELL-MM outperforms state-of-the-art studies by a statically significant margin on learning concepts from noisy web video data. In addition, the results also verify that WELL-MM is robust to the level of noisiness in the video data. The result suggests that with more webly-labeled data, which is not hard to obtain, WELL-MM can potentially outperform models trained on any existing manually-labeled data.

Chapter 4

Spatial-Temporal Alignment Network for Action Recognition and Detection

In this chapter, we explore viewpoint-invariant feature representations that aim to have better generalization abilities for action recognition and detection. We propose a novel Spatial-Temporal Alignment Network (*STAN*) that aims to learn geometric invariant representations for action recognition and action detection.

4.1 Motivation

Human vision can recognize video actions efficiently despite the variations of viewpoints.

Convolutional neural networks (CNNs) [21, 55, 199, 218, 223] fully utilize the power of GPUs/TPUs and employ spatial-temporal filters to recognize actions, which outperforms traditional models including oriented filtering in space time (HOG3D) [104], spatial-temporal interest points [115], motion history images [221], and trajectories [233]. However, due to the high variations in space-time, the state of the art of action recognition is still far from satisfactory, compared with the success of 2D CNNs in image recognition [70].

A key challenge of action recognition is to capture the variations across space and time. Since CNN assumes the filters share weights at different locations, it can not explicitly model the viewpoint changes and other variations. To solve such limitations, a practical way is to expand feature representations to accommodate a higher degree of freedom. For example, the two-stream network [202] proposes to integrate optical flow with RGB features. More recently, SlowFast [56] combines both slow and fast pathways to learn different temporal information, and obtain good performance. However, such feature expanding approaches quickly lead to

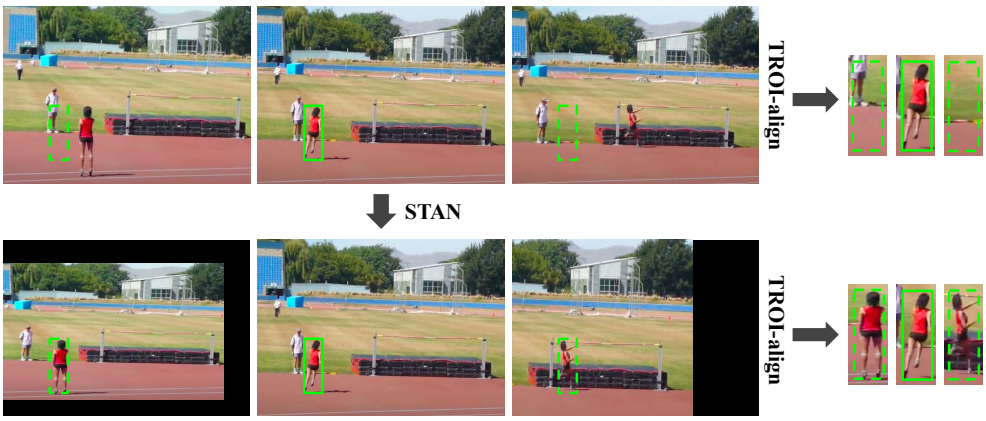

Figure 4.1: An illustrative example of how *STAN* can address the bounding box misalignment issue in spatial temporal localization. In prior work [56, 66], the detected person box (solid line) is expanded along the temporal axis (dotted line) to form a 3D cube for the subsequent temporal ROI-align (TROI-align). If the actor's movement is large across frames as shown in the top row (or there are substantial camera movements), the TROI-aligned features will be noisy with the majority being background pixels. *STAN* learns a spatial temporal transformation that puts feature maps in the same coordinate system and in the bottom row example it is the central frame.

cumbersome, high-dimensional feature maps, which not only make the computation more expensive but also miss the geometric interpretation of the subjects. In this chapter we show that our method can improve on both simpler backbone networks like ResNet3D and more complex ones like SlowFast, with minimal computation overhead.

We proposes a different approach to capture the variation in action recognition. Instead of relying on stacking deeper CNN layers, we aims to explicitly learn geometric transformations and viewpoint invariant features. Our idea is motivated by [107], which believes that human vision relies on coordinate frames. However, the stacked capsule autoencoder in [107] is designed for 2D images, and too expensive for large scale visual recognition.

Fig. 4.1 illustrates the importance of alignment in the problem of spatial temporal localization. Following the existing action detection pipeline [56, 66], the action representation is obtained by first cropping a 3D cube within the spatial-temporal feature maps centered at the detected actor's bounding box, followed by a Max Pooling operation. Without alignment, the representation is contaminated by background pixels as shown in the top row in Fig. 4.1. With alignment the representation as shown in the bottom row is much more focused on the actor.

In this chapter, we propose a Spatial Temporal Alignment Network (*STAN*) that aims to

learn viewpoint invariant representations for action recognition and action detection. The *STAN* block is very light-weighted and generic, which could be plugged into existing action recognition models like ResNet3D, Non-Local Network [237] and the SlowFast network [56]. As discussed in Section 4.2.5, we insert a *STAN* block between res_2 and res_3 in the ResNet3D architecture, and add it at the same location on the Fast pathway in the SlowFast model. For the SlowFast + *STAN* model, the FLOPS increase is only **relatively 2.1%** (134.5 GFLOPS vs. 131.7 GFLOPS) for action recognition on Charades, but we achieve **5% relative** improvement on mean average precision.

The contribution of this chapter is three-fold: (1) To the best of our knowledge, this is the first work to explore explicit spatial-temporal alignment in 3D CNNs for action detection. (2) Our *STAN* requires very low extra FLOPS in addition to the backbone network. (3) Extensive experiments on different datasets suggest the model using *STAN* can outperform the state of the art.

4.2 The *STAN* Model

In this section, we describe our spatial-temporal alignment network for action recognition and detection, which we call *STAN*. Motivated by the previous works in image understanding [79, 84, 134], our work considers a generalized model in video domain, so that it can handle dynamic viewpoint changes in action recognition and detection. The key idea of *STAN* is to utilize spatial-temporal alignment for feature maps to account for viewpoint changes and actor movements in the videos. The learned affine transformations (Eqn. 4.2) at each temporal group can achieve effects including camera stabilization (See Fig. 4.4), which is important in Internet videos to counter camera motions. In general, given an input spatial-temporal feature map $\mathcal{I}_{in} \in \mathbf{R}^{C\times T\times H\times W}$ where H stands for height, W for width, T for time and C for channels, the alignment function is defined as

$$\text{STAN}(\mathcal{I}_{in}) = \mathcal{T}(\theta, \mathcal{I}_{in}) + \mathcal{I}_{in}, \text{where } \theta = \mathcal{D}(\mathcal{I}_{in}) \tag{4.1}$$

The output of *STAN* function is of the same dimension as $\mathcal{I}_{in}$. Function $\mathcal{D}$ represents the deformation network, where the feature map alignment parameter $\theta \in \mathbf{R}^{4\times 4}$ is computed based on the input feature map. Function $\mathcal{D}$ can be in the form of a simple feed-forward network using spatial-temporal features [224]. Function $\mathcal{T}$ is defined as the warping function, where input feature maps are warped based on the alignment parameter. We add a residual connection between the input feature maps and output feature maps for faster training and avoiding

Layer	Filter size	Output channels
conv	1×7^2	$C//4$
conv	7×1^2	$C//2$
maxpool	4×7^2 stride 4×7^2	$C//2$
conv	7×7^2	C
globalpool		C
fc		variable

Table 4.1: Deformation network architecture used in our experiments. The filter sizes are $T \times S^2$ where T is the temporal kernel size and S is the spatial size. The global pool layer averages features across all spatial-temporal locations. "fc" is the final fully-connected layer.

boundary effect described in [134]. The *STAN* layer can be added to different locations of the backbone to account for the alignment needs for different level of feature maps.

4.2.1 Network Architecture

The overall *STAN* architecture is shown in Fig. 4.4. Our model uses a 3D CNN as the backbone network to extract spatial-temporal feature maps from video frames. In addition, *STAN* has the following key components:

Deformation Network produces the alignment / deformation parameter θ in Eqn. 4.1.

Warping Module samples from the input feature maps based on θ and outputs the final transformed feature maps.

Temporal Grouping learns a separate alignment for different temporal segments within the video clip, similar to multihead attention [230] and correlation networks [235].

In the following, we will introduce the above modules in details.

4.2.2 Deformation Network

The deformation network $\mathcal{D}$ produces a transformation parameter, $\theta \in \mathbf{R}^{4\times4}$. Our network is based on R(2+1)D [224] although other options are possible, such as a simple feed-forward network, or a recurrent network and compositional function as proposed in [134]. Suppose the number of input channels is C, the details of our network architecture is presented in Table 4.1. All convolution layers are followed by batch normalization [83] and ReLU [62, 68]. The dimension of the final FC layer depends on the type of parameterization we choose for the spatial-temporal alignment. Taking affine transformation for example, the dimension of the

network output ($\mathbf{p}_{\text{affine}} = [p_1 \ ... \ p_{12}]^T$) is of size 12 and θ is constructed as

$$\theta(\mathbf{p}_{\text{affine}}) = \begin{pmatrix} 1+p_1 & p_2 & p_3 & p_4 \\ p_5 & 1+p_6 & p_7 & p_8 \\ p_9 & p_{10} & 1+p_{11} & p_{12} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.2}$$

θ can be more restrictive as in the case of attention [249], where cropping, translation and scaling are allowed for transformation, the dimension of the network output $\mathbf{p}_{\text{att}} = [p_1, ...p_6]^T$ is a vector of size 6 and θ is constructed as

$$\theta(\mathbf{p}_{\text{att}}) = \begin{pmatrix} 1+p_1 & 0 & 0 & p_4 \\ 0 & 1+p_2 & 0 & p_5 \\ 0 & 0 & 1+p_3 & p_6 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.3}$$

The design of *STAN* is flexible and can be any type of transformation. The key intuition of the deformation on CNN feature maps is to compensate for the fact that CNNs are not rotation, scale, and shear transformation equivariant [107].

4.2.3 Warping Module

After computing the transformation matrix θ, we utilize a differentiable warping function $\mathcal{T}$ to transform the feature maps with better alignment of the content. See Fig. 4.2. The warping function is essentially a resampling of features from the input feature maps to the output at each corresponding pixel location. Note that the feature maps could also be images. Extending from the notation of 2D alignment [84] , we define the output feature maps $\mathcal{I} \in \mathbf{R}^{C \times T \times H \times W}$ to lie on a spatial-temporal regular grid $G = \{G_i\}$, where each element of the grid $G_i = (t_i^o, x_i^o, y_i^o)$ corresponds to a vector of output features of size $\mathbf{R}^C$. Hence, the pointwise sampling between the input and output feature maps is written as

$$\begin{bmatrix} t_i^s \\ x_i^s \\ y_i^s \\ 1 \end{bmatrix} = \mathcal{T}_g(\theta, G_i) = \begin{pmatrix} 1+p_1 & 0 & 0 & p_4 \\ 0 & 1+p_2 & 0 & p_5 \\ 0 & 0 & 1+p_3 & p_6 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} t_i^o \\ x_i^o \\ y_i^o \\ 1 \end{bmatrix} \tag{4.4}$$

where (t_i^o, x_i^o, y_i^o) are the output feature map coordinates in the regular grid and (t_i^s, x_i^s, y_i^s) are the corresponding input feature map coordinates for feature sampling. Here we use attention transformation as an example, where the deformation matrix is θ parameterized by

$\mathbf{p}_{\text{att}}$ (Eqn. 4.3). Given the coordinates mapping, as the computed corresponding coordinates in the input feature maps might not be integers, we utilize the differentiable trilinear interpolation to sample input features from the eight closest points based on their distance to the computed point (t_i^s, x_i^s, y_i^s). In this way, we iterate through every point in the regular grid, $(t_i^o \in [1, ..., T], x_i^o \in [1, ..., W], y_i^o \in [1, ..., H])$ and compute the output feature maps identically for each channel.

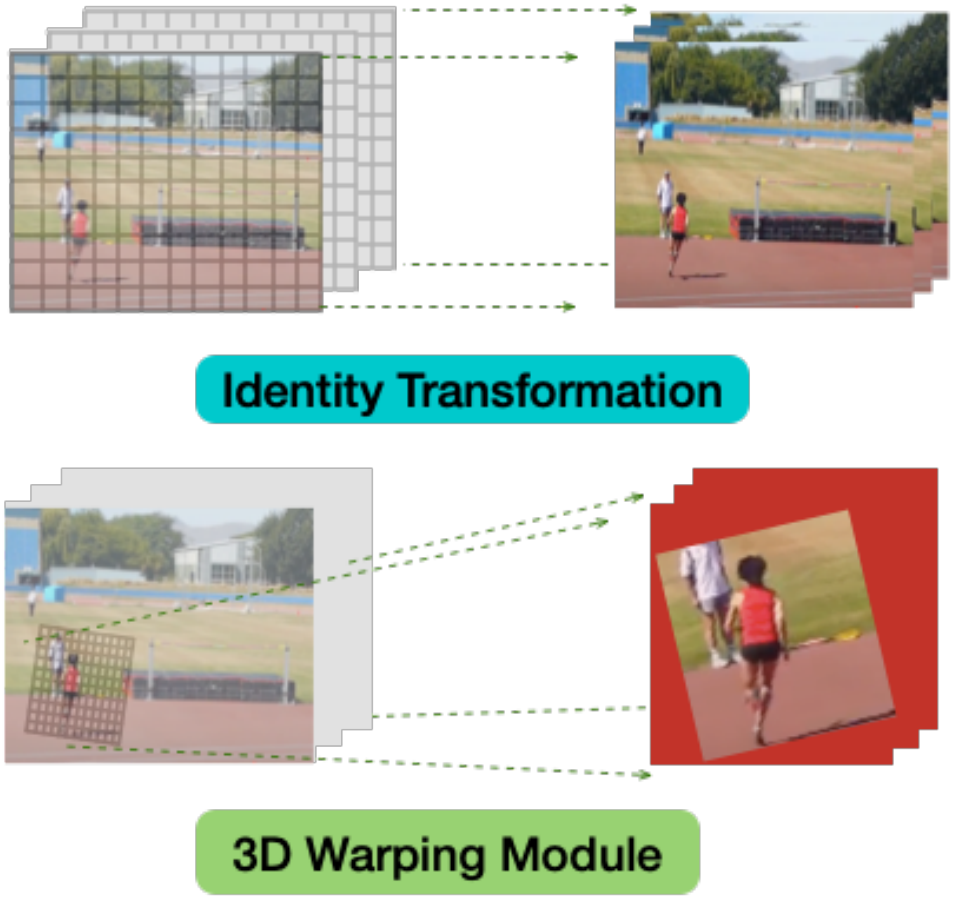

Figure 4.2: Two examples of applying spatial-temporal warping/transform to an input sequence. The top row shows an identity transformation while the bottom row shows an example 3D warping where the actor is rotated and zoomed in.

4.2.4 Temporal Grouping

Many actions are composed of a sequence of sub-actions that require different alignments. For example, action "High Jump" may consist of "stand", "run" and "jump" (see Fig. 4.4), and the

actor may have moved around between video frames. Based on this observation, we propose temporal grouping to allow the model to learn different alignments at different time periods. It is written as

$$\text{STAN}_{tg}(\mathcal{I}_{in}) = \text{concat}\left(\text{STAN}(\mathcal{I}_{in}^{1}), ..., \text{STAN}(\mathcal{I}_{in}^{T_g})\right) \tag{4.5}$$

where T_g is the number of temporal groups and concatenation is operated on the temporal axis. Each group alignment $\text{STAN}(\mathcal{I}_{in}^{T_g})$ is of size $\mathbf{R}^{C\times T//T_g\times H\times W}$.

4.2.5 Integration with Backbone CNNs

Now that we have defined *STAN* function we now discuss effective ways to add it into existing backbone CNN networks. In this chapter, we focus on designing *STAN* for the RGB stream and the extension to optical flow will be left as one of our future work. Recent works [56, 237] and their variants [160] achieve single-stream state-of-the-art performance on action recognition and detection with 3D CNNs. We therefore explore adding *STAN* layers to the ResNet3D model [56, 237] and the SlowFast [56] network. Intuitively, placing *STAN* layer at shallower layers allows earlier feature alignments that could potentially lead to cleaner representation for better action recognition. However, shallow layers may not have enough abstraction in the feature maps for *STAN* to learn the right alignments. Based on this trade-off, we experiment with adding one *STAN* layer after res_2 block. For SlowFast, we only add *STAN* layers on the Fast pathway since it contains more temporal information compared to the Slow pathway. Other insertion locations are explored in the ablation experiments (Section 4.3.1).

4.2.6 Detection Architecture

For action recognition, the final CNN outputs are passed through global averaging (and a concatenation for SlowFast) and a fully-connected layer to get the action class probabilities. For action detection, following previous works [56, 66, 210], we use pre-computed actor proposals. The bounding boxes are used to extract region-of-interest (RoI) features using RoI-Align [71] at the last feature map of "res_5" after temporal global average pooling. Our *STAN* can fix the actor misalignment problem by aligning the 2D RoIs along the temporal axis. Finally, the RoI features are then max-pooled and fed to a fully-connected layer.

4.3 Experiments

To demonstrate the efficacy of our models, specifically, the viewpoint-invariant design that helps action models to generalize, we experiment on two recent action detection datasets, including AVA [66] and AVA-Kinetics [119], and several major action recognition datasets, Kinetics-400 [102], Charades [198], Charades-Ego [200], MEVA [40]. In the experiments, we aim to showcase that STAN can achieve significant performance improvement with minimal computation overhead.

4.3.1 Action Detection

This section evaluates our *STAN* model for the task of spatial temporal localization for actions using AVA and AVA-Kinetics. We show significant improvement over baselines while only adding a fraction of computation.

Datasets. The AVA dataset [66] is an action detection dataset where models are required to output action classification results with bounding boxes. Spatial-temporal labels are provided for one frame per second, with every person annotated with a bounding box and (possibly multiple) actions. There are 211k training and 57k validation video clips. We use AVA v2.1 and follow the standard protocol [56, 66] of evaluating on 60 classes. The performance metric is mean Average Precision (mAP) over 60 classes, using an IoU threshold of 0.5. We report mAP on the official validation set.

The AVA-Kinetics dataset [119] is a recent action detection dataset which follows AVA annotation protocol to annotate relevant videos on the Kinetics-700 [22] dataset. We utilize the part where all videos are from Kinetics for training and testing. There are about 141k training videos and 32k validation videos. The AVA-Kinetics also contains the same 60 classes for evaluation and we utilize the mAP metric. Following [119], we evaluate action detection model performance with both ground truth person boxes and pre-computed person boxes. We report mAP on the official validation set as well.

Action Proposals. The action detection models have to output action/person bounding boxes, and only predicted boxes with IoU (intersection over union) area w.r.t the ground truth boxes above a threshold of 0.5 are considered true positives. As mentioned in Section 4.2.6, we follow previous works [56, 66, 210] and use pre-computed person boxes as action proposals. For AVA, we utilize the same person proposals from [56]. For AVA-Kinetics, we finetune a Mask-RCNN [71] with ResNet-101 backbone trained on COCO [136] on the person boxes in the train-

ing set and extract boxes as described in Section 4.2.6. The region proposals for action detection are detected person boxes with a confidence score of larger than 0.8, which has a recall of 83.1% and a precision of 62.1% for the person class, given IoU threshold of 0.5. The average precision of the person class is 0.732 on the validation set.

Training. We initialize the network weights from the Kinetics-400 classification models, following previous works [56]. We use a learning rate of 0.1 and cosine learning rate decay. We use synchronized SGD to train for 10 epochs with a batch size of 16 on a 4-GPU machine, with a linear warm-up from 0.000125 for the first 2 epochs. We use a weight decay of 10^{-7}. Ground-truth boxes and video clips centered at the annotated key frames are sampled for training. The video is first resized to 256x320, and then we use random 224×224 crops and horizontal flipping following [56].

Inference. Since the annotations on AVA and AVA-Kinetics are one (key) frame per second, we sample a single video clip temporally centered around the key frame for evaluation. Following [56], we resize the spatial dimension such that its shorter side is 256 pixels. Ground truth boxes or pre-computed boxes are used as inputs. We report the inference time computational cost (in FLOPs) of a single 256x320 clip using Tensorflow's Profiler.

Implementation Details. We add one *STAN* layer to the backbone CNN network as described in Section 4.2.5. We use affine transformation (Eqn. 4.2) and the deformation network is defined in Table 4.1. We use a temporal group of 2 and the number of base convolution filters in the deformation network is capped at 8 for ResNet3D to keep the FLOPs low. For SlowFast, we set this number as described in Table 4.1.

Baselines. To demonstrate the effectiveness of our proposed model, we experiment with recent 3D-CNN based models for action recognition and detection. **ResNet3D** is a model based on ResNet-50 [70] with additional 3D convolutional filters.

The number of input frames is 8 and we sample 1 frame every 8 frames (i.e., 8x8 frames). **ResNet3D + *STAN*** is our proposed model added to ResNet3D as described in Section 4.2.5 with the same inputs. **SlowFast** is a recent efficient model [56] with a Slow pathway and a Fast pathway, which takes 8x8 and 32x2 frames as inputs respectively. We use ResNet-50 backbone for SlowFast as well. If we use only the Slow pathway, the model will become the same as ResNet3D. **SlowFast + *STAN*** is our proposed model added to SlowFast as described in Section 4.2.5.

Models	mAP	GFLOPs	MParams
ResNet3D (8x8)	0.234	208.0	31.75
ResNet3D + *STAN*	**0.247**	216.6	32.02
SlowFast [56] (32x2)	0.252	242.6	33.77
SlowFast + *STAN*	**0.268**	247.4	33.96

Table 4.2: Experiment results on AVA dataset. We show the model performance on mAP, computation cost (in billions) and number of parameters (in millions). The models are based on ResNet-50 backbone. The computational cost is of a single 256x320 video clip of length 8x8 or 32x2 (number of frames x sampling rate) frames.

Models	mAP	GFLOPs
Action Transformer [119]	0.337 / 0.191	-
ResNet3D (8x8)	0.315 / 0.224	208.0
ResNet3D + *STAN*	**0.336** / **0.238**	216.6
SlowFast [56] (32x2)	0.341 / 0.242	242.6
SlowFast + *STAN*	**0.358** / **0.253**	247.4

Table 4.3: Experiment results on AVA-Kinetics dataset. We show both mAP with ground truth boxes and detected boxes. We are not able to get Action Transformer's FLOPs as the code is not public. Note that the pre-computed boxes for Action Transformer is different from ours.

Main Results

AVA. Table 4.2 shows the results on AVA dataset [66]. We follow the SlowFast [56] paper's evaluation protocol and use the same predicted person bounding boxes provided by the authors with ROIAlign to classify actions. For both ResNet3D and SlowFast, we use ResNet-50 as their base architecture. Compared with baseline methods, our model is able to achieve significant improvement with minimal computation overhead (about 2-4% more). The parameter increase is also minimal.

AVA-Kinetics is a recent action detection dataset with Internet videos from the Kinetics-700 dataset [22], which has more moving cameras and objects with smaller bounding boxes than AVA. Table 4.3 shows the results on AVA-Kinetics dataset [119]. We follow the baseline [119] paper's evaluation protocol and experiment with both ground truth person boxes and detected person boxes from the described finetuned Mask-RCNN model. Our model is able to achieve

	Diff	mAP	GFLOPs
SlowFast	-	0.252	242.55
+ *STAN*	**+1.6%**	**0.268**	247.40
+ *STAN* ($pool_1$)	-	0.253	242.88
+ *STAN* (res_3)	+0.7%	0.259	247.35
+ *STAN* (res_4)	+0.3%	0.255	247.39
+ *STAN* (Att, 6)	+0.7%	0.259	247.40
+ *STAN* (H, 15)	+0.3%	0.255	247.40
+ *STAN* (no tg)	+0.8%	0.260	247.40
+ *STAN* (tg=#frames)	-	0.254	246.16
+ *STAN* (fixed W_θ)	+1.2%	0.264	247.40

Table 4.4: Ablation experiment results (on AVA). The computational cost is of a single 256x320 video clip of length 32x2 (number of frames x sampling rate) frames. The "Diff" column shows absolute improvement of the variant models compared to the baseline SlowFast model. "(Att, 6)" means attention transformation with 6 degree-of-freedom (DoF), and "(H, 15)" means homography transformation with 15 DoF. "tg" means temporal grouping. See text for details.

2.1% absolute improvement on mAP for ResNet3D with only relatively 4.1% more computation and 1.7% improvement with 2% more computation on SlowFast. These results show the consistently significant improvement of STAN.

Ablation Experiments

In this section, we perform ablation studies on the AVA dataset with the SlowFast model as the backbone network. We run each ablation experiments two times and show the averaged results. The differences between the same run are within 0.1%. To understand how action models can benefit from *STAN*, we explore the following questions (results are shown in Table 4.4):

Where to insert *STAN* layer? In CNN networks, shallower layers tend to encode low-level visual features like edges and patterns while deeper layers may contain more abstract information. Placing *STAN* layer at shallower layers allows earlier feature alignments that could potentially lead to cleaner representation. To verify this hypothesis, we experiment with adding one *STAN* layer at deeper layers than res_2 block and before. We first try adding *STAN* layer after res_3 and res_4. As we see, the model performance deteriorates significantly. We then try adding STAN layer after $pool_1$ (before res_2). There is virtually no improvement and the addi-

tion of FLOPS is low due to the low number of features channels at that location. We suspect that STAN at earlier layer struggles to understand the visual content hence it is unable to produce meaningful alignment. We find that right after res_2 seems to be a good sweet spot. We have also experimented with multiple insertions of STAN layers but it only leads to marginal improvement.

What is the best parameterization for the deformation network? In Section 4.2.2, we have discussed two ways of parameterization for the deformation network, affine transformation (Eqn. 4.2) and attention transformation (Eqn. 4.3). Each has 12 and 6 degree-of-freedom (DoF), respectively. We use affine transformation in our main experiments and here we experiment with attention transformation and homography transformation. For homography transformation, the last element in the 4×4 matrix is set as 1 hence the DoF is 15, which is shown in Table 4.4 ("H, 15"). Results are shown in Table 4.4. As we see, the model with the most free parameters does not necessarily lead to better performance.

Does temporal grouping help? In this experiment, we validate the efficacy of temporal grouping (Section 4.2.4). The main experiments are conducted with a temporal group of 2, and the model performance drops by a big margin if temporal grouping is removed. We then try applying a temporal group of 32, which is the same as the number of video frames, to see whether having individual transformations on every frame would help. As we see, the computation cost drops slightly and no improvement over the baseline.

Does *STAN* transfer well? Finally, we conduct an experiment to see whether the deformation network learned from a dataset can be generalized to another. We train the original *STAN* model on Kinetics-400 dataset, and then only fine-tune the layers after the *STAN* layer on AVA. As we see in Table 4.4 ("fixed W_θ"), *STAN* can still achieve reasonable improvement on AVA, suggesting the deformation can be transferred from one dataset to another.

4.3.2 Action Recognition

The action recognition task is defined to be a classification task given a trimmed video clip. To evaluate the generalization abilities of our proposed model, we consider three major datasets, Kinetics-400 [102], Charades [198] and Charades-Ego [200].

Datasets. Kinetics-400 [102] consists of about 240k training videos and 20k validation videos in 400 human action classes. The videos are about 10 seconds long. Following previous works [56, 237], we report top-1 and top-5 classification accuracy. Charades [198] is a dataset with longer (about 30 seconds on average) videos of indoor activities. There are about 9.8k training videos

and 1.8k validation videos in 157 classes in a multi-label classification setting. Charades-Ego [200] has the same 157 action labels but consists of both third-person view and first-person view videos. Essentially, this dataset shows different perspectives of the same actions. MEVA [40] is a multi-view dataset with actions of 35 classes. Many action instances are captured by multiple synchronized cameras, which makes it ideal to test our proposed method for viewpoint-invariant representations. Performance is measured in mean Average Precision (mAP).

Training. Our models on Kinetics are trained from scratch with random initialization, without using any pre-training (same as in [56]). We use a learning rate of 0.2 and cosine learning rate decay. We use synchronized SGD to train for 100 epochs with a batch size of 16 on a 4-GPU machine, with a linear warm-up from 0.01 for the first 20 epochs. We use a weight decay of 10^{-7}. On Charades, Charades-Ego and MEVA, we initialize the models using models trained on Kinetics-400. We use a learning rate of 0.02 and train for 50 epochs, where learning rate reduces to its 1/10 at epoch 40. We use a linear warm-up from 0.000125 for the first 2 epochs. For all four datasets, we use random 224×224 crops and horizontal flipping from a video clip, which is randomly sampled from the full-length video and resized to a shorter edge side of randomly sampled in [256, 320] pixels.

Inference. Following previous works [56, 237], we sample 3×10 clips for each video during testing: we uniformly sample 10 clips for the temporal domain and 3 spatial crops of size 256×256 after the shorter edge size are resized to 256 pixels. We average the softmax scores across all clips for final prediction for Kinetics-400 and use the maximum of the softmax scores for Charades, Charades-Ego and MEVA. We report the computational cost of a single, spatially center-cropped clip of size 256×256.

Recognition Results

We compare the same baselines, as mentioned in the previous section. For Kinetics-400, the input frames are the same as the previous section. For Charades and Charades-Ego, we use 16×8 input frames for ResNet3D and 32×4 input frames for SlowFast, following the SlowFast paper [56].

Kinetics-400. Table 4.5 shows the experiments on Kinetics-400. Our method can improve top-1 accuracy by 1.6 and 1.5 point for ResNet3D and SlowFast, respectively, at the cost of 3.6% and 2.1% relatively more computation.

Charades. Table 4.6 shows the experiments on the Charades dataset. Our method is able to improve mAP by 2.3 and 2 points for ResNet3D and SlowFast, respectively, with only 3.6% and

Models	top-1	top-5	GFLOPs
I3D [102]	0.711	0.893	-
R(2+1)D [224]	0.720	0.900	-
DynamoNet (32 frames) [48]	0.714	0.900	-
NL-R50 (32 frames) [237]	0.749	0.916	-
ResNet3D (8x8)	0.735	0.908	109.2
ResNet3D + *STAN*	**0.751**	**0.916**	113.2
SlowFast [56] (32x2)*	0.759	0.920	131.7
SlowFast + *STAN*	**0.774**	**0.931**	134.5

Table 4.5: Experiment results on Kinetics-400 dataset. We compare recent models with ResNet-50 backbone as well as some classic methods. *We have implemented the ResNet3D and SlowFast model using Tensorflow based on the official code. We convert the released model weights and notice a drop of top-1 accuracy from 77.0% in the paper [56] to 75.9%. The reason might be that a small portion of videos in Kinetics-400 validation set are unavailable at the time of our experiments. In terms of FLOPs, we compute FLOPs using Tensorflow's profiler on each dataset (vs. PyTorch profiler in the SlowFast paper). The FLOPs difference might be due to profiler differences. We will release our code and models for reproduction.

Models	mAP	GFLOPs	MParams
ResNet3D (16x8)	0.354	218.4	32.40
ResNet3D + *STAN*	**0.377**	226.4	32.47
SlowFast [56] (32x4)	0.386	131.7	34.51
SlowFast + *STAN*	**0.406**	134.5	34.53

Table 4.6: Experiment results on the Charades dataset. We show the model performance on mAP, computation cost (in billions), and number of parameters (in millions). The computational cost is of a single 256x256 video clip of length 16x8 or 32x4 (number of frames x sampling rate) frames.

Models	1st-person	3rd-person
Baseline v1.0 [200]	0.282	0.232
ResNet3D (16x8)	0.298	0.361
ResNet3D + *STAN*	**0.318**	**0.366**
SlowFast [56] (32x4)	0.316	0.391
SlowFast + *STAN*	**0.326**	**0.396**

Table 4.7: Experiment results on Charades-Ego dataset. Models are trained on both 1st-person and 3rd-person view videos. We show the test set with 1st-person and 3rd-person view videos separately. The computation cost and number of parameters are the same as the Charades experiment.

Models	mAP
ResNet3D (16x8)	0.455
ResNet3D + *STAN*	**0.497**
SlowFast [56] (32x4)	0.484
SlowFast + *STAN*	**0.531**

Table 4.8: Experiment results on MEVA dataset.

2.1% relatively more computation. Our method only contains a few additional parameters.

Charades-Ego. Table 4.7 show the results on the Charades-Ego dataset. The test set is divided into 1st-person videos and 3rd-person videos. Note that the training set of Charades-Ego dataset includes mostly 3rd-person videos, and we can observe from the results that our *STAN* model

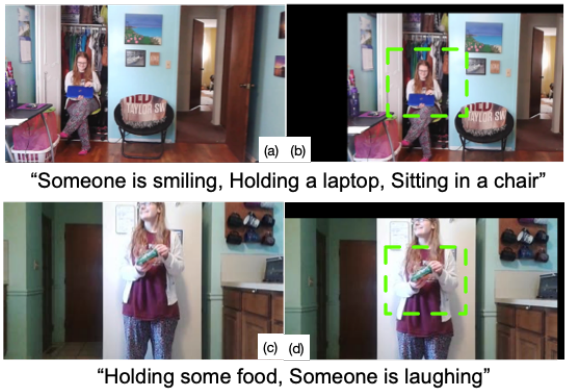

Figure 4.3: Qualitative analysis. Videos are from Charades. (a) and (c) are original frames. (b) and (d) are transformed frames. The green bounding box denotes a center reference. The sentences below the images are the ground truth labels for the video clip. See main text for details of analysis.

achieves more significant improvement on the 1st-person test set, verifying the efficacy of our model's generalization ability. With SlowFast and *STAN* model, we are able to achieve state-of-the-art performance on this dataset.

MEVA. Table 4.8 show the results on the MEVA dataset. We can observe from the results that our *STAN* model achieves significantly better results with both backbones.

Qualitative Analysis. *STAN* can provide intuitive geometric interpretation of human actions. To illustrate such effects, we visualize example transformations of our model on Charades videos on two videos with correct classification labels in Fig. 4.3. We use the output (θ, Eqn. 4.2) from the *STAN* layer (deformation network) of the first temporal group and visualize the spatial transformation using the middle frame. Fig. 4.3 (a) and (c) are the original frames. Fig. 4.3 (b) and (d) are the transformed frames using the transformation matrix predicted by *STAN*. We use a center bounding box as a reference in the visualization. As we see, for the first example, the person originally is located to the left of the scene and *STAN* learns a transformation that centers the main actor. In the second example, the person is already in the center and the transformation does not alter much of the frames.

Limitations. When examining the model transformation, we have noticed that the model does not learn much meaningful temporal transformation as we have expected. Ideally, for action videos like dunking basketball, the model should be able to slow-down video segments like jumping and speed-up parts like running. More work needs to be done during training of the transformation to allow the model to learn better temporal transformation. For example,

removing residual connection for STAN layer might help, but it may require much longer training time. In addition, different people might perform the same action differently. This kind of personal variances should also be considered in future work.

4.4 Related Work

The research of action recognition has advanced with both new datasets and new models. As one of the earliest action recognition benchmarks, KTH[194] collects videos of individual actors repetitively performing six types of human actions (walking, jogging, running, boxing, hand waving and clapping) with a clean background. Because these videos are very simple, KTH dataset turns out to be a very easy benchmark since studies quickly obtained near-perfect accuracy on it [20, 234].

To overcome the limitation of KTH, the HMDB dataset [110] was proposed in 2011 with 51 actions in 7000 video clips, while UCF101 [204] extended this effort by collecting 101 action classes in 13000 clips. Both benchmarks are captured with more diversified backgrounds. In the past decade, we have witnessed a steady improvement of accuracy on these two datasets by different methods including features fusion [163], two-stream network [202], C3D [223], I3D [21], graph-based approaches [35, 168, 236, 265] and others [26, 77, 248]. However, some clips in the UCF101 test set are taken from the same YouTube video as the training set [102], which makes it relatively easy to obtain good accuracy on UCF101. As a result, the SOTA on UCF101 dataset is more than 98%[102].

The modern benchmarks for action recognition and detection is the Kinetics dataset [102] and the AVA dataset [119], respectively. The Kinetics dataset proposes a bigger benchmark with more categories and more videos (e.g., 400 categories 160,000 clips in [102]) as a harder benchmark. The action labels in AVA [119] are annotated with spatial temporal locations, which is more challenging than the setting of one label per clip.

Many new approaches [56, 135, 224, 254, 267] have been carried on these two datasets, of which the SlowFast network [56] obtains good performance. Note that the SlowFast contains more parameters than C3D or I3D networks, by integrating features at both high and low frame rates. We can see the trend of action recognition in the last two decades is to collect bigger datasets (e.g. Kinetics) as well as build bigger models (e.g., I3D and SlowFast).

In the recent years, there has been a consistent effort to use alignment for image recognition [42, 79, 84, 107, 134, 247]. However, many previous studies show that alignment models are not as competitive as data-driven approaches like data augmentation or spatial pooling for

image recognition. Some recent works have to rely on very expensive models such as recurrent networks [134] or stacked capsules [107]. As a result, a lot of alignment-based recognition methods are limited to MNIST [107] and face recognition [247]. Some follow-up works on capsule network [52] and 2D spatial alignment network [80, 255] have been proposed but they are limited to action recognition on small datasets like UCF101 [204] and JHMDB [88]. One popular related work, deformable convolution [42], computes spatial offsets to deform traditional convolution operations based on the visual content in images. Our method defers in that: (1) we aim for spatial-temporal alignment of features; (2) we explicitly compute geometric transformation. Another related work, trajectory convolutions [267], relies on optical flow estimation to compute spatial offset at different times for 3D convolutions. Our method utilizes higher level content understanding to account for actor movements and viewpoint changes (act as camera stabilization, etc.). Our method achieves 0.853 (vs. 0.798 in [267]) on Kinetics-400 on avg. top-1&5 acc. In this chapter, we show that it is possible to build an efficient spatial-temporal alignment for both action recognition and detection, and improve recent networks with very few extra parameters.

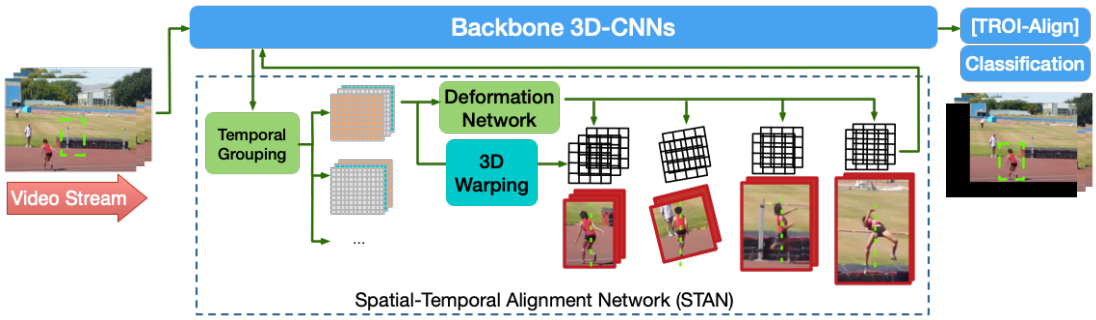

Figure 4.4: Spatial Temporal Alignment Network (*STAN*) design. In our design, different temporal slices of the feature map can undergo different transformations to account for intra-clip camera motion and actor movement. We show four alignment examples with a reference line where the actor is aligned on for demonstration. After the features are transformed and processed through the backbone network, they are passed to a temporal ROI-Align layer (TROI-Align) if the task is action detection, and then finally to the classification layer.

4.5 Summary

In this chapter we introduce a new spatial-temporal alignment network, *STAN*, for action recognition and detection. Our study is the first to explore explicit spatial-temporal alignment in 3D

CNNs for action detection. Our model can be conveniently inserted into existing networks and provides significant improvement with a low extra computation cost. We have shown that our method achieves state-of-the-art performance on multiple challenging action recognition and detection benchmarks.

Part II

Pedestrian Trajectory Prediction with Scene Semantics

In this part, we focus on the human future trajectory prediction problem. We study how trajectory prediction can benefit from scene semantic understanding of the scene. Since the future is uncertain, we first introduce the *Multiverse* model to tackle the multiple-future trajectory prediction problem (chapter 5). To alleviate the limited training data challenge as mentioned in previous section, we propose a machine learning algorithm called *SimAug*, to efficiently learn from 3D simulation data for trajectory prediction (chapter 6).

Chapter 5

Multiple-future Pedestrian Trajectory Prediction

In this chapter, we study the uncertainty of future trajectory predictions, by proposing the *Multiverse* model [132], which generates multiple-future trajectories with probability distributions. We also develop a novel multiple-future trajectory benchmark, called the *ForkingPaths* dataset, using 3D simulation.

5.1 Motivation

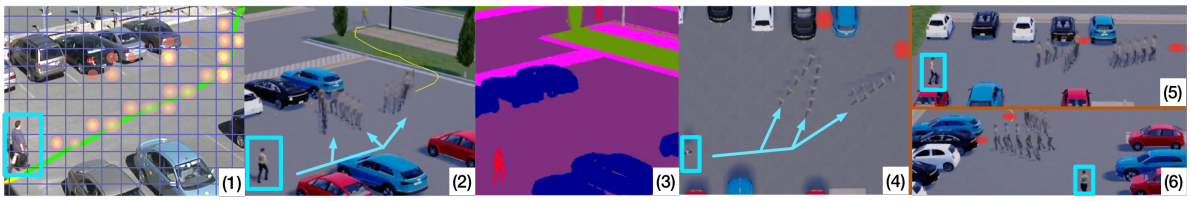

Figure 5.1: Illustration of person trajectory prediction. (1) A person walks towards a car (data from the VIRAT/ActEV dataset). The green line is the actual future trajectory and the yellow-orange heatmaps are example future predictions. Although these predictions near the cars are plausible, they would be considered errors in the real video dataset. (2) To combat this, we propose a new dataset called "Forking Paths"; here we illustrate 3 possible futures created by human annotators controlling agents in a synthetic world derived from real data. (3) Here we show semantic segmentation of the scene. (4-6) Here we show the same scene rendered from different viewing angles, where the red circles are future destinations.

Forecasting future human behavior is a fundamental problem in video understanding. In particular, future path prediction, which aims at forecasting a pedestrian's future trajectory in the next few seconds, has received a lot of attention in our community [3, 67, 103, 121]. This functionality is a key component in a variety of applications such as autonomous driving [11, 24], long-term object tracking [96, 190], safety monitoring [131], robotic planning [179, 180], etc.

Of course, the future is often very uncertain: Given the same historical trajectory, a person may take different paths, depending on their (latent) goals. Thus recent work has started focusing on *multi-future trajectory prediction* [24, 117, 121, 146, 217, 219].

Consider the example in Fig. 5.1. We see a person moving from the bottom left towards the top right of the image, and our task is to predict where he will go next. Since there are many possible future trajectories this person might follow, we are interested in learning a model that can generate multiple plausible futures. However, since the ground truth data only contains one trajectory, it is difficult to evaluate such probabilistic models.

To overcome the aforementioned challenges, our first contribution is the creation of a realistic synthetic dataset that allows us to compare models in a quantitative way in terms of their ability to predict multiple plausible futures, rather than just evaluating them against a single observed trajectory as in existing studies. We create this dataset using the 3D CARLA [50] simulator, where the scenes are manually designed to be similar to those found in the challenging real-world benchmark VIRAT/ActEV [7, 158]. Once we have recreated the static scene, we automatically reconstruct trajectories by projecting real-world data to the 3D simulation world. See Fig. 5.1 and 5.3. We then semi-automatically select a set of plausible future destinations (corresponding to semantically meaningful locations in the scene), and ask human annotators to create multiple possible continuations of the real trajectories towards each such goal. In this way, our dataset is "anchored" in reality, and yet contains plausible variations in high-level human behavior, which is impossible to simulate automatically.

We call this dataset the "Forking Paths" dataset, a reference to the short story by Jorge Luis Borges.[1] As shown in Fig. 5.1, different human annotations have created forkings of future trajectories for the identical historical past. So far, we have collected 750 sequences, with each covering about 15 seconds, from 10 annotators, controlling 127 agents in 7 different scenes. Each agent contains 5.9 future trajectories on average. We render each sequence from 4 different views, and automatically generate dense labels, as illustrated in Fig. 5.1 and 5.3. In total,

[1] https://en.wikipedia.org/wiki/The_Garden_of_Forking_Paths

this amounts to 3.2 hours of trajectory sequences, which is comparable to the largest person trajectory benchmark VIRAT/ActEV [7, 158] (4.5 hours), or 5 times bigger than the common ETH/UCY [118, 140] benchmark. We therefore believe this will serve as a benchmark for evaluating models that can predict multiple futures.

Our second contribution is to propose a new probabilistic model, *Multiverse*, which can generate multiple plausible trajectories given the past history of locations and the scene. The model contains two novel design decisions. First, we use a multi-scale representation of locations. In the first scale, the coarse scale, we represent locations on a 2D grid, as shown in Fig. 5.1(1). This captures high level uncertainty about possible destinations and leads to a better representation of multi-modal distributions. In the second fine scale, we predict a real-valued offset for each grid cell, to get more precise localization. This two-stage approach is partially inspired by object detection methods [178]. The second novelty of our model is to design convolutional RNNs [246] over the spatial graph as a way of encoding inductive bias about the movement patterns of people.

In addition, we empirically validate our model on the challenging real-world benchmark VIRAT/ActEV [7, 158] for single-future trajectory prediction, in which our model achieves the best-published result. On the proposed simulation data for multi-future prediction, experimental results show our model compares favorably against the state-of-the-art models across different settings. To summarize, the main contributions of this chapter are as follows: *(i)* We introduce the first dataset and evaluation methodology that allows us to compare models in a quantitative way in terms of their ability to predict multiple plausible futures. *(ii)* We propose a new effective model for multi-future trajectory prediction. *(iii)* We establish a new state of the art result on the challenging VIRAT/ActEV benchmark, and compare various methods on our multi-future prediction dataset.

5.2 The *Multiverse* Model

In this section, we describe our model for forecasting agent trajectories, which we call *Multiverse*. We focus on predicting the locations of a single agent for multiple steps into the future, $L_{h+1:T}$, given a sequence of past video frames, $V_{1:h}$, and agent locations, $L_{1:h}$, where h is the history length and $T-h$ is the prediction length. Since there is inherent uncertainty in this task, our goal is to design a model that can effectively predict multiple plausible future trajectories, by computing the multimodal distribution $p(L_{h+1:T}|L_{1:h}, V_{1:h})$. See Fig. 5.2 for a high level summary of the model, and the sections below for more details.

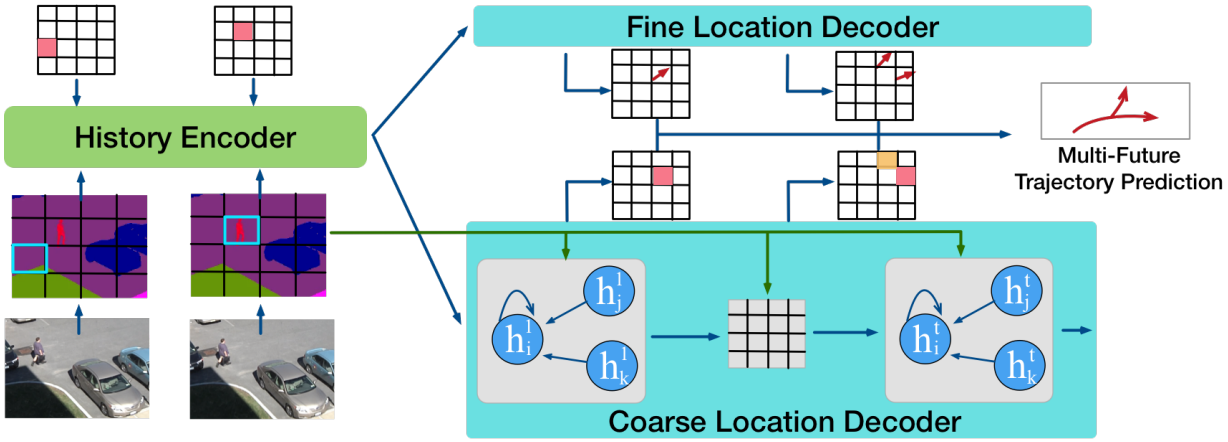

Figure 5.2: Overview of our model. The input to the model is the ground truth location history, and a set of video frames, which are preprocessed by a semantic segmentation model. This is encoded by the "History Encoder" convolutional RNN. The output of the encoder is fed to the convolutional RNN decoder for location prediction. The coarse location decoder outputs a heatmap over the 2D grid of size $H \times W$. The fine location decoder outputs a vector offset within each grid cell. These are combined to generate a multimodal distribution over $\mathbb{R}^2$ for predicted locations.

5.2.1 History Encoder

The encoder computes a representation of the scene from the history of past locations, $L_{1:h}$, and frames, $V_{1:h}$. We encode each ground truth location L_t by an index $Y_t \in G$ representing the nearest cell in a 2D grid G of size $H \times W$, indexed from 1 to HW. Inspired by [116, 137], we encode location with two different grid scales (36×18 and 18×9); we show the benefits of this multi-scale encoding in Section 5.4.3. For simplicity of presentation, we focus on a single $H \times W$ grid.

To make the model more invariant to low-level visual details, and thus more robust to domain shift (e.g., between different scenes, different views of the same scene, or between real and synthetic images), we preprocess each video frame V_t using a pre-trained semantic segmentation model, with $K = 13$ possible class labels per pixel. We use the Deeplab model [31] trained on the ADE20k [269] dataset, and keep its weights frozen. Let S_t be this semantic segmentation map modeled as a tensor of size $H \times W \times K$.

We then pass these inputs to a convolutional RNN [238, 246] to compute a spatial-temporal feature history:

$$H^e_t = \text{ConvRNN}(\text{one-hot}(Y_t) \odot (W * S_t), H^e_{t-1}) \tag{5.1}$$

where $\odot$ is element wise product, and $*$ represents 2D-convolution. The function one-hot$(\cdot)$ projects a cell index into an one-hot embedding of size $H \times W$ according to its spatial location.

We use the final state of this encoder $H_t^e \in \mathbb{R}^{H \times W \times d_{enc}}$, where d_{enc} is the hidden size, to initialize the state of the decoders. We also use the temporal average of the semantic maps, $\overline{S} = \frac{1}{h}\sum_{t=1}^{h} S_t$, during each decoding step. The context is represented as $\mathcal{H} = [H_h^e, \overline{S}]$.

5.2.2 Coarse Location Decoder

After getting the context $\mathcal{H}$, our goal is to forecast future locations. We initially focus on predicting locations at the level of grid cells, $Y_t \in G$. In Section 5.2.3, we discuss how to predict a continuous offset in $\mathbb{R}^2$, which specifies a "delta" from the center of each grid cell, to get a fine-grained location prediction.

Let the coarse distribution over grid locations at time t (known as the "belief state") be denoted by $C_t(i) = p(Y_t = i | Y_{h:t-1}, \mathcal{H})$, for $\forall i \in G$ and $t \in [h+1, T]$. For brevity, we use a single index i to represent a cell in the 2D grid. Rather than assuming a Markov model, we update this using a convolutional recurrent neural network, with hidden states H_t^C. We then compute the belief state by:

$$C_t = \text{softmax}(W * H_t^C) \in \mathbb{R}^{HW} \tag{5.2}$$

Here we use 2D-convolution with one filter and flatten the spatial dimension before applying softmax. The hidden state is updated using:

$$H_t^C = \text{ConvRNN}(\text{GAT}(H_{t-1}^C), \text{embed}(C_{t-1})) \tag{5.3}$$

where embed(C_{t-1}) embeds into a 3D tensor of size $H \times W \times d_e$ and d_e is the embedding size. GAT(H_{t-1}^C) is a graph attention network [231], where the graph structure corresponds to the 2D grid in G. More precisely, let h_i be the feature vector corresponding to the i-th grid cell in H_{t-1}^C, and let $\tilde{h}_i$ be the corresponding output in $\tilde{H}_{t-1}^C = \text{GAT}(H_{t-1}^C) \in \mathbb{R}^{H \times W \times d_{dec}}$, where d_{dec} is the size of the decoder hidden state. We compute these outputs of GAT using:

$$\tilde{h}_i = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} f_e([v_i, v_j]) + h_i \tag{5.4}$$

where $\mathcal{N}_i$ are the neighbors of node v_i in G with each node represented as $v_i = [h_i, \overline{S}_i]$, where $\overline{S}_i$ collects the cell i's feature in $\overline{S}$. f_e is some edge function (implemented as an MLP in our experiments) that computes the attention weights.

The graph-structured update function for the RNN ensures that the probability mass "diffuses out" to nearby grid cells in a controlled manner, reflecting the prior knowledge that people do not suddenly jump between distant locations. This inductive bias is also encoded in the convolutional structure, but adding the graph attention network gives improved results, because the weights are input-dependent and not fixed.

5.2.3 Fine Location Decoder

The 2D heatmap is useful for capturing multimodal distributions, but does not give very precise location predictions. To overcome this, we train a second convolutional RNN decoder H_t^O to compute an offset vector for each possible grid cell using a regression output, $O_t = \text{MLP}(H_t^O) \in \mathbb{R}^{H \times W \times 2}$. This RNN is updated using

$$H_t^O = \text{ConvRNN}(\text{GAT}(H_{t-1}^O), O_{t-1}) \in \mathbb{R}^{H \times W \times d_{dec}} \tag{5.5}$$

To compute the final prediction location, we first flatten the spatial dimension of O_t into $\tilde{O}_t \in \mathbb{R}^{HW \times 2}$. Then we use

$$L_t = Q_i + \tilde{O}_{ti} \tag{5.6}$$

where i is the index of the selected grid cell, $Q_i \in \mathbb{R}^2$ is the center of that cell, and $\tilde{O}_{ti} \in \mathbb{R}^2$ is the predicted offset for that cell at time t. For single-future prediction, we use greedy search, namely $i = \operatorname{argmax} C_t$ over the belief state. For multi-future prediction, we use beam search in Section 5.2.5.

This idea of combining classification and regression is partially inspired by object detection methods (e.g., [178]). It is worth noting that in concurrent work, [24] also designed a two-stage model for trajectory forecasting. However, their classification targets are pre-defined anchor trajectories. Ours is not limited by the predefined anchors.

5.2.4 Training

Our model trains on the observed trajectory from time 1 to h and predicts the future trajectories (in xy-coordinates) from time $h+1$ to T. We supervise this training by providing ground truth targets for both the heatmap (belief state), C_t^*, and regression offset map, O_t^*. In particular, for the coarse decoder, the cross-entropy loss is used:

$$\mathcal{L}_{cls} = -\frac{1}{T} \sum_{t=h+1}^{T} \sum_{i \in G} C_{ti}^* \log(C_{ti}) \tag{5.7}$$

For the fine decoder, we use the smoothed L_1 loss used in object detection [178]:

$$\mathcal{L}_{reg} = \frac{1}{T} \sum_{t=h+1}^{T} \sum_{i \in G} \text{smooth}_{L_1}(O_{ti}^*, O_{ti}) \tag{5.8}$$

where $O_{ti}^* = L_t^* - Q_i$ is the delta between the true location and the center of the grid cell at i and L_t^* is the ground truth for L_t in Eq.(5.6). We impose this loss on every cell to improve the robustness.

The final loss is then calculated using

$$\mathcal{L}(\theta) = \mathcal{L}_{cls} + \lambda_1 \mathcal{L}_{reg} + \lambda_2 \|\theta\|_2^2 \tag{5.9}$$

where λ_2 controls the ℓ_2 regularization (weight decay), and $\lambda_1 = 0.1$ is used to balance the regression and classification losses.

Note that during training, when updating the RNN, we feed in the predicted soft distribution over locations, C_t. See Eq. (5.2). An alternative would be to feed in the true values, C_t^*, i.e., use teacher forcing. However, this is known to suffer from problems [173].

5.2.5 Inference

To generate multiple qualitatively distinct trajectories, we use the diverse beam search strategy from [120]. To define this precisely, let B_{t-1} be the beam at time $t-1$; this set contains K trajectories (history selections) $M_{t-1}^k = \{\hat{Y}_1^k, \ldots, \hat{Y}_{t-1}^k\}$, $k \in [1, K]$, where $\hat{Y}_t^k$ is an index in G, along with their accumulated log probabilities, P_{t-1}^k. Let $C_t^k = f(M_{t-1}^k) \in \mathbb{R}^{HW}$ be the coarse location output probability from Eq. (5.2) and (5.3) at time t given inputs M_{t-1}^k.

The new beam is computed using

$$B_t = \text{topK}\left(\{P_{t-1}^k + \log(C_t^k(i)) + \gamma(i) | \forall i \in G, k \in [1, K]\}\right) \tag{5.10}$$

where $\gamma(i)$ is a diversity penalty term, and we take the top K elements from the set produced by considering values with $k = 1 : K$. If $K = 1$, this reduces to greedy search.

Once we have computed the top K future predictions, we add the corresponding offset vectors to get K trajectories by $L_t^k \in \mathbb{R}^2$. This constitutes the final output of our model.

5.3 The Forking Paths Dataset

In this section, we describe our human-annotated simulation dataset, called Forking Paths, for multi-future trajectory evaluation.

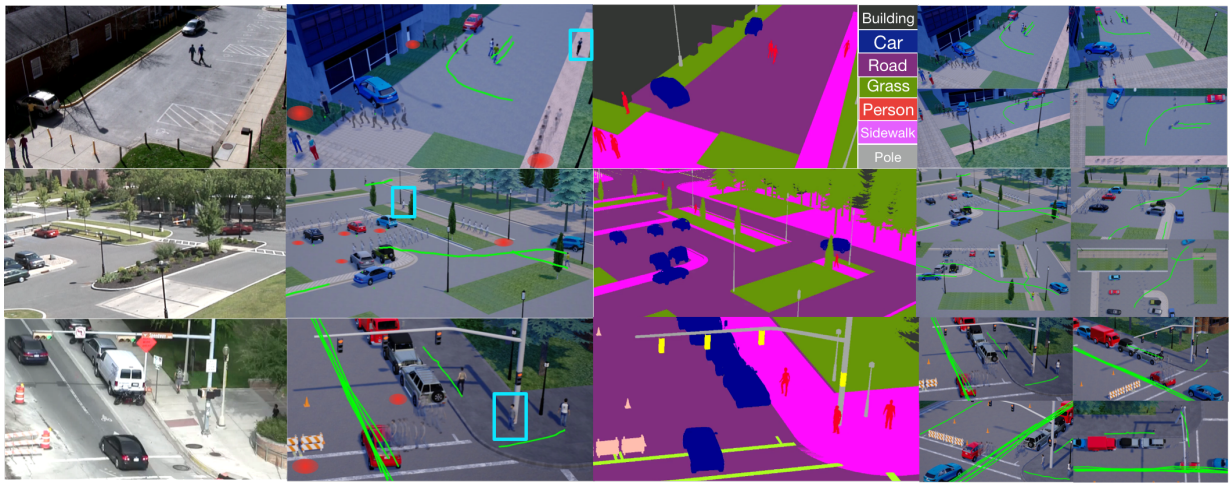

Figure 5.3: Visualization of the Forking Paths dataset. On the left is examples of the real videos and the second column shows the reconstructed scenes. The person in the blue bounding box is the controlled agent and multiple future trajectories annotated by humans are shown by overlaid person frames. The red circles are the defined destinations. The green trajectories are future trajectories of the reconstructed uncontrolled agents. The scene semantic segmentation ground truth is shown in the third column and the last column shows all four camera views including the top-down view.

Existing datasets. There are several real-world datasets for trajectory evaluation, such as SDD [184], ETH/UCY [118, 162], KITTI [59], nuScenes [18] and VIRAT/ActEV [7, 158]. However, they all share the fundamental problem that one can only observe one out of many possible future trajectories sampled from the underlying distribution. This is broadly acknowledged in prior works [24, 67, 146, 180, 181, 219] but has not yet been addressed.

The closest work to ours is the simulation used in [24, 146, 219]. However, these only contain artificial trajectories, not human generated ones. Also, they use a highly simplified 2D space, with pedestrians oversimplified as points and vehicles as blocks; no other scene semantics are provided.

Reconstructing reality in simulator. In this work, we use CARLA [50], a near-realistic open source simulator built on top of the Unreal Engine 4. Following prior simulation datasets [57, 185], we *semi-automatically* reconstruct static scenes and their dynamic elements from the real-world videos in ETH/UCY and VIRAT/ActEV. There are 4 scenes in ETH/UCY and 5 in VIRAT/ActEV. We exclude 2 cluttered scenes (UNIV & 0002) that we are not able to reconstruct in CARLA, leaving 7 static scenes in our dataset.

For dynamic movement of vehicle and pedestrian, we first convert the ground truth trajectory annotations from the real-world videos to the ground plane using the provided homography matrices. We then match the real-world trajectories' origin to correct locations in the re-created scenes.

Human generation of plausible futures. We manually select sequences with more than one pedestrian. We also require that at least one pedestrian could have multiple plausible alternative destinations. We insert plausible pedestrians into the scene to increase the diversity of the scenarios. We then select one of the pedestrians to be the "controlled agent" (CA) for each sequence, and set meaningful destinations within reach, like a car or an entrance of a building. On average, each agent has about 3 destinations to move towards. In total, we have 127 CAs from 7 scenes. We call each CA and their corresponding scene a scenario.

For each scenario, there are on average 5.9 human annotators to control the agent to the defined destinations. Specifically, they are asked to watch the first 5 seconds of video, from a first-person view (with the camera slightly behind the pedestrian) and/or an overhead view (to give more context). They are then asked to control the motion of the agent so that it moves towards the specified destination in a "natural" way, e.g., without colliding with other moving objects (whose motion is derived from the real videos, and is therefore unaware of the controlled agent). The annotation is considered successful if the agent reached the destination without colliding within the time limit of 10.4 seconds. All final trajectories in our dataset are examined by humans to ensure reliability.

Note that our videos are up to 15.2 seconds long. This is slightly longer than previous works (e.g. [3, 67, 121, 131, 191, 262, 266]) that use 3.2 seconds of observation and 4.8 seconds for prediction. (We use 10.4 seconds for the future to allow us to evaluate longer term forecasts.)

Generating the data. Once we have collected human-generated trajectories, 750 in total after data cleaning, we render each one in four camera views (three 45-degree and one top-down view). Each camera view has 127 scenarios in total and each scenario has on average 5.9 future trajectories. With CARLA, we can also simulate different weather conditions, although we did not do so in this work. In addition to agent location, we collect ground truth for pixel-precise scene semantic segmentation from 13 classes including sidewalk, road, vehicle, pedestrian, etc. See Fig. 5.3.

5.4 Experimental Results

This section evaluates various methods, including our *Multiverse* model, for multi-future trajectory prediction on the proposed Forking Paths dataset. To allow comparison with previous works, we also evaluate our model on the challenging VIRAT/ActEV [7, 158] benchmark for single-future path prediction.

Multi-Future Evaluation. Let $Y^{ij} = Y^{ij}_{t=(h+1)\cdots T}$ be the j-th true future trajectory for the i-th test sample, for $\forall j \in [1, J]$, and let $\hat{Y}^{ik}$ be the k'th sample from the predicted distribution over trajectories, for $k \in [1, K]$. Since there is no agreed-upon evaluation metric for this setting, we simply extend the above metrics, as follows: i) *Minimum Average Displacement Error Given K Predictions with Multi-modal Ground-Truth* (minADE^M_K): similar to the metric described in [24, 67, 180, 181], for each true trajectory j in test sample i, we select the closest overall prediction (from the K model predictions), and then measure its average error:

$$\text{minADE}^M_K = \frac{\sum_{i=1}^{N}\sum_{j=1}^{J}\min_{k=1}^{K}\sum_{t=h+1}^{T}\|Y^{ij}_t - \hat{Y}^{ik}_t\|_2}{N \times (T-h) \times J} \tag{5.11}$$

ii) *Minimum Final Displacement Error Given K Predictions with Multi-modal Ground-Truth* (minFDE^M_K): similar to minADE^M_K, but we only consider the predicted points and the ground truth point at the final prediction time instant:

$$\text{minFDE}^M_K = \frac{\sum_{i=1}^{N}\sum_{j=1}^{J}\min_{k=1}^{K}\|Y^{ij}_T - \hat{Y}^{ik}_T\|_2}{N \times J} \tag{5.12}$$

iii) *Negative Log-Likelihood* (NLL): Similar to NLL metrics used in [24, 146], we measure the fit of ground-truth samples to the predicted distribution.

5.4.1 Multi-Future Prediction on Forking Paths

Dataset & Setups. The proposed Forking Paths dataset is used for multi-future trajectory prediction evaluation. Following the setting in previous works [3, 67, 131, 146, 191], we downsample the videos to 2.5 fps and extract person trajectories using code released in [131], and let the models observe 3.2 seconds (8 frames) of the controlled agent before outputting trajectory coordinates in the pixel space. Since the length of the ground truth future trajectories are different, each model needs to predict the maximum length at test time but we evaluate the predictions using the actual length of each true trajectory.

Baseline methods. We compare our method with two simple baselines, and three recent methods with released source code, including a recent model for multi-future prediction and the

Method	Input Types	minADE_{20}^{M}		minFDE_{20}^{M}	
		45-degree	top-down	45-degree	top-down
Linear	Traj.	213.2	197.6	403.2	372.9
LSTM	Traj.	201.0 $\pm$2.2	183.7 $\pm$2.1	381.5 $\pm$3.2	355.0 $\pm$3.6
Social-LSTM [3]	Traj.	197.5 $\pm$2.5	180.4 $\pm$1.0	377.0 $\pm$3.6	350.3 $\pm$2.3
Social-GAN (PV) [67]	Traj.	191.2 $\pm$5.4	176.5 $\pm$5.2	351.9 $\pm$11.4	335.0 $\pm$9.4
Social-GAN (V) [67]	Traj.	187.1 $\pm$4.7	172.7 $\pm$3.9	342.1 $\pm$10.2	326.7 $\pm$7.7
Next [131]	Traj.+Bbox+RGB+Seg.	186.6 $\pm$2.7	166.9 $\pm$2.2	360.0 $\pm$7.2	326.6 $\pm$5.0
Ours	Traj.+Seg.	**168.9** $\pm$2.1	**157.7** $\pm$2.5	**333.8** $\pm$3.7	**316.5** $\pm$3.4

Table 5.1: Comparison of different methods on the Forking Paths dataset. Lower numbers are better. The numbers for the column labeled "45 degrees" are averaged over 3 different 45-degree views. For the input types, "Traj.", "RGB", "Seg." and "Bbox." mean the inputs are xy coordinates, raw frames, semantic segmentations and bounding boxes of all objects in the scene, respectively. All models are trained on real VIRAT/ActEV videos and tested on synthetic (CARLA-rendered) videos.

state-of-the-art model for single-future prediction: ***Linear*** is a single layer model that predicts the next coordinates using a linear regressor based on the previous input point. ***LSTM*** is a simple LSTM [74] encoder-decoder model with coordinates input only. ***Social LSTM*** [3]: We use the open source implementation from (https://github.com/agrimgupta92/sgan/). ***Next*** [131] is the state-of-the-art method for single-future trajectory prediction on the VIRAT/ActEV dataset. We train the Next model without the activity labels for fair comparison using the code from (https://github.com/google/next-prediction/). ***Social GAN*** [67] is a recent multi-future trajectory prediction model trained using Minimum over N (MoN) loss. We train two model variants (called PV and V) detailed in the paper using the code from [67] .

All models are trained on real videos (from VIRAT/ActEV – see Section 5.4.2 for details) and tested on our synthetic videos (with CARLA-generated pixels, and annotator-generated trajectories). Most models just use trajectory data as input, except for our model (which uses trajectory and semantic segmentation) and Next (which uses trajectory, bounding box, semantic segmentation, and RGB frames).

Implementation Details. We use ConvLSTM [246] cell for both the encoder and decoder. The embedding size is set to 32, and the hidden sizes for the encoder and decoder are both 256. The scene semantic segmentation features are extracted from the deeplab model [31], pretrained on

Method	$T_{pred} = 1$	$T_{pred} = 2$	$T_{pred} = 3$
(PV) [14]	10.08 $\pm$0.25	17.28 $\pm$0.42	23.34 $\pm$0.47
(V) [14]	9.95 $\pm$0.35	17.38 $\pm$0.49	23.24 $\pm$0.54
Next [27]	8.32 $\pm$0.10	14.98 $\pm$0.19	22.71 $\pm$0.11
Ours	**2.22** $\pm$0.54	**4.46** $\pm$1.33	**8.14** $\pm$2.81

Table 5.2: Negative Log-likelihood comparison of different methods on the Forking Paths dataset. For methods that output multiple trajectories, we quantize the xy-coordinates into the same grid as our method and get a normalized probability distribution prediction.

the ADE-20k [269] dataset. We use Adadelta optimizer [259] with an initial learning rate of 0.3 and weight decay of 0.001. Other hyper-parameters for the baselines are the same to the ones in [67, 131]. We evaluate the top $K = 20$ predictions for multi-future trajectories. For the models that only output a single trajectory, including Linear, LSTM, Social-LSTM, and Next, we duplicate the output for K times before evaluating. For Social-GAN, we use K different random noise inputs to get the predictions. For our model, we use diversity beam search [120, 164] as described in Section 5.2.5.

Quantitative Results. Table 5.1 lists the multi-future evaluation results, where we divide the evaluation according to the viewing angle of camera, 45-degree vs. top-down view. We repeat all experiments (except "linear") 5 times with random initialization to produce the mean and standard deviation values. As we see, our model outperforms baselines in all metrics and it performs significantly better on the minADE_K^M metric, which suggests better prediction quality over all time instants. Notably, our model outperforms Social GAN by a large margin of at least 8 points on all metrics. We also measure the standard negative log-likelihood (NLL) metric for the top methods in Table 5.2.

Qualitative analysis. We visualize some outputs of the top 4 methods in Fig. 5.4. In each image, the yellow trajectories are the history trajectory of each controlled agent (derived from real video data) and the green trajectories are the ground truth future trajectories from human annotators. The predicted trajectories are shown in yellow-orange heatmaps for multi-future prediction methods, and in red lines for single-future prediction methods. As we see, our model correctly generally puts probability mass where there is data, and does not "waste" probability mass where there is no data.

Error analysis. We show some typical errors our model makes in Fig. 5.5. The first image shows our model misses the correct direction, perhaps due to lack of diversity in our sampling

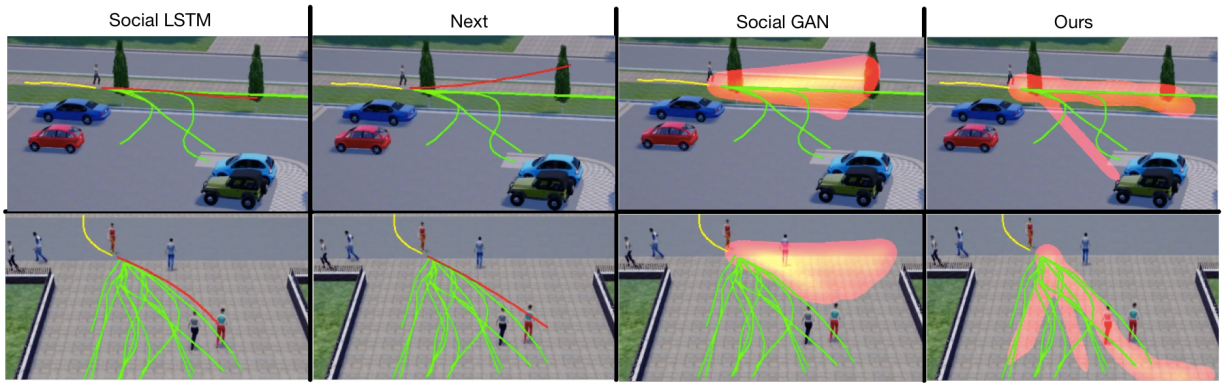

Figure 5.4: Qualitative analysis. The red trajectories are single-future method predictions and the yellow-orange heatmaps are multi-future method predictions. The yellow trajectories are observations and the green ones are ground truth multi-future trajectories. See text for details.

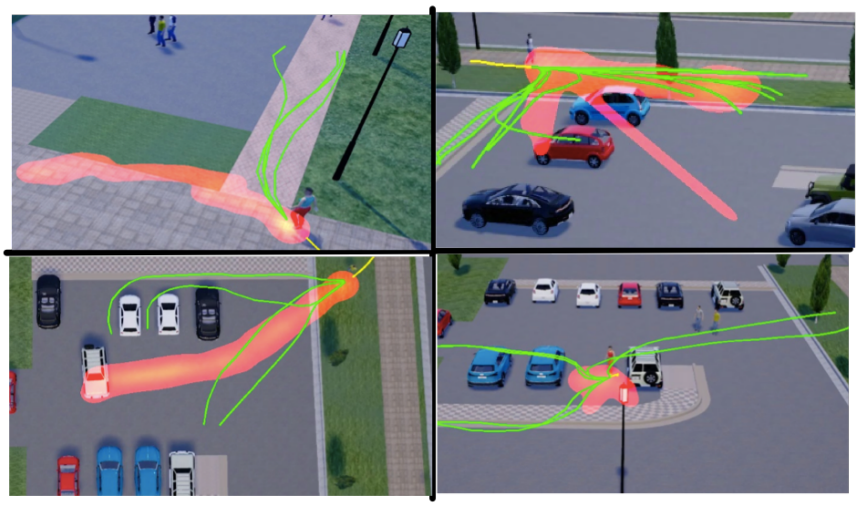

Figure 5.5: Error analysis. See text for details.

procedure. The second image shows our model sometimes predicts the person will "go through" the car (diagonal red beam) instead of going around it. This may be addressed by adding more training examples of "going around" obstacles. The third image shows our model predicts the person will go to a moving car. This is due to the lack of modeling of the dynamics of other far-away agents in the scene. The fourth image shows a hard case where the person just exits the vehicle and there is no indication of where they will go next (so our model "backs off" to a sensible "stay nearby" prediction). We leave solutions to these problems to future work.

5.4.2 Single-Future Prediction on VIRAT/ActEV

Dataset & Setups. NIST released VIRAT/ActEV [7] for activity detection research in streaming videos in 2018. This dataset is a new version of the VIRAT [158] dataset, with more videos and annotations. The length of videos with publicly available annotations is about 4.5 hours. Following [131], we use the official training set for training and the official validation set for testing. Other setups are the same as in Section 5.4.1, except we use the single-future evaluation metric.

Quantitative Results. Table 5.3 (first column) shows the evaluation results. As we see, our model achieves state-of-the-art performance. The improvement is especially large on Final Displacement Error (FDE) metric, attributing to the coarse location decoder that helps regulate the model prediction for long-term prediction. The gain shows that our model does well at both single future prediction (on real data) and multiple future prediction on our quasi-synthetic data.

Generalizing from simulation to real-world. As described in the previous section, we generate simulation data first by reconstructing from real-world videos. To verify the quality of the reconstructed data, and the efficacy of learning from simulation videos, we train all the models on the simulation videos derived from the real data. We then evaluate on the real test set of VIRAT/ActEV. As we see from the right column in Table 5.3, all models do worse in this scenario, due to the difference between synthetic and real data. We find the performance ranking of different methods are consistent between the real and our simulation training data. This suggests the errors mainly coming from the model, and substantiates the rationality of using the proposed dataset to compare the relative performance of different methods.

There are two sources of error. The synthetic trajectory data only contains about 60% of the real trajectory data, due to difficulties reconstructing all the real data in the simulator. In addition, the synthetic images are not photo realistic. Thus methods (such as Next [131]) that rely on RGB input obviously suffer the most, since they have never been trained on "real pixels". Our method, which uses trajectories plus high level semantic segmentations (which transfers from synthetic to real more easily) suffers the least drop in performance, showing its robustness to "domain shift". See Table 5.1 for input source comparison between methods.

5.4.3 Ablation Experiments

We test various ablations of our model on both the single-future and multi-future trajectory prediction to substantiate our design decisions. Results are shown in Table 5.4, where the ADE/FDE

Method	Trained on Real.	Trained on Sim.
Linear	32.19 / 60.92	48.65 / 90.84
LSTM	23.98 / 44.97	28.45 / 53.01
Social-LSTM [3]	23.10 / 44.27	26.72 / 51.26
Social-GAN (V) [67]	30.40 / 61.93	36.74 / 73.22
Social-GAN (PV) [67]	30.42 / 60.70	36.48 / 72.72
Next [131]	19.78 / 42.43	27.38 / 62.11
Ours	**18.51** / **35.84**	**22.94** / **43.35**

Table 5.3: Comparison of different methods on the VIRAT/ActEV dataset. We report ADE/FDE metrics. First column is for models trained on real video training set and second column is for models trained on the simulated version of this dataset.

Method	Single-Future	Multi-Future
Our full model	18.51 / 35.84	166.1 / 329.5
No spatial graph	28.68 / 49.87	184.5 / 363.2
No fine location decoder	53.62 / 83.57	232.1 / 468.6
No multi-scale grid	21.09 / 38.45	171.0 / 344.4

Table 5.4: Performance on ablated versions of our model on single and multi-future trajectory prediction. Lower numbers are better.

metrics are shown in the "single-future" column and minADE_{20}^{M}/minFDE_{20}^{M} metrics (averaged across all views) in the "multi-future" column. We verify three of our key designs by leaving the module out from the full model.

(1) *Spatial Graph:* Our model is built on top of a spatial 2D graph that uses graph attention to model the scene features. We train model without the spatial graph. As we see, the performance drops on both tasks. (2) *Fine location decoder:* We test our model without the fine location decoder and only use the grid center as the coordinate output. As we see, the significant performance drops on both tasks verify the efficacy of this new module proposed in our study. (3) *Multi-scale grid:* We utilize two different grid scales (36×18) and (18×9) in training. We see that performance is slightly worse if we only use the fine scale (36×18) .

5.5 Related Work

This work falls under the category of sequential models that utilize both static and dynamic environmental cues in the human motion prediction literature [186]. In the following, we also review a few relevant recent approaches based on their outputs. Then we also review the trajectory prediction datasets.

Single-future trajectory prediction. Recent works have tried to predict a single best trajectory for pedestrians or vehicles. Early works [149, 251, 262] focused on modeling person motions by considering them as points in the scene. These research works [105, 131, 143, 253] have attempted to predict person paths by utilizing visual features. Recently Liang et al. [131] proposed a joint future activity and trajectory prediction framework that utilized multiple visual features using focal attention [128, 130]. Many works [11, 76, 117, 192, 266] in vehicle trajectory prediction have been proposed. CAR-Net [192] proposed attention networks on top of scene semantic CNN to predict vehicle trajectories. Chauffeurnet [11] utilized imitation learning for trajectory prediction.

Multi-future trajectory prediction. Many works have tried to model the uncertainty of trajectory prediction. Various papers (e.g. [103, 180, 181] use Inverse Reinforcement Learning (IRL) to forecast human trajectories. Social-LSTM [3] is a popular method using social pooling to predict future trajectories. Other works [5, 67, 121, 191] like Social-GAN [67] have utilized generative adversarial networks [63] to generate diverse person trajectories. In vehicle trajectory prediction, DESIRE [117] utilized variational auto-encoders (VAE) to predict future vehicle trajectories. Many recent works [24, 146, 217, 219] also proposed probabilistic frameworks for multi-future vehicle trajectory prediction. Different from these previous works, we present a flexible two-stage framework that combines multi-modal distribution modeling and precise location prediction.

Trajectory Prediction Datasets. Many vehicle trajectory datasets [18, 25] have been proposed as a result of self-driving's surging popularity. With the recent advancement in 3D computer vision research [50, 72, 124, 182, 185, 195, 264], many research works [44, 45, 57, 171, 211, 242, 271] have looked into 3D simulated environment for its flexibility and ability to generate enormous amount of data. We are the first to propose a 3D simulation dataset that is reconstructed from real-world scenarios complemented with a variety of human trajectory continuations for multi-future person trajectory prediction.

5.6 Summary

In this chapter, we have introduced the Forking Paths dataset, and the *STAN* model for multi-future forecasting. Our study is the first to provide a quantitative benchmark and evaluation methodology for multi-future trajectory prediction by using human annotators to create a variety of trajectory continuations under the identical past. Our model utilizes multi-scale location decoders with graph attention model to predict multiple future locations. We have shown that our method achieves state-of-the-art performance on two challenging benchmarks: the large-scale real video dataset and our proposed multi-future trajectory dataset. We believe our dataset, together with our models, will facilitate future research and applications on multi-future prediction.

Chapter 6

Learning from 3D Simulation

In this chapter, we explore the benefit of multi-camera-view video data from 3D simulation created in chapter 5 to train a robust trajectory prediction model that could perform fairly well on out-of-domain testing datasets.

6.1 Motivation

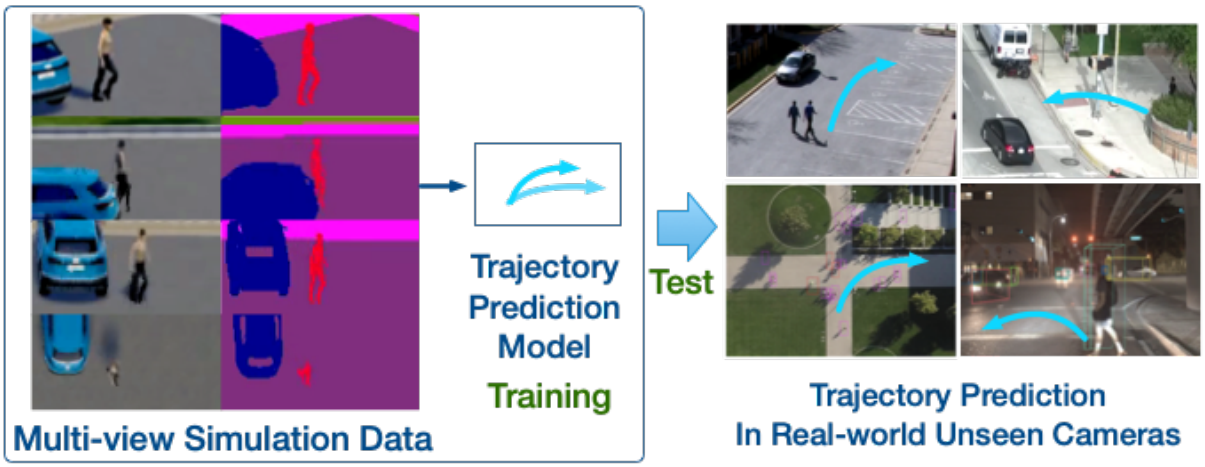

Figure 6.1: Illustration of pedestrian trajectory prediction in unseen cameras. We propose to learn robust representations only from 3D simulation data that could generalize to real-world videos captured by unseen cameras.

Future trajectory prediction [3, 67, 103, 117, 131, 132, 191] is a fundamental problem in video analytics, which aims at forecasting a pedestrian's future path in the video in the next

few seconds. Recent advancements in future trajectory prediction have been successful in a variety of vision applications such as self-driving vehicles [11, 24, 25], safety monitoring [131], robotic planning [179, 180], among others.

A notable bottleneck for existing works is that the current model is closely coupled with the video cameras on which it is trained, and generalizes poorly on new cameras with novel views or scenes. For example, prior works have proposed various models to forecast a pedestrian's trajectories in video cameras of different types such as stationary outdoor cameras [3, 67, 118, 130, 140, 158], drone cameras [47, 121, 191], ground-level egocentric cameras [179, 208, 253], or dash cameras [25, 148, 207]. However, existing models are all separately trained and tested within one or two datasets, and there have been no attempts at successfully generalizing the model across datasets of novel camera views. This bottleneck significantly hinders the application whenever there is a new camera because it requires annotating new data to fine-tune the model, resulting in a procedure that is not only expensive but also tardy in deploying the model.

An ideal model should be able to disentangle human behavioral dynamics from specific camera views, positions, and scenes. It should produce robust trajectory prediction despite the variances in these camera settings. Motivated by this idea, in this work, we learn a robust representation for future trajectory prediction that can generalize to unseen video cameras. Different from existing works, we study a *real-data-free* setting where a model is trained only on synthetic data but tested, out of the box, on unseen real-world videos, without further re-training or fine-tuning the model. Following the success of learning from simulation [45, 57, 182, 187, 229, 263], our synthetic data is generalized from a 3D simulator, called CARLA [50], which anchors to the static scene and dynamic elements in the VIRAT/ActEV videos [158]. By virtue of the 3D simulator, we can generate multiple views and pixel-precise semantic segmentation labels for each training trajectory, as illustrated in Figure 6.1. Meanwhile, following the previous works [132, 191], scene semantic segmentation is used instead of RGB pixels to alleviate the influence of different lighting conditions, scene textures, subtle noises produced by camera sensors, etc. At test time, we extract scene features from real videos using pretrained segmentation models. The use of segmentation features is helpful but is insufficient for learning robust representation for future trajectory prediction.

To tackle this issue, we propose a novel data augmentation method called *SimAug* to augment the features of the simulation data with the goal of learning robust representation to various semantic scenes and camera views in real videos. To be specific, first, after representing each training trajectory by high-level scene semantic segmentation features, we defend

our model from adversarial examples generated by white-box attack methods [64]. Second, to overcome the changes in camera views, we generate multiple views for the same trajectory, and encourage the model to focus on overcoming the "hardest" view to which the model has learned. Following [93, 94], the classification loss is adopted and the view with the highest loss is favored during training. Finally, the augmented trajectory is computed as a convex combination of the trajectories generated in previous steps. Our trajectory prediction backbone model is built on a recent work called Multiverse [132]. The final model is trained to minimize the empirical vicinal risk over the distribution of augmented trajectories. Our method is partially inspired by recent robust deep learning methods using adversarial training [37, 113], Mixup [261], and MentorMix [94].

We empirically validate our model, which is trained only on simulation data, on three real-world benchmarks for future trajectory prediction: VIRAT/ActEV [7, 158], Stanford Drone [184], and Argoverse [25]. These benchmarks represent three distinct camera views: 45-degree view, top-down view and dashboard camera view with ego-motions. The results show our method performs favorably against baseline methods including standard data augmentation, adversarial learning, and imitation learning. Notably, our method achieves better results compared to the state-of-the-art on the VIRAT/ActEV and Stanford Drone benchmark. Our code and models are released at `https://next.cs.cmu.edu/simaug`. To summarize, our contribution is threefold:

- We study a new setting of future trajectory prediction in which the model is trained only on synthetic data and tested, out of the box, on any unseen real video with novel views or scenes.
- We propose a novel and effective approach to augment the representation of trajectory prediction models using multi-view simulation data.
- Ours is the first work on future trajectory prediction to demonstrate the efficacy of training on 3D simulation data, and establishes new state-of-the-art results on three public benchmarks.

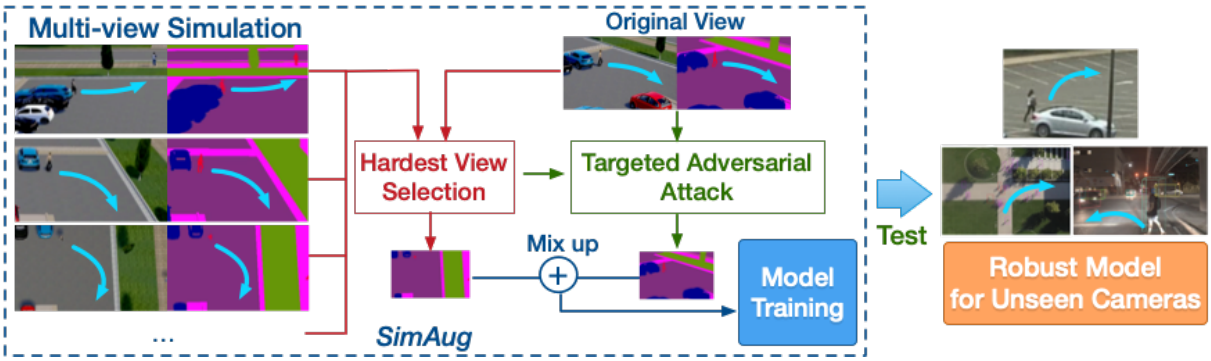

Figure 6.2: Overview of our method *SimAug* that is trained on simulation and tested on real unseen videos. Each training trajectory is represented by multi-view segmentation features extracted from the simulator. *SimAug* mixes the feature of the hardest camera view with the adversarial feature of the original view.

6.2 The *SimAug* Model

6.3 Approach

In this section, we describe our approach to learn robust representation for future trajectory prediction, which we call *SimAug*. Our goal is to train a model only on simulation training data that can effectively predict the future trajectory in the real-world test videos that are unseen during training.

6.3.1 Problem Formulation

We focus on predicting the locations of a single agent for multiple steps into the future. Given a sequence of historical video frames $V_{1:h}$ of the past h steps and the past agent locations $L_{1:h}$ in training, we learn a probabilistic model on simulation data to estimate $P(L_{h+1:T}|L_{1:h}, V_{1:h})$ for $T-h$ steps into the future. At test time, our model takes as input an agent's observable past $(V_{1:h}, L_{1:h})$ in real videos to predict the agent's future locations $L_{h+1:T} = \{y_{h+1}, \ldots, y_T\}$, where y_t is the location coordinates. As the test real videos are unseen during training, the model is supposed to be invariant to the variances in semantic scenes, camera views, and camera motions.

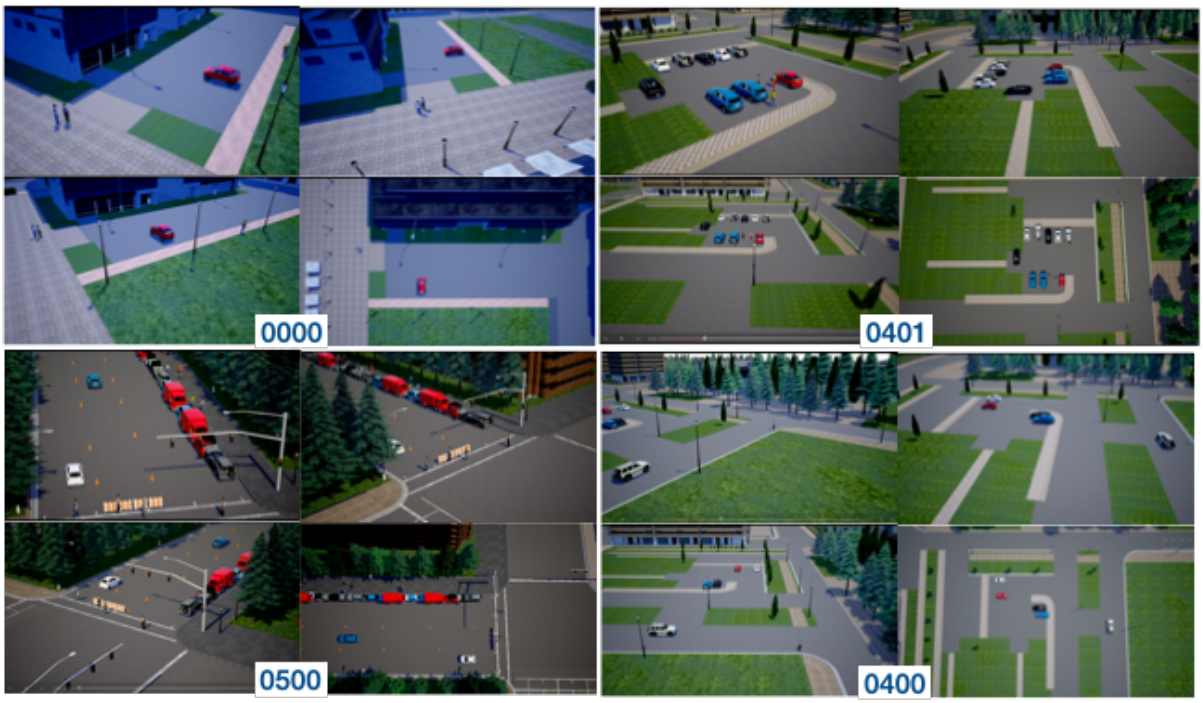

Figure 6.3: Visualization of the multi-view 3D simulation data used in *SimAug* training. Data generation process is described in Section 6.3.2. We use 4 camera views from 4 scenes defined in [132]. "0400" and "0401" scene have overlapping views. The top-left views are the original views from VIRAT/ActEV dataset.

6.3.2 Training Data Generation From Simulation

Our model is trained only on simulation data. To ensure high-level realism, the training trajectories are generated by CARLA [50], an open-source 3D simulator built on top of the state-of-the-art game engine *Unreal Engine 4*. We use the trajectories from the Forking Paths dataset [132] that are semi-manually recreated from the VIRAT/ActEV benchmark that projects real-world annotations to the 3D simulation world. Note that it is not our intention to build an exact replica of the real-world scene, nor it is necessary to help train a model for real-world tasks as suggested in previous works [57, 132, 185, 263].

With CARLA, we record multiple views of the same trajectory of different camera angles and positions. For a trajectory $(V_{1:T}, L_{1:T})$ in original view, let $\mathcal{S} = \{(V_{1:T}^{(i)}, L_{1:T}^{(i)})\}_{i=1}^{|\mathcal{S}|}$ denote a set of additional views for the same trajectory. In our experiments, we use four camera parameters pre-specified in [132], including three 45-degree views and one top-down view. We use a total of 4 scenes shown in Fig. 6.3. The ground-truth location varies under different camera views i.e. $L_{1:T}^{(i)} \neq L_{1:T}^{(j)}$ for $i \neq j$. Note that these camera positions and angles are defined in [132] specifically for VIRAT/ActEV dataset. The top-down view cameras in Stanford Drone dataset [184] are still considered unseen to the model since the scenes and camera positions are quite different.

In simulation, we also collect the ground-truth scene semantic segmentation for $K = 13$ classes including sidewalk, road, vehicle, pedestrian, etc. At test time, we extract the semantic segmentation feature from real videos using a pre-trained model with the same number of class labels per pixel. To be specific, we use the Deeplab model [31] trained on the ADE20k [269] dataset and keep its weights frozen. To bridge the gap between real and simulated video frames, we represent all trajectory $V_{1:T}$ as a sequence of scene semantic segmentation features, following previous works [47, 131, 132, 191]. As we show in our experiments, the use of segmentation features is helpful but is still insufficient for learning the robust representation.

6.3.3 Multi-view Simulation Augmentation (*SimAug*)

In this subsection, we first describe *SimAug* for learning robust representations. Our trajectory prediction backbone model is built on the Multiverse model [132] and will be discussed in Section 6.3.4.

Given a trajectory in its original view $(V_{1:T}, L_{1:T})$, we generate a set of additional views in $\mathcal{S} = \{(V_{1:T}^{(i)}, L_{1:T}^{(i)})\}_{i=1}^{|\mathcal{S}|}$ as described in the previous section, where $V_t^{(i)}$ represents the scene

semantic feature of view i at time t. $L_{1:T}^{(i)}$ is a sequence of ground-truth locations for the i-th view.

Each time given a camera view, we use it as an anchor to search for the "hardest" view that is most inconsistent with what the model has learned. Inspired by [93], we use the classification loss as the criteria and compute:

$$j^* = \underset{j \in [1, |\mathcal{S}|]}{\operatorname{argmax}} \{\mathcal{L}_{\text{cls}}(V_{1:h} + \delta, L_{h+1:T}^{(j)})\}, \tag{6.1}$$

where δ is the ℓ_∞-bounded random perturbation applied to the input features. $\mathcal{L}_{\text{cls}}$ is the location classification loss used in our backbone Multiverse model and will be discussed in the next subsection.

Then for the original view, we generate an adversarial trajectory by the targeted-FGSM attack [113]:

$$V_{1:h}^{adv} = V_{1:h} - \epsilon \cdot \text{sign}(\nabla_{V_{1:h}} \mathcal{L}_{\text{cls}}(V_{1:h} + \delta, L_{h+1:T}^{(j^*)})), \tag{6.2}$$

where ϵ is the hyper-parameter. The attack tries to make the model predict the future locations in the selected "hardest" camera view rather than the original view. In essence, the resulting adversarial feature is "warped" to the "hardest" camera view by a small perturbation. By defending against such adversarial trajectory, our model learns representations that are robust against variances in camera views.

Finally, we mix up the trajectory locations of the selected view and the adversarial trajectory locations by a convex combination function [261] over their features and one-hot location labels.

$$\begin{aligned} V_{1:h}^{aug} &= \lambda \cdot V_{1:h}^{adv} + (1 - \lambda) \cdot V_{1:h}^{(j^*)} \\ y_t^{aug} &= \lambda \cdot \text{one-hot}(y_t) + (1 - \lambda) \cdot \text{one-hot}(y_t^{(j^*)}) \quad \forall t \in [h+1, T] \\ L_{h+1:T}^{aug} &= [y_{h+1}^{aug}, \ldots, y_T^{aug}] \end{aligned} \tag{6.3}$$

where $[y_{h+1}, \cdots, y_T] = L_{h+1:T}$ are the ground-truth locations of the original view. The one-hot$(\cdot)$ function projects the location in xy coordinates into an one-hot embedding over the predefined grid used in our backbone trajectory prediction model. Please find the details in [132]. Following [261], λ is drawn from a Beta distribution Beta(α, α) controlled by the hyper-parameter α.

The algorithm for training with one training step is listed in Algorithm 2. To train robust models to various camera views and semantic scenes, we learn representations over augmented

training trajectories to overcome (i) feature perturbations (Step 3 and 5) (ii) targeted adversarial attack (Step 5), and (iii) the "hardest" feature from other views (Step 4). By the mix-up operation in Eq. (6.3), our model is trained to minimize the empirical vicinal risk over a new distribution constituted by the generated augmented trajectories, which is proved to be useful in improving model robustness to real-world distributions under various settings [94].

Algorithm 2 Multi-view Simulation Adversarial Augmentation (*SimAug*)

Input : Mini-batch of trajectories; hyper-parameters α and ϵ

Output: Classification loss $\mathcal{L}_{\text{cls}}$ computed over augmented trajectories

1 **for** *each trajectory* $(V_{1:T}, L_{1:T})$ *in the mini-batch* **do**
2 Generate trajectories from additional views $\mathcal{S} = \{(V_{1:T}^{(i)}, L_{1:T}^{(i)})\}$
3 Compute the loss for each camera view $\mathcal{L}_{\text{cls}}(V_{1:h} + \delta, L_{h+1:T}^{(j)})$
4 Select the view with the largest loss j^* by Eq. (6.1)
5 Generate an adversarial trajectory $V_{1:h}^{adv}$ by Eq. (6.2)
6 Mix up $(V_{1:h}^{adv}, L_{h+1:T})$ and $(V_{1:h}^{(j^*)}, L_{h+1:T}^{(j*)})$ by Eq. (6.3)
7 Compute $\mathcal{L}_{\text{cls}}$ over the augmented trajectory $(V_{1:h}^{aug}, L_{h+1:T}^{aug})$ from Step 6
8 **end**
9 **return** averaged $\mathcal{L}_{\text{cls}}$ over the augmented trajectories

6.3.4 Backbone Model for Trajectory Prediction

We employ Multiverse [132] as our backbone network, a state-of-the-art multi-future trajectory prediction model. Although we showcase the use of *SimAug* to improve the robustness of Multiverse, *SimAug* is a general approach that can be applied to other trajectory prediction models.

Input Features. The model is given the past locations, $L_{1:h}$, and the scene, $V_{1:h}$. Each ground-truth location L_t is encoded by an one-hot vector $y_t \in \mathbb{R}^{HW}$ representing the nearest cell in a 2D grid of size $H \times W$. In our experiment, we use a grid scale of 36×18. Each video frame V_t is encoded as semantic segmentation feature of size $H \times W \times K$ where $K = 13$ is the total number of class labels as in [131, 132]. As discussed in the previous section, we use *SimAug* to generate augmented trajectories $(V_{1:h}^{aug}, L_{1:h}^{aug})$ as our training features.

History Encoder. A convolutional RNN [238, 246] is used to get the final spatial-temporal feature state $H_t \in \mathbb{R}^{H \times W \times d_{enc}}$, where d_{enc} is the hidden size. The context is a concatenation of the last hidden state and the historical video frames, $\mathcal{H} = [H_h, V_{1:h}]$.

Location Decoder. After getting the context $\mathcal{H}$, a coarse location decoder is used to predict locations at the level of grid cells at each time-instant by:

$$\hat{y}_t = \text{softmax}(f_c(\mathcal{H}, H^c_{t-1})) \in \mathbb{R}^{HW} \tag{6.4}$$

where f_c is the convolutional recurrent neural network (ConvRNN) with graph attention proposed in [132] and H^c_t is the hidden state of the ConvRNN. Then a fine location decoder is used to predict a continuous offset in $\mathbb{R}^2$, which specifies a "delta" from the center of each grid cell, to get a fine-grained location prediction:

$$\hat{O}_t = \text{MLP}(f_o(\mathcal{H}, H^o_{t-1})) \in \mathbb{R}^{HW \times 2}, \tag{6.5}$$

where f_o is a separate ConvRNN and H^o_t is its hidden state. To compute the final prediction location, we use

$$\hat{L}_t = Q_g + \hat{O}_{tg} \tag{6.6}$$

where $g = \operatorname{argmax} \hat{y}_t$ is the index of the selected grid cell, $Q_g \in \mathbb{R}^2$ is the center of that cell, and $\hat{O}_{tg} \in \mathbb{R}^2$ is the predicted offset for that cell at time t.

Training. We use *SimAug* (see Section 6.3.3) to generate $L^{aug}_{h+1:T} = \{y^{aug}_{h+1}, \ldots, y^{aug}_T\}$ as labels for training. For the coarse decoder, the cross-entropy loss is used:

$$\mathcal{L}_{\text{cls}} = -\frac{1}{T} \sum_{t=h+1}^{T} \sum_{c=1}^{HW} y^{aug}_{tc} \log(\hat{y}_{tc}) \tag{6.7}$$

For the fine decoder, we use the original ground-truth location label $L_{h+1:T}$:

$$\mathcal{L}_{\text{reg}} = \frac{1}{T} \sum_{t=h+1}^{T} \sum_{c=1}^{HW} \text{smooth}_{l_1}(O_{tc}, \hat{O}_{tc}) \tag{6.8}$$

where $O_{tc} = L_t - Q_c$ is the delta between the ground true location and the center of the c-th grid cell. The final loss is then calculated using

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{cls}} + \lambda_1 \mathcal{L}_{\text{reg}} + \lambda_2 \|\theta\|_2^2 \tag{6.9}$$

where λ_2 controls the ℓ_2 regularization (weight decay), and $\lambda_1 = 0.5$ is used to balance the regression and classification losses.

6.4 Experiments

In this section, we evaluate various methods, including our *SimAug* method, on three public video benchmarks of real-world videos captured under different camera views and scenes: the VIRAT/ActEV [7, 158] dataset, the Stanford Drone Dataset (SDD) [184], and the autonomous driving dataset Argoverse [25]. We demonstrate the efficacy of our method for unseen cameras in Section 6.4.2 and how our method can also improve state-of-the-art when fine-tuned on the real training data in Section 6.4.3 and Section 6.4.4.

6.4.1 Evaluation Metrics

Following prior works [3, 132], we utilize two common metrics for trajectory prediction evaluation. Let $L^i = L^i_{t=(h+1)\cdots T}$ be the true future trajectory for the i^{th} test sample, and $\hat{L}^{ik}$ be the corresponding k^{th} prediction sample, for $k \in [1, K]$.
i) *Minimum Average Displacement Error Given K Predictions* (minADE_K): for each true trajectory sample i, we select the closest K predictions, and then measure its average error:

$$\text{minADE}_K = \frac{\sum_{i=1}^{N} \min_{k=1}^{K} \sum_{t=h+1}^{T} \|L^i_t - \hat{L}^{ik}_t\|_2}{N \times (T-h)} \tag{6.10}$$

ii) *Minimum Final Displacement Error Given K Predictions* (minFDE_K): similar to minADE_K, but we only consider the predicted points and the ground truth point at the final prediction time instant:

$$\text{minFDE}_K = \frac{\sum_{i=1}^{N} \min_{k=1}^{K} \|L^i_T - \hat{L}^{ik}_T\|_2}{N} \tag{6.11}$$

iii) *Grid Prediction Accuracy* (Grid_Acc): As our base model also predicts coarse grid locations as described in Section 6.3.4, we also evaluate the accuracy between the predicted grid $\hat{y}_t$ and the ground truth grid y_t. This is an intermediate metric and hence is less indicative than the minADE_K and minFDE_K.

6.4.2 Main Results

Dataset & Setups. We compare *SimAug* with classical data augmentation methods as well as adversarial learning methods to train robust representations. All methods are trained using the same backbone on the same *simulation training data* described in Section 6.3.2, and tested on the same benchmarks. Real videos are not allowed to be used during training except in our finetuning experiments. For VIRAT/ActEV and SDD, we use the standard test split as in

[131, 132] and [47, 191], respectively. For Argoverse, we use the official validation set from the 3D tracking task, and the videos from the "ring_front_center" camera are used.

These datasets have different levels of difficulties. VIRAT/ActEV is the easiest one because its training trajectories have been projected in the simulation training data. SDD is more difficult as its camera positions and scenes are different from the training data. Argoverse is the most challenging one with distinct scenes, camera views, and ego-motions.

Following the setting in previous works [3, 3, 47, 67, 67, 131, 132, 146, 191], the models observe 3.2 seconds (8 frames) of every pedestrian and predict the future 4.8 seconds (12 frames) of the person trajectory. We use the pixel values for the trajectory coordinates as it is done in [11, 24, 47, 76, 117, 121, 131, 146, 253]. By default, we evaluate the top $K = 1$ future trajectory prediction of all models.

Baseline methods. We compare *SimAug* with the following baseline methods for learning robust representations. All methods are built on the same backbone network and trained using the same simulation training data. *Base Model* is the trajectory prediction model proposed in [132]. *Standard Aug* is the base model trained with standard data augmentation techniques including horizontal flipping and random input jittering. *Fast Gradient Sign Method (FGSM)* is the base model trained with adversarial examples generated by the targeted-FGSM attack method [64]. Random labels are used for the targeted-FGSM attack. *Projected Gradient Descent (PGD)* is learned with an iterative adversarial learning method [145, 244]. The number of iterations is set to 10 and other hyper-parameters following [244].

Implementation Details. We use $\alpha = 0.2$ for the Beta distribution in Eq (6.3) and we use $\epsilon = \delta = 0.1$ in Eq (6.2). As the random perturbation is small, we do not normalize the perturbed features and the normalized features yield comparable results. All models are trained using Adadelta optimizer [259] with an initial learning rate of 0.3 and a weight decay of 0.001. Other hyper-parameters for the baselines are the same as the ones in [132].

Quantitative Results. Table 6.1 shows the evaluation results. Our method performs favorably against other baseline methods across all evaluation metrics on all three benchmarks. In particular, "Standard Aug" seems to be not generalizing well to unseen cameras. FGSM improves significantly on the "Grid_Acc" metric but fails to translate the improvement to final location predictions. *SimAug* is able to improve the model overall stemming from the effective use of multi-view simulation data. All other methods are unable to improve trajectory prediction on Argoverse, whose data characteristics include ego-motions and distinct dashboard-view cameras. The results substantiate the efficacy of *SimAug* for future trajectory prediction in unseen cameras. Note as the baseline methods use the same features as ours, the results indicate the

Table 6.1: Comparison to the standard data augmentation method and recent adversarial learning methods on three datasets. We report three metrics: Grid_Acc(↑)/minADE$_1$(↓)/minFDE$_1$(↓). The units of ADE/FDE are pixels. All methods are built on the same backbone model in [132] and trained using the same multi-view simulation data described in Section 6.3.2.

Method	VIRAT/ActEV	Stanford Drone	Argoverse
Base Model [132]	44.2%/26.2/49.7	31.4%/21.9/42.8	26.6%/69.1/183.9
Standard Aug	45.5%/25.8/48.3	21.3%/23.7/47.6	28.9%/70.9/183.4
PGD [145, 244]	47.5%/25.1/48.4	28.5%/21.0/42.2	25.9%/72.8/184.0
FGSM [64]	48.6%/25.4/49.3	42.3%/19.3/39.9	29.2%/71.1/185.4
SimAug	**51.1%/21.7/42.2**	**45.4%/15.7/30.2**	**30.9%/67.9/175.6**

use of segmentation features is insufficient for learning robust representations.

Qualitative Analysis. We visualize outputs of the base model with and without *SimAug* in Fig. 6.4. We show visualizations on all three datasets. In each image, the yellow trajectories denote historical trajectories and the green ones are ground truth future trajectories. Outputs of the base model without SimAug are colored with blue heatmaps and the yellow-orange heatmaps are from the same model with SimAug. As we see, the base model with SimAug augmentation yields more accurate trajectories for turnings (Fig. 6.4 1a., 3a.) while without it the model sometimes predicts the wrong turns (Fig. 6.4 1b., 1c., 2a., 3a., 3b.). In addition, the length of SimAug predictions is more accurate (Fig. 6.4 1d., 2b., 2c., 2d.).

6.4.3 State-of-the-Art Comparison on Stanford Drone Dataset

In this section, we compare our *SimAug* model with the state-of-the-art generative models, including Social-LSTM [3], Social-GAN [67], DESIRE [117], and SoPhie [191]. We also compare with the imitation learning model, IDL [121], and the inverse reinforcement learning model, P2T$_{\text{IRL}}$ [47] for trajectory prediction on the Stanford Drone Dataset. Following previous works, we evaluate the minimal errors over $K = 20$ predictions.

Results & Analysis. The results are shown in Table 6.3, where *SimAug* is built on top of the *Multiverse* model. As it shows, *SimAug* model trained only on the simulation data (second to the last row) achieves comparable or even better performance than other state-of-the-art models that are trained on in-domain real videos. By further fine-tuning on the learned representations of *SimAug*, we achieve the state-of-the-art performance on the Stanford Drone Dataset. The

Table 6.2: State-of-the-art comparison on the VIRAT/ActEV dataset. Numbers are minimal errors over 1 predictions and lower the better.

Method	$\text{minADE}_1(\downarrow)$	minFDE_1 $(\downarrow)$
Social-LSTM [3]	23.10	44.27
Social-GAN [67]	30.42	60.70
Next [131]	19.78	42.43
Multiverse [132]	18.51	35.84
Multiverse (Trained on Sim.) [132]	22.94	43.35
SimAug	21.73	42.22
SimAug + finetune	**17.96**	**34.68**

promising results demonstrate the efficacy of *SimAug* for future trajectory prediction in unseen cameras.

6.4.4 State-of-the-Art Comparison on VIRAT/ActEV

In this section, we compare our *SimAug* model with state-of-the-art models on VIRAT/ActEV. Following the previous work [132], we compute the errors for the top $K = 1$ prediction. Experimental results are shown in Table 6.2 (b), where all models in the top four rows are trained on the real-world training videos in VIRAT/ActEV. Our model trained on simulation data achieves competitive performance and outperforms *Multiverse* [132] model that is trained on the same data. With fine-tuning, which means using exactly the same training data without any extra annotation of real trajectories compared to [3, 67, 131, 132], we achieve the best performance on the VIRAT/ActEV benchmark.

6.4.5 Ablation Experiments

We test various ablations of our approach to validate our design decisions. Results are shown in Table 6.4, where the top-1 prediction is used in the evaluation. We verify four key design choices by removing each, at a time, from the full model. The results show that by introducing viewpoint selection (Eq. (6.1)) and adversarial perturbation (Eq. (6.2)), our method improves model generalization.

(1) *Multi-view data:* Our method is trained on multi-view simulation data and we use 4

Table 6.3: State-of-the-art comparison on the Stanford Drone Dataset (SDD). Numbers are minimal errors over 20 predictions and lower the better. Baseline numbers are taken from [47, 191]. "SimAug" is trained without using SDD training data and "SimAug + finetune" is further finetuned on SDD training data.

Method	$\text{minADE}_{20}(\downarrow)$	minFDE_{20} $(\downarrow)$
Social-LSTM [3]	31.19	56.97
Social-GAN [67]	27.25	41.44
DESIRE [117]	19.25	34.05
SoPhie [191]	16.27	29.38
Multiverse [132]	14.78	27.09
IDL [121]	13.93	24.40
P2T_{IRL} [47]	12.58	22.07
SimAug	12.03	23.98
SimAug + finetune	**10.27**	**19.71**

camera views in our experiments. We test our method without the top-down view because it is similar to the ones in the SDD dataset. As we see, the performance drops due to the fewer number of data and less diverse views, suggesting that we should use more views in augmentation (which is effortless to do in 3D simulators).

(2) *Random perturbation:* We test our model without random perturbation on the original view trajectory samples by setting $\delta = 0$ in Eq. (6.1). This leads to the performance drop on all three datasets and particularly on the more difficult Argoverse dataset.

(3) *Adversarial attack:* We test our model without adversarial attack by replacing Eq. (6.2) with $V_{1:h}^{adv} = V_{1:h}$. This is similar to applying the Mixup method [261] to two views in the feature space. The performance drops across all three benchmarks.

(4) *View selection:* We replace Eq. (6.1) with random search to see the effect of view selection. As we see, the significant performance drops, especially on the Stanford Drone dataset, verifying the effectiveness of this design.

Table 6.4: Performance on ablated versions of our method on three benchmarks. We report the $\text{minADE}_1(\downarrow)/\text{minFDE}_1(\downarrow)$ metrics.

Method	VIRAT/ActEV	Stanford Drone	Argoverse
SimAug full model	21.7 / 42.2	15.7 / 30.2	67.9 / 175.6
- top-down view data	22.8 / 43.6	18.4 / 35.6	68.4 / 178.3
- random perturbation	23.6 / 43.8	18.7 / 35.6	69.1 / 180.2
- adversarial attack	23.1 / 43.8	17.4 / 32.9	68.0 / 177.5
- view selection	23.0 / 42.9	19.6 / 38.2	68.6 / 177.0

6.5 Related Work

This work studies the novel direction of robustness in trajectory prediction model [186]. We are the first to look into the generalization of trajectory prediction model and study the invariant representation for different environments and camera views. In the following, we also review some of the trajectory prediction works based on their specific camera views. We then review more broadly of learning from simulation and robust model learning.

Trajectory prediction. Recently there is a large body of work on predicting person future trajectories in a variety of scenarios. Many works [3, 131, 132, 191, 251, 262] focused on modeling person motions in videos recorded with 45-degree-view stationary cameras. Datasets like VIRAT/ActEV [158], ETH/UCY [118, 140] have been used for such direction. Meanwhile, many works [11, 76, 117, 121, 146, 180, 192, 266] have been proposed for top-down view videos for trajectory prediction. Notably, the Stanford Drone Dataset (SDD) [184] is used in many works [47, 121, 191] for trajectory prediction with drone videos. Other works have also looked into pedestrian prediction in dashcam videos [105, 117, 148, 207] and first-person videos [208, 253]. Many vehicle trajectory datasets [18, 25, 257] have been proposed as a result of self-driving's surging popularity.

Learning from 3D simulation data. As the increasing research focus in 3D computer vision [50, 72, 124, 182, 185, 195, 264], many research works have used 3D simulation for training and evaluating real-world tasks [10, 29, 45, 57, 98, 132, 211, 213, 242, 271]. Many works [45, 57, 171] were proposed to use data generated from 3D simulation for video object detection, tracking, and action recognition analysis. For example, Sun et al. [211] proposed a forecasting model by using a gaming simulator. AirSim [195] and CARLA [50] were proposed for

robotic autonomous controls for drones and vehicles. Zeng et al. [260] proposed to use 3D simulation for adversarial attacks. RSA [263] used randomized simulation data for human action recognition. The ForkingPaths dataset [132] was proposed for evaluating multi-future trajectory prediction. Human annotators were asked to control agents in a 3D simulator to create a multi-future trajectory dataset.

Robust Deep Learning. Traditional domain adaptation approaches [17, 58, 97, 226] may not be applicable as our target domain is considered "unseen" during training. Methods for learning using privileged information [114, 139, 141, 227] is not applicable for a similar reason. Closest to ours is robust deep learning methods. In particular, our approach is inspired by the following methods: (i) *adversarial training* [64, 145, 244, 260] to defend the adversarial attacks generated on-the-fly during training using gradient-based methods [36, 64, 145, 222]; (ii) data augmentation methods to overcome unknown variances between training and test examples such as Mixup [261], MentorMix [94], AugMix [37], etc; (iii) example re-weighting or selection [91, 93, 123, 157, 177] to mitigate network memorization. Different from prior work, ours uses 3D simulation data as a new perspective for data augmentation and is carefully designed for future trajectory prediction.

6.6 Summary

In this chapter, we have introduced *STAN*, a novel simulation data augmentation method to learn robust representations for trajectory prediction. Our model is trained only on 3D simulation data and applied out-of-the-box to a wide variety of real video cameras with novel views or scenes. We have shown that our method achieves competitive performance on three public benchmarks with and without using the real-world training data. We believe our approach will facilitate future research and applications on learning robust representation for trajectory prediction with limited or zero training data. Other directions to deal with camera view dependence include using a homography matrix, which may require an additional step of manual or automatic calibration of multiple cameras. We leave them for future work.

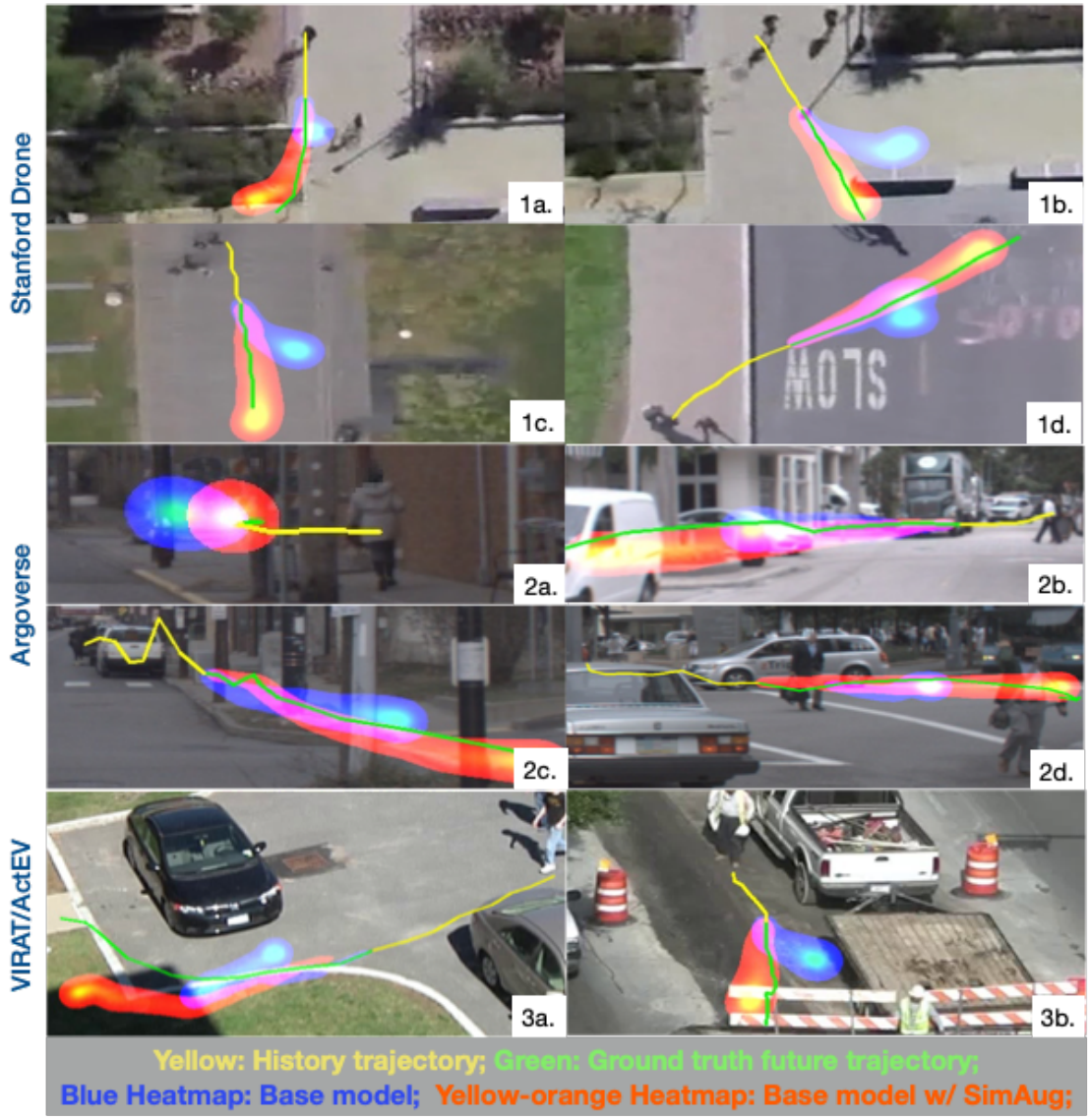

Figure 6.4: Qualitative analysis. Trajectory predictions from different models are colored and overlaid in the same image. See text for details.

Part III

Joint Analysis of Human Actions and Trajectory Prediction

In the final part, we aim to build robust prediction model with enhanced contextual cues from scene semantics and human actions through multi-task learning. We first study jointly predicting short-term pedestrian trajectories and activities on common benchmarks (chapter 7). Since short-term future prediction is not enough to ensure safe operations in autonomous driving applications, we introduce a new, human-annotated long-term trajectory and action prediction benchmark with multi-view camera data in urban traffic scenes. (chapter 8). Finally, we utilize our findings from the first two parts and develop a robust model for the aforementioned challenging long-term trajectory prediction task (chapter 9).

Chapter 7

Joint Pedestrian Trajectory and Action Prediction on Common Benchmarks

In this chapter, we explore joint trajectory and action prediction in common benchmarks for short-term prediction (short-term means prediction 3-5 seconds into the future). We propose the *Next* model [131] for pedestrian prediction which utilizes rich visual features and multi-task learning.

7.1 Motivation

With the advancement in deep learning, systems now are able to analyze an unprecedented amount of rich visual information from videos. An important analysis is forecasting the future path of pedestrians, called future person trajectory prediction. This problem has received increasing attention in the computer vision community [3, 67, 103]. It is regarded as an essential building block in video understanding because looking at the visual information from the past to predict the future is useful in many applications like self-driving cars, socially-aware robots [140], etc.

Humans navigate through public spaces often with specific purposes in mind, ranging from simple ones like entering a room to more complicated ones like putting things into a car. Such intention, however, is mostly neglected in existing work. Consider the example in Fig. 7.1, the person (at the top-right corner) might take different paths depending on their intention, e.g., they might take the green path to *transfer object* or the yellow path to *load object into the car*. Inspired by this, this work is interested in modeling the future path jointly with such intention in videos. We model the intention in terms of a predefined set of 29 activities provided by the

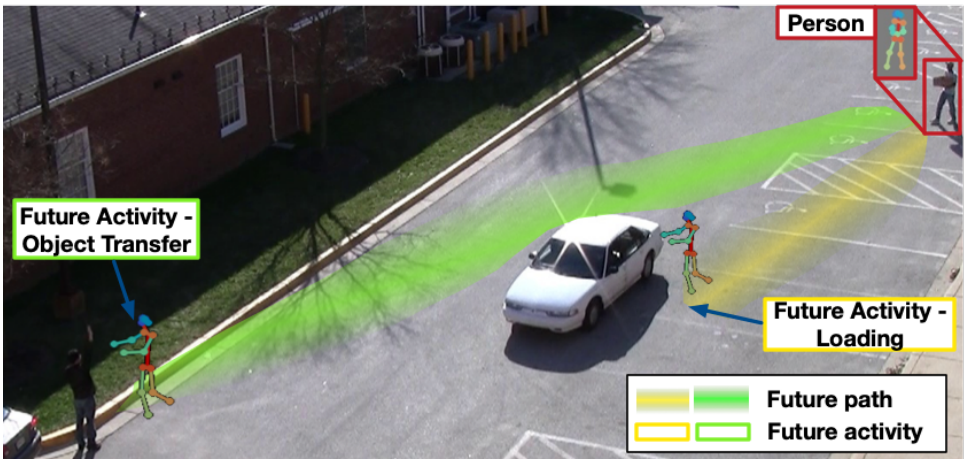

Figure 7.1: Our goal is to jointly predict a person's future path and activity. The green and yellow line show two possible future trajectories and two possible activities are shown in the green and yellow boxes. Depending on the future activity, the person (top right) may take different paths, e.g., the yellow path for "loading" and the green path for "object transfer".

NIST such as "loading", "object transfer", etc. See supplementary material for the full list.

The joint prediction model can have two benefits. First, learning the activity together with the path may benefit the future path prediction. Intuitively, humans are able to read from others' body language to anticipate whether they are going to cross the street or continue walking along the sidewalk. In the example of Fig. 7.1, the person is carrying a box, and the man at the bottom left corner is waving at the person. Based on common sense, we may agree that the person will take the green path instead of the yellow path. Second, the joint model advances the capability of understanding not only the future path but also the future activity by taking into account the rich semantic context in videos. This increases the capabilities of automated video analytics for social good, such as safety applications like anticipating pedestrian movement at traffic intersections or a road robot helping humans transport goods to a car. Note that our techniques focus on predicting a few seconds into the future, and should not be useful for non-routine activities.

To this end, we propose a multi-task learning model called *Next* which has prediction modules for learning future paths and future activities simultaneously. As predicting future activity is challenging, we introduce two new techniques to address the issue. First, unlike most of the existing work [3, 67, 103, 149, 191, 245] which oversimplifies a person as a point in space, we encode a person through rich semantic features about visual appearance, body movement and interaction with the surroundings, motivated by the fact that humans derive such predictions

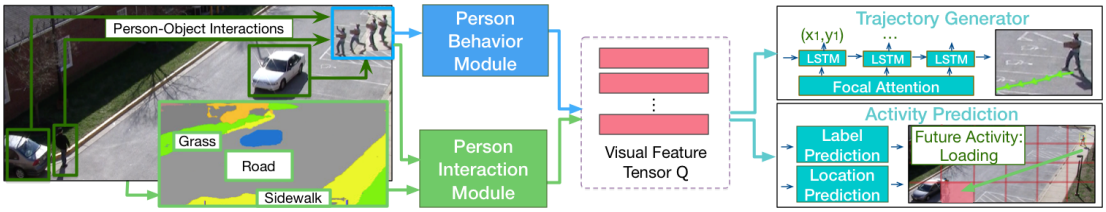

Figure 7.2: Overview of our model. Given a sequence of frames containing the person for prediction, our model utilizes person behavior module and person interaction module to encode rich visual semantics into a feature tensor.

by relying on similar visual cues. Second, to facilitate the training, we introduce an auxiliary task for future activity prediction, i.e., activity location prediction. In the auxiliary task, we design a discretized grid which we call the Manhattan Grid as location prediction target for the system.

To the best of our knowledge, our work is the first on joint future path and activity prediction in streaming videos, and more importantly the first to demonstrate such joint modeling can considerably improve the future path prediction. We empirically validate our model on two benchmarks: ETH & UCY [118, 162], and ActEV/VIRAT [7, 158]. Experimental results show that our method outperforms state-of-the-art baselines, achieving the best-published result on two common benchmarks and producing additional prediction about the future activity. To summarize, the contributions of this work are threefold: **(i)** We conduct a pilot study on joint future path and activity prediction in videos. We are the first to empirically demonstrate the benefit of such joint learning. **(ii)** We propose a multi-task learning framework with new techniques to tackle the challenge of joint future path and activity prediction. **(iii)** Our model achieves the best-published performance on two public benchmarks. Ablation studies are conducted to verify the contribution of the proposed sub-modules.

7.2 The *Next* Model

Humans navigate through spaces often with specific purposes in mind. Such purposes may considerably orient the future trajectory/path. This motivates us to study the future path prediction jointly with the intention. In this work, we model the intention in terms of a predefined set of future activities such as “walk”, “open_door”, “talk”, etc.

Problem Formulation: Following [3, 67, 191], we assume each scene is first processed to obtain the spatial coordinates of all people at different time instants. Based on the coordinates,

we can automatically extract their bounding boxes. Our system observes the bounding box of all the people from time 1 to T_{obs}, and objects if there are any, and predicts their positions (in terms of xy-coordinates) for time T_{obs+1} to T_{pred}, meanwhile estimating the possibilities of future activity labels at time T_{pred}.

7.2.1 Network Architecture

Fig. 7.2 shows the overall network architecture of our *Next* model. Unlike most of the existing work [3, 67, 103, 149, 191, 245] which oversimplifies a person as a point in space, our model employs two modules to encode rich visual information about each person's behavior and interaction with the surroundings. *Next* has the following key components:

Person behavior module extracts visual information from the behavioral sequence of the person.

Person interaction module looks at the interaction between a person and their surroundings.

Trajectory generator summarizes the encoded visual features and predicts the future trajectory by the LSTM decoder with focal attention [128, 130].

Activity prediction utilizes rich visual semantics to predict the future activity label for the person. In addition, we divide the scene into a discretized grid of multiple scales, which we call the Manhattan Grid, to compute classification and regression for robust activity location prediction.

In the rest of this section, we will introduce the above modules and the learning objective in details.

7.2.2 Person Behavior Module

This module encodes the visual information about every individual in a scene. As opposed to oversimplifying a person as a point in space, we model the person's the appearance and body movement. To model appearance changes of a person, we utilize a pre-trained object detection model with "RoIAlign" [71] to extract fixed size CNN features for each person bounding box. See Fig. 7.3. We average the features along the spatial dimensions for each person and feed them into an LSTM encoder. Finally, we obtain a feature representation of $T_{obs} \times d$, where d is the hidden size of the LSTM. To capture the body movement, we utilize a person keypoint detection model trained on MSCOCO dataset [54] to extract person keypoint information. We apply the linear transformation to embed the keypoint coordinates before feeding into the LSTM encoder. The shape of the encoded feature has the shape of $T_{obs} \times d$. These appearance and movement

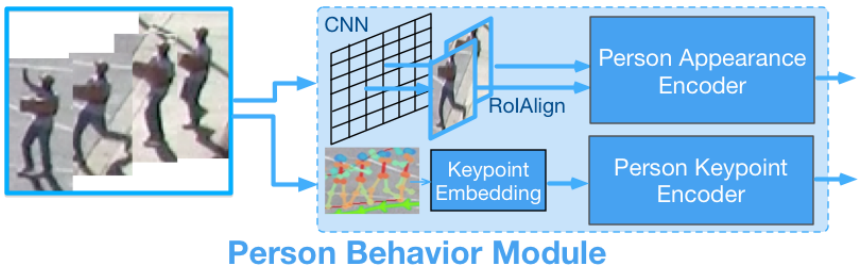

Figure 7.3: Person behavior module given a sequence of person frames.

features are commonly used in a wide variety of studies and thus do not introduce new concern on machine learning fairness.

7.2.3 Person Interaction Module

This module looks at the interaction between a person and their surroundings, i.e. person-scene and person-objects interactions.

Person-scene. To encode the nearby scene of a person, we first use a pre-trained scene segmentation model [32] to extract pixel-level scene semantic classes for each frame. We use totally $N_s = 10$ common scene classes, such as roads, sidewalks, etc. The scene semantic features are integers (class indexes) of the size $T_{obs} \times h \times w$, where h, w are the spatial resolution. We first transform the integer tensor into N_s binary masks (one mask for each class), and average along the temporal dimension. This results in N_s real-valued masks, each of the size of $h \times w$. We apply two convolutional layers on the mask feature with a stride of 2 to get the *scene CNN features* in two scales.

Given a person's xy-coordinate, we pool the scene features at the person's current location from the convolution feature map. As the example shown at the bottom of Fig. 7.4, the red part of the convolution feature is the discretized location of the person at the current time instant. The receptive field of the feature at each time instant, i.e. the size of the spatial window around the person which the model looks at, depends on which scale is being pooled from and the convolution kernel size. In our experiments, we set the scale to 1 and the kernel size to 3, which means our model looks at the 3-by-3 surrounding area of the person at each time instant. The person-scene representation for a person is in $\mathbb{R}^{T_{obs} \times C}$, where C is the number of channels in the convolution layer. We feed this into a LSTM encoder in order to capture the temporal information and get the final person-scene features in $\mathbb{R}^{T_{obs} \times d}$.

Person-objects. Unlike previous work [3, 67] which relies on LSTM hidden states to model

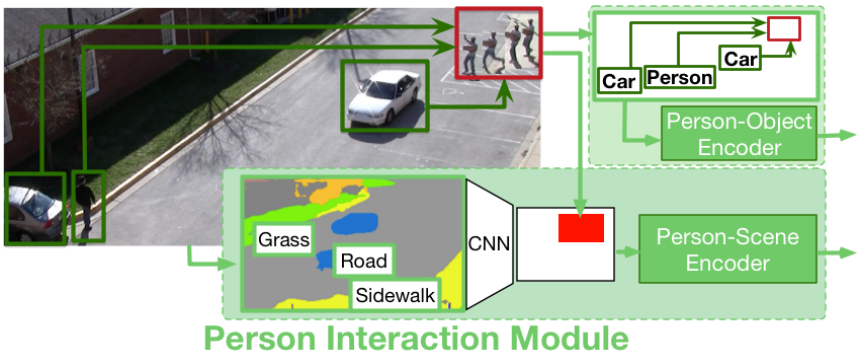

Figure 7.4: The person interaction module includes person-scene and person-objects modeling.

nearby people, our module explicitly models the *geometric relation* and the *object type* of all the objects/persons in the scene. At any time instant, given the observed box of a person (x_b, y_b, w_b, h_b) and K other objects/persons in the scene $\{(x_k, y_k, w_k, h_k)|k \in [1, K]\}$, we encode the geometric relation into $\mathcal{G} \in \mathbb{R}^{K\times 4}$, the k-th row of which equals to:

$$\mathcal{G}_k = [\log(\frac{|x_b - x_k|}{w_b}), \log(\frac{|y_b - y_k|}{h_b}), \log(\frac{w_k}{w_b}), \log(\frac{h_k}{h_b})] \tag{7.1}$$

This encoding computes the geometric relation in terms of the geometric distance and the fraction box size. We use a logarithmic function to reflect our observation that human trajectories are more likely to be affected by close-by objects or people. This encoding has been proven effective in object detection [78]. For the object type, we simply use one-hot encoding to get the feature in $\mathbb{R}^{K\times N_o}$, where N_o is the total number of object classes. We then embed the geometric features and the object type features at the current time into d_e-dimensional vectors and feed the embedded features into an LSTM encoder to obtain the final feature in $\mathbb{R}^{T_{obs}\times d}$.

As shown in the example from Fig. 7.4, the person-objects feature can capture how far away the person is to the other person and the cars. The person-scene feature can capture whether the person is near the sidewalk or grass. We design this information to the model with the hope of learning things like a person walks more often on the sidewalk than the grass and tends to avoid bumping into cars.

7.2.4 Trajectory Generation with Focal Attention

As discussed, the above four types of visual features, i.e. appearance, body movement, person-scene, and person-objects, are encoded by separate LSTM encoders into the same dimension. Besides, given a person's trajectory output from the last time instant, we extract the trajectory

embedding by

$$e_{t-1} = \tanh\{W_e[x_{t-1}, y_{t-1}]\} + b_e \in \mathbb{R}^d, \tag{7.2}$$

where $[x_{t-1}, y_{t-1}]$ is the trajectory prediction of time $t-1$ and W_e, b_e are learnable parameters. We then feed the embedding e_{t-1} into another LSTM encoder for the trajectory. The hidden states of all encoders are packed into a tensor named $Q \in \mathbb{R}^{M \times T_{obs} \times d}$, where $M = 5$ denotes the total number of features and d is the hidden size of the LSTM.

Following [67], we use an LSTM decoder to directly predict the future trajectory in the xy-coordinate. The hidden state of this decoder is initialized using the last state of the person's trajectory LSTM encoder. At each time instant, the xy-coordinate will be computed from the decoder state $h_t = \text{LSTM}(h_{t-1}, [e_{t-1}, \tilde{q}_t])$ and by a fully connected layer. $\tilde{q}_t$ is an important attended feature vector which summarizes salient cues in the input features Q. We employ an effective focal attention [128] to this end. It was originally proposed to carry out multimodal inference over a sequence of images for visual question answering. The key idea is to project multiple features into a space of correlation, where discriminative features can be easier to capture by the attention mechanism. To do so, we compute a correlation matrix $S^t \in \mathbb{R}^{M \times T_{obs}}$ at every time instant t, where each entry $S^t_{ij} = h^\top_{t-1} \cdot Q_{ij:}$ is measured using the dot product similarity and : is a slicing operator that extracts all elements from that dimension. Then we compute two attention matrices:

$$A^t = \text{softmax}(\max_{i=1}^{M} S^t_{i:}) \in \mathbb{R}^M \tag{7.3}$$

$$B^t = [\text{softmax}(S^t_{1:}), \cdots, \text{softmax}(S^t_{M:})] \in \mathbb{R}^{M \times T_{obs}} \tag{7.4}$$

Then the attended feature vector is given by:

$$\tilde{q}_t = \sum_{j=1}^{M} A^t_j \sum_{k=1}^{T_{obs}} B^t_{jk} Q_{jk:} \in \mathbb{R}^d \tag{7.5}$$

As shown, the focal attention models the correlation among different features and summarizes them into a low-dimensional attended vector.

7.2.5 Activity Prediction

Since the trajectory generation module outputs one location at a time, errors may accumulate across time and the final destination would deviate from the actual location. Using the wrong location for activity prediction may lead to bad accuracy. To counter this disadvantage, we

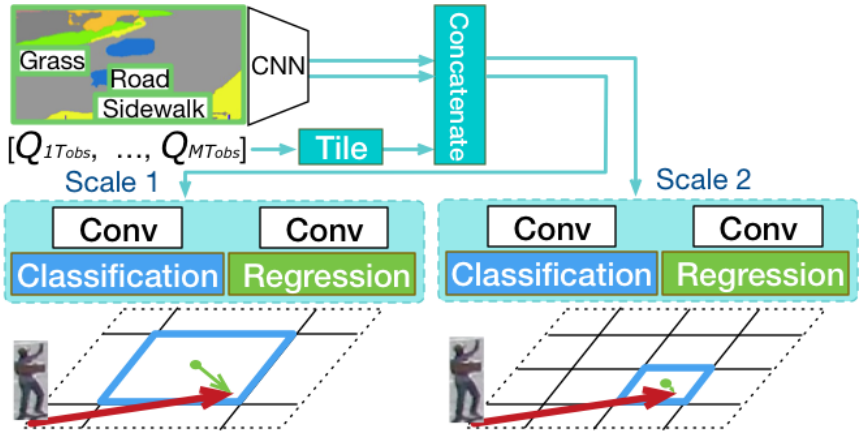

Figure 7.5: Activity location prediction with classification and regression on the multi-scale Manhattan Grid.

introduce an auxiliary task, i.e. activity location prediction, in addition to predicting the future activity label of the person. We describe the two prediction modules in the following.

Activity location prediction with the Manhattan Grid. To bridge the gap between trajectory generation and activity label prediction, we propose an activity location prediction module to predict the final location of where the person will engage in the future activity. The activity location prediction includes two tasks, *location classification* and *location regression*. As illustrated in Fig. 7.5, we first divide a video frame into a discretized $h \times w$ grid, namely *Manhattan Grid*, and learn to classify the correct grid block and at the same time to regress from the center of that grid block to the actual location. Specifically, the aim for the classification task is to predict the correct grid block in which the final location coordinates reside. After classifying the grid block, the aim for the regression task is to predict the deviation of the grid block center (green dots in the figure) to the final location coordinate (the end of green arrows). The reason for adding the regression task are: (i) it will provide more precise locations than just a grid block area; (ii) it is complementary to the trajectory prediction which requires xy-coordinates localization. We repeat this process on the Manhattan Grid of different scales and use separate prediction heads to model them. These prediction heads are trained end-to-end with the rest of the model. Our idea is partially inspired by the region proposal network [178] and our intuition is that similar to object detection problem, we need accurate localization using multi-scale features in a cost-efficient way.

As shown in Fig. 7.5, we first concatenate the scene CNN features (see Section 7.2.3) with the last hidden state of the encoders (see Section 7.2.4). For compatibility, we tile the hidden state

$Q_{:T_{obs}:}$ along the height and width dimension resulting in a tensor of the size $M \times d \times w \cdot h$, where $w \cdot h$ is the total number of the grid blocks. The hidden state contains rich information from all encoders and allow gradients flow smoothly through from prediction to feature encoders.

The concatenated features are fed into two separate convolution layers for classification and regression. The convolution output for grid classification $\mathsf{cls}_{grid} \in \mathbb{R}^{w \cdot h \times 1}$ indicates the probability of each grid block being the correct destination. In comparison, the convolution output for grid regression $\mathsf{rg}_{grid} \in \mathbb{R}^{w \cdot h \times 2}$ denotes the deviation, in the xy-coordinates, between the final destination and every grid block center. A row of rg_{grid} represents the difference to a grid block, calculated from $[x_t - x_{ci}, y_t - y_{ci}]$ where (x_t, y_t) denotes the predicted location and (x_{ci}, y_{ci}) is the center of the i-th grid block. The ground truth for the grid regression can be computed in a similar way. During training, only the correct grid block receives gradients for regression. Recent work [149] also incorporates the grid for location prediction. Our model differs in that we link grid locations to scene semantics, and use a classification layer and a regression layer together to make more robust predictions.

Activity label prediction. Given the encoded visual observation sequence, the activity label prediction module predicts the future activity at time instant T_{pred}. We compute the future N_a activity probabilities using the concatenated last hidden states of the encoders:

$$\mathsf{cls}_{act} = \text{softmax}(W_a \cdot [Q_{1T_{obs}:}, \cdots, Q_{MT_{obs}:}]) \tag{7.6}$$

where W_a is a learnable weight. The future activity of a person could be multi-class, e.g. a person could be "walking" and "carrying" at the same time.

7.2.6 Training

The entire network is trained end-to-end by minimizing a multi-task objective. The primary loss is the common L_2 loss between the predicted future trajectories and the ground-truth trajectories [67, 149, 191]. The loss is summed into L_{xy} over all persons from T_{obs+1} to T_{pred}.

The second category of loss is the activity location classification and regression loss. We have $L_{grid\_cls} = \sum_{i=1}^{N} \text{ce}(\mathsf{cls}^i_{grid}, \mathsf{cls}^{*i}_{grid})$, where cls^{*i}_{grid} is the ground-truth final location grid block ID for the i^{th} training trajectory. Likewise $L_{grid\_reg} = \sum_{i=1}^{N} \text{smooth}_{L_1}(\mathsf{rg}^i_{grid}, \mathsf{rg}^{*i}_{grid})$ and rg^{*i}_{grid} is the ground-truth difference to the correct grid block center. This loss is designed to bridge the gap between the trajectory generation task and activity label prediction task.

The third loss is for activity label prediction. We employ the cross-entropy loss: $L_{act} =$

$\sum_{i=1}^{N} \text{ce}(\text{cls}_{act}^{i}, \text{cls}_{act}^{*i})$. The final loss is then calculated from:

$$L = L_{xy} + \lambda(L_{grid\_cls} + L_{grid\_reg}) + L_{act} \tag{7.7}$$

We use a balance controller $\lambda = 0.1$ for location destination prediction to offset their higher loss values during training.

7.3 Experimental Results

We evaluate the proposed *Next* model on two common benchmarks for future path prediction: ETH [162] and UCY [118], and ActEV/VIRAT [7, 158].

7.3.1 ActEV/VIRAT

Dataset & Setups. ActEV/VIRAT [7] is a public dataset released by NIST in 2018 for activity detection research in streaming video (https://actev.nist.gov/). This dataset is an improved version of VIRAT [158], with more videos and annotations. It includes 455 videos at 30 fps from 12 scenes, more than 12 hours of recordings. Most of the videos have a high resolution of 1920x1080. We use the official training set for training and the official validation set for testing. Following [3, 67, 191], the models observe 3.2 seconds (8 frames) of every person and predict the future 4.8 seconds (12 frames) of person trajectory. We downsample the videos to 2.5 fps and extract person trajectories using the code released in [67]. Since we do not have the homographic matrix, we use the pixel values for the trajectory coordinates as it is done in [253].
Evaluation Metrics. Following prior work [3, 67, 191], we use two error metrics for person trajectory prediction:
i) *Average Displacement Error* (ADE): The average Euclidean distance between the ground truth coordinates and the prediction coordinates over all time instants,

$$\text{ADE} = \frac{\sum_{i=1}^{N}\sum_{t=1}^{T_{pred}} \|\tilde{Y}_t^i - Y_t^i\|_2}{N * T_{pred}} \tag{7.8}$$

ii) *Final Displacement Error* (FDE): The euclidean distance between the predicted points and the ground truth point at the final prediction time instant T_{pred},

$$\text{FDE} = \frac{\sum_{i=1}^{N} \|\tilde{Y}_{T_{pred}}^i - Y_{T_{pred}}^i\|_2}{N} \tag{7.9}$$

The errors are measured in the pixel space on ActEV/VIRAT whereas in meters on ETH and UCY. For future activity prediction, we use mean average precision (mAP).

	Method	ADE	FDE	move_ADE	move_FDE
Single Model	Linear	32.19	60.92	42.82	80.18
	LSTM	23.98	44.97	30.55	56.25
	Social LSTM	23.10	44.27	28.59	53.75
	SGAN-PV	30.51	60.90	37.65	73.01
	SGAN-V	30.48	62.17	35.41	68.77
	Ours	**17.99**	**37.24**	**20.34**	**42.54**
	Ours-Noisy	34.32	57.04	40.33	66.73
20 Outputs	SGAN-PV-20	23.11	41.81	29.80	53.04
	SGAN-V-20	21.16	38.05	26.97	47.57
	Ours-20	**16.00**	**32.99**	**17.97**	**37.28**

Table 7.1: Comparison to baseline methods on the ActEV/VIRAT validation set. Top uses the single model output. Bottom uses 20 outputs. Numbers denote errors thus lower are better.

Baseline methods. We compare our method with the two simple baselines and two recent methods: ***Linear*** is a single layer model that predicts the next coordinates using a linear regressor based on the previous input point. ***LSTM*** is a simple LSTM encoder-decoder model with coordinates input only. ***Social LSTM*** [3]: We train the social LSTM model to directly predict trajectory coordinates instead of Gaussian parameters. ***SGAN*** [67]: We train two model variants (PV & V) detailed in this work using the released code from Social-GAN [67] (https://github.com/agrimgupta92/sgan/).

Aside from using a single model at test time, Gupta et al. [67] also used 20 model outputs per frame and selected the best prediction to count towards the final performance. Following the practice, we train 20 identical models using random initializations and report the same evaluation results, which are marked "20 outputs" in Table 7.1.

Implementation Details. We use LSTM cell for both the encoder and decoder. The embedding size d_e is set to 128, and the hidden sizes d of encoder and decoder are both 256. Ground truth bounding boxes of persons and objects are used during the observation period (from time 1 to T_{obs}). For person keypoint features, we utilize the pre-trained pose estimator from [54] to extract 17 joints for each ground truth person box. For person appearance feature, we utilize the pre-trained object detection model FPN [137] to extract appearance features from person bounding boxes. The scene semantic segmentation features are resized to (64, 36) and the scene convolution layers are set to have a kernel size of 3, a stride of 2 and the channel dimension is

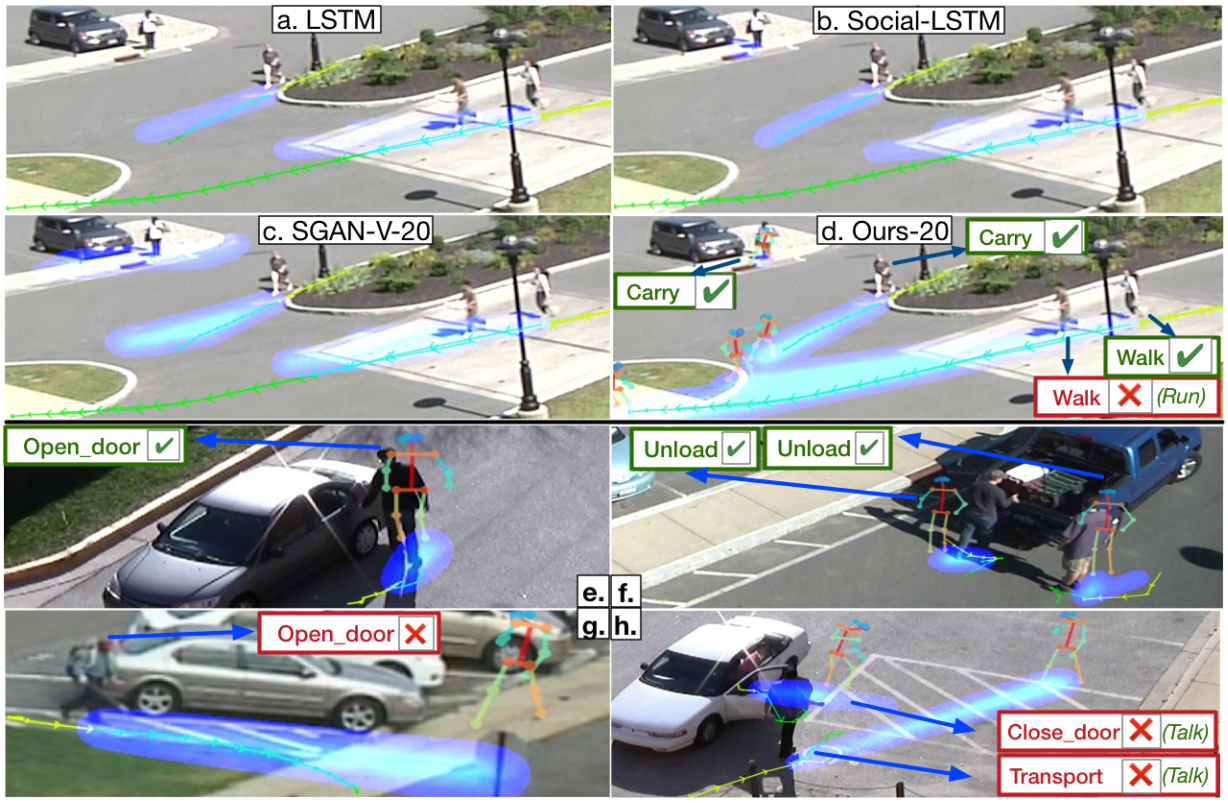

Figure 7.6: (Better viewed in color.) Qualitative comparison between our method and the baselines. Yellow path is the observable trajectory and green path is the ground truth trajectory during the prediction period. Predictions are shown as blue heatmaps. Our model also predicts the future activity, which is shown in the text and with the person pose template.

64. We resize all videos to 1920x1080 and utilize two grid scales, 32x18 and 16x9. The activation function is $tanh$ if not stated otherwise and we do not use any normalization. For training, we use Adadelta optimizer [259] with an initial learning rate of 0.1 and the dropout value is 0.3. We use gradient clipping of 10 and weight decay of 0.0001. For Social LSTM, the neighbor is set to 256 pixels as in [253]. All baselines use the same embedding size and hidden size as our model, therefore all encoder-decoder models have about the same numbers of parameters. Other hyper-parameters we use for the baselines follow the ones in [67].

Main Results. Table 7.1 lists the testing error, where the top part is the error of a single model output and the bottom shows the best result of 20 model outputs. The "ADE" and "FDE" columns summarize the error over all trajectories, and the last two columns further detail the subset trajectories of moving activities ("walk", "run", and "ride_bike"). We report the mean performance of 20 runs of our single model at Row 7. The standard deviation on "ADE" metric is 0.043. Full

Method	ADE ↓	FDE ↓	Act mAP ↑
Our full model	17.91	37.11	0.192
No p-behavior	18.99	39.82	0.139
No p-interaction	18.83	39.35	0.163
No focal attention	19.93	42.08	0.144
No act label loss	19.48	41.45	-
No act location loss	19.07	39.91	0.152
No multi-task	20.37	42.79	-

Table 7.2: Multi-task performance & ablation experiments.

numbers can be found in supplemental material. As we see, our method performs favorably against other methods, especially in predicting the trajectories of moving activities. For example, our model outperforms Social-LSTM and Social-GAN by a large margin of 10 points in terms of the "move_FDE" metric. The results demonstrate the efficacy of the proposed model and its state-of-the-art performance on future trajectory prediction. Additionally, as a step towards real-world application, we train our model with noisy outputs from object detection and tracking during the observation period. For evaluation, following common practise in tracking [241], for each trajectory, we assume the person bounding box location at time 1 is close to the ground truth location, and we evaluate the model prediction using tracking inputs and other visual features from time 1 to T_{obs} as shown in Table 7.1 "Ours-Noisy".

Qualitative analysis. We visualize and compare our model outputs and the baselines in Fig. 7.6. As we see, our method outputs more accurate trajectories for each person, especially for the two persons on the right that were about to accelerate their movement. Our method is also able to predict most of the activities correct except one (walk versus run). Our model successfully predicts the activity "carry" and the static trajectory of the person near the car, while in Fig 7.6(c), SGAN predicts several moving trajectories in different directions. We further provide a qualitative analysis of our model predictions. (i) Successful cases: In Fig 7.6(e) and 7.6(f), both the trajectory prediction and future activity prediction are correct. (ii) Imperfect case: In Fig 7.6(g), although the trajectory prediction is mostly correct, our model predicts that the person is going to open the door of the car, given the observation that he is walking towards the side of the car. (iii) Failed case: In Fig 7.6(h), our model fails to capture the subtle interactions between the two persons and predicts that they will go separate ways, while in fact they are going to stop and talk to each other.

Method	ADE ↓	FDE ↓	Act mAP ↑
Our full model	17.91	37.11	0.192
MobileNet-ADE20K	18.37	38.41	0.185
Xception-CityScapes	18.54	38.37	0.173
MobileNet-CityScapes	18.77	38.99	0.178
No p-interaction	18.83	39.35	0.163

Table 7.3: Scene semantic segmentation ablation experiments.

7.3.2 Ablation Model

In Table 7.2, we systematically evaluate our method through a series of ablation experiments, where "ADE" and "FDE" denotes the errors thus lower are better. "Act" is the mean Average Precision (mAP) of the activity label prediction over 29 activities and higher are better.

Efficacy of rich visual features. We investigate the feature contribution of person behavior and person interactions. As shown in the first three rows in Table 7.2, both features are important to trajectory prediction while person behavior features are more essential for activity prediction. Individual feature ablations are in the supplementary material.

Effect of focal attention. In the fourth row of Table 7.2, we replace focal attention in Eq. (7.5) with a simple average of the last hidden states from all encoders. Both trajectory and activity prediction hurt as a result.

Impact of multi-task learning. In the last three rows of Table 7.2, we remove the additional tasks of predicting the activity label or the activity location or both to see the impact of multi-task learning. Results show the benefit of our multi-task learning method.

Effect of scene semantic segmentation models. Since we use a pre-trained scene semantic segmentation model to extract features for our model, we want to see how different level of accuracies for the segmentation model would affect our prediction performance. In Table 7.3, we compare the segmentation model that we use in the full model (Xception trained on ADE20K) with MobileNet, which is a light-weight segmentation model that is about 10% worse on the scene segmentation benchmark [32]. We also compare models trained on a different dataset (ADE20K vs. CityScapes). As we see, a stronger scene segmentation model leads to a slightly better performance and ADE20K models are better in general.

	Method	ETH	HOTEL	UNIV *	ZARA1	ZARA2	AVG
Single Model	Linear	1.33 / 2.94	0.39 / 0.72	0.82 / 1.59	0.62 / 1.21	0.77 / 1.48	0.79 / 1.59
	LSTM	1.09 / 2.41	0.86 / 1.91	**0.61** / **1.31**	**0.41** / **0.88**	0.52 / 1.11	0.70 / 1.52
	Alahi et al. [3]	1.09 / 2.35	0.79 / 1.76	0.67 / 1.40	0.47 / 1.00	0.56 / 1.17	0.72 / 1.54
	Ours-single	**0.88** / **1.98**	**0.36** / **0.74**	0.62 / 1.32	0.42 / 0.90	**0.34** / **0.75**	**0.52** / **1.14**
20 Outputs	[67](V)	0.81 / 1.52	0.72 / 1.61	0.60 / 1.26	0.34 / 0.69	0.42 / 0.84	0.58 / 1.18
	[67](PV)	0.87 / 1.62	0.67 / 1.37	0.76 / 1.52	0.35 / 0.68	0.42 / 0.84	0.61 / 1.21
	[191]	**0.70** / **1.43**	0.76 / 1.67	**0.54** / **1.24**	**0.30** / **0.63**	0.38 / 0.78	0.54 / 1.15
	Ours-20	0.73 / 1.65	**0.30** / **0.59**	0.60 / 1.27	0.38 / 0.81	**0.31** / **0.68**	**0.46** / **1.00**

Table 7.4: Comparison of different methods on ETH (Column 3 and 4) and UCY datasets (Column 5-7). * We use a smaller test set on UNIV since 1 video is unable to download.

7.3.3 ETH & UCY

Dataset. ETH [162] and UCY [118] are common datasets for person trajectory prediction benchmark [3, 67, 149, 191]. Same as previous works, we report performance by averaging over both datasets. We use the same data processing method and settings detailed in [67]. This benchmark includes videos from five scenes: ETH, HOTEL, UNIV, ZARA1 and ZARA2. Leave-one-scene-out data split is used and we evaluate our model on 5 sets of data. We follow the same testing scenario and baselines as in the previous section. We have also cited the latest state-of-the-art results from [191]. Due to 1 video cannot be downloaded, we use a smaller test set for UNIV and a smaller training set across all splits. The other 4 test sub-datasets are the same as in [67] so the numbers are comparable.

Since there is no activity annotation, we do not use activity label prediction module in our model. Since the annotation is only a point for each person and the human scale in each video doesn't change much, we apply a fixed size expansion from points for each video to get the person bounding box annotation for feature pooling. We do not use any other bounding box. We don't use any additional annotation compared to baselines to ensure a fair comparison.

Implementation Details. We do not use person keypoint feature. Final location loss and trajectory L2 loss are used. Unlike [191], we don't utilize any data augmentation. We train our model for 40 epochs with the adadelta optimizer. Other hyper-parameters are the same as in Section 7.3.1.

Results & Analysis. Experiments are shown in Table 7.4. Our model outperforms other methods in both evaluations, where we obtain the best-published single model on ETH and best

average performance on the ETH & UCY benchmark. As shown in the table, our model performs much better on HOTEL and ZARA2. The average movement at each time-instant in these two scenes are 0.18 and 0.22, respectively, much lower than others: 0.389 (ZARA1), 0.460 (ETH), 0.258 (UNIV). Recall that the leave-one-scene-out data split is used in training. The results suggest other methods are more likely to overfit to the trajectories of large movements, e.g. Social-GAN [67] often "over-shoot" when predicting the future trajectories. In comparison, our method uses attention to find the "right" visual signal and show better performance for trajectories of small movements on HOTEL and ZARA2 while still being competitive for trajectories of large movements.

7.4 Related Work

This work falls under the category of sequential models that utilize both static and dynamic environmental cues in the human motion prediction literature [186]. We utilize multiple contextual visual cues from both the target agent and the environment. We are the first work to jointly optimize activity and trajectory prediction with a unified framework. In the following, we review a few relevant recent approaches based on their use of these contextual cues. Then we also review some activity prediction works.

Person-person models for trajectory prediction. Person trajectory prediction models try to predict the future path of people, mostly pedestrians. A large body of work learns to predict person path by considering human social interactions and behaviors in crowded scene [250, 256]. Zou et al. in [272] learned human behaviors in crowds by imitating a decision-making process. Social-LSTM [3] added social pooling to model nearby pedestrian trajectory patterns. Social-GAN [67] added adversarial training on Social-LSTM to improve performance. Different from these previous work, we represent a person by rich visual features instead of simply considering a person as points in the scene. Meanwhile we use *geometric relation* to explicitly model the person-person relations in the scene, which has not been used in previous work.

Person-scene models for trajectory prediction. A number of works focused on learning the effects of the physical scene, e.g., people tend to walk on the sidewalk instead of grass. Kitani et al. in [103] used Inverse Reinforcement Learning to forecast human trajectory. Xie et al. in [245] considered pedestrian as “particles” whose motion dynamics are modeled within the framework of Lagrangian Mechanics. Scene-LSTM [149] divided the static scene into Manhattan Grid and predict pedestrian’s location using LSTM. CAR-Net [87] proposed an attention network on top of scene semantic CNN to predict person trajectory. SoPhie [191] combined

deep neural network features from scene semantic segmentation model and generative adversarial network (GAN) using attention to model person trajectory. A disparity to [191] is that we explicitly pool scene semantic features around each person at each time instant so that the model can directly learn from such interactions.

Person visual features for trajectory prediction. Some recent works have attempted to predict person path by utilizing individual's visual features instead of considering them as points in the scene. Kooij et al. in [105] looked at pedestrian's faces to model their awareness to predict whether they will cross the road using a Dynamic Bayesian Network in dash-cam videos. Yagi et al. in [253] used person keypoint features with a convolutional neural network to predict future path in first-person videos. Different from these works, we consider rich visual semantics for future prediction that includes both the person behavior and their interactions with soundings .

Activity prediction/early recognition & Tracking. Many works have been proposed to anticipate future human actions using Recurrent Neural Network (RNN). [142] and [4] proposed different losses to encourage LSTM to recognize actions early in internet videos. Srivastava et al. in [206] utilized unsupervised learning with LSTM to reconstruct and predict video representations. Another line of works is anticipating human activities in robotic vision [85, 106]. There are previous works that take into account multiple cues in videos for tracking [86, 190] and group activity recognition [39, 196, 197]. Our work differs in that rich visual features and focal attention are used for joint person path and activity prediction. Meanwhile, our work utilizes novel activity location prediction to bridge the two tasks.

7.5 Summary

In this chapter, we have presented a new neural network model for predicting human trajectory and future activity simultaneously. We first encode a person through rich visual features capturing human behaviors and interactions with their surroundings. Then we add an auxiliary task of predicting the activity locations to facilitate the joint training process. We refer to the resulting model as *Next*. We showed the efficacy of our model on both popular and recent large-scale video benchmarks on person trajectory prediction. In addition, we quantitatively and qualitatively demonstrated that our *Next* model successfully predicts meaningful future activities.

However, since short-term future prediction is not enough to ensure safe operations in autonomous driving applications, in the following chapters, we introduce a new, human-annotated

long-term trajectory and action prediction benchmark and new models to tackle this challenge.

Chapter 8

A Multi-view Long-term Trajectory Prediction Benchmark

In the following chapters, we explore the problem of long-term trajectory prediction, which aims to expand the prediction horizon two-fold to three-fold compared to most previous works so that the models can provide better traffic safety. In this chapter, we first describe our new human-annotated long-term trajectory prediction dataset with multi-view camera data in urban traffic scenes.

8.1 Motivation

Figure 8.1: Overview of the MEVA dataset. Image taken from [40].

Most previous works [67, 131] on pedestrian trajectory prediction only focus on predicting the near future (within 5 seconds). In applications like autonomous systems, safe operation and collision avoidance are crucial for the systems to be co-mingling with humans. This requires predicting human motion beyond a short time-horizon. While near-future prediction can be informed by recent observed trajectory history, long-term prediction depends on inferring human intents or goals, which requires the model to make use of scene constraints, social interactions, person-vehicle interactions and even common sense reasoning.

Long-term prediction has the following challenges. First, it is highly uncertain compared to short-term forecasting. Using only the trajectory history is not enough to achieve accurate long-term prediction, and semantic reasoning with scene constraints and agent behaviors is crucial, especially in traffic scenes. Second, the target person might be traveling out-of-frame and hence require the models to infer the final intended destination without actually seeing it. Third, long-term prediction may require more observation inputs that result in greater computation demand while it is critical to have an efficient system that could output prediction in a timely fashion.

To tackle the aforementioned challenges, we need to construct a new dataset as existing datasets like ETH/UCY [118, 162], Stanford Drone [184], only have short-term trajectories and no rich behavioral annotations about the agents like activities. Another important aspect of these previous datasets is that they lack person-vehicle interactions, which is important for trajectory prediction in urban traffic scenes. We turn to the recently release the Multiview Extended Video with Activities (MEVA) dataset [40], for its characteristics that could help us solve the problems mentioned before. There are three main reason to use the MEVA dataset. First, the MEVA dataset contains long-term tracks that last up to a few minutes that are captured by at lease one camera. See Figure 8.1 on the right, where two women with jackets traveled from the cafe in a building to the bus station two hundred meters away. Second, the MEVA dataset contains human-annotated activities in a urban traffic setting, which includes person-only activities and person-vehicle activities. It could be utilized for enhanced modeling of human behaviors for trajectory prediction in traffic scenes. Last but not least, as shown in Figure 8.1, there are multiple overlapping camera views, which could help us study how to solve the second challenge where the pedestrians traveled out-of-frame to other cameras. Since the MEVA dataset does not contain object tracking annotations (only individual activity instance annotations are provided), in the next section, we describe how we collect and annotate the MEVA trajectory prediction dataset (MEVA-Trajectory) in the next section.

Existing datasets. There are several popular pedestrian trajectory prediction benchmarks,

	Classes
Object	Bike, Person, Bag, Vehicle
Activity	person_opens_vehicle_door, person_enters_vehicle, person_talks_to_person, person_opens_facility_door, person_closes_vehicle_door, person_texts_on_phone, person_enters_scene_through_structure,person_opens_trunk, person_unloads_vehicle,person_closes_trunk, person_picks_up_object,person_loads_vehicle, person_embraces_person

Table 8.1: MEVA Object & Activity Classes.

such as SDD [184], ETH/UCY [118, 162], KITTI [59] and nuScenes [18], all of which are for short-term (1-5 seconds in the future) trajectory prediction. These datasets do not include human action/activity annotations. The VIRAT/ActEV [158] dataset has longer tracks with activity annotations but it does not have the multi-view property as discussed before. There are other datasets from the object tracking community like Duke MTMC [183], which is also multi-view, but it is no longer available due to privacy concerns and there is no activity annotation.

Statistics of the MEVA dataset. The MEVA dataset is a large multi-view activity dataset with 28 cameras of 1920x1080 resolution, recorded on a large military training facility with scripted and unscripted actions. All video data are formatted into 5-minutes video clips. The type of activities include in our data collection (pedestrian-related) are listed in Table 8.1. We download the evaluation split with 266 videos and the training split (drop-09) with 1277 videos from the MEVA data repository [1].

8.2 The MEVA-Trajectory Dataset

8.2.1 Data Collection

As mentioned before, we need to get object tracks from the MEVA dataset to form trajectories for our task. The MEVA-Trajectory dataset is released at https://github.com/JunweiLiang/MEVA-Trajectory. Code to reproduce the results is released at https:

[1] https://gitlab.kitware.com/meva/meva-data-repo/-/tree/master/annotation/DIVA-phase-2/MEVA

//github.com/JunweiLiang/Object_Detection_Tracking. The data collection process includes the following steps.

Single video object detection and tracking. First, we use our efficient and accurate object detection and tracking system proposed in chapter 2 to get individual object tracks, including persons and vehicles. In addition, in order to have more accurate long-term tracks, we utilize re-identification techniques to refine the person and vehicle tracklets. Specifically, we compare each detected tracklets that could potentially be considered the same (based on spatial and temporal constraints) using re-identification models, and the tracklets are refined if the distance score is below a threshold. For person re-identification, we utilize the OSNet [270] model trained on Market1501 dataset [268]. For vehicle re-identification, we utilize the winning entry [170] of vehicle tracking competition from the AI City Challenge 2020 [156]. In Figure 8.2, we show an example of before and after using the person re-identification model. As we see, this technique is able to correct the ID switch in the original results. Empirically, we find that the re-identification model can reduce about 10% ID switch errors from the original tracking results.

Figure 8.2: Single video object detection and tracking with re-identification. On the left is the original tracking results and on the right is after re-identification. See text for details.

Cross-view tracklet association. After we have object tracklets from individual videos, we need to associate the same object across multiple videos. Therefore we compute a homography matrix between each camera to the ground plane as a spatial constraint for the tracklet association process. The MEVA dataset provides camera models (intrinsic and extrinsic parameters) and 3D point clouds of the whole facility. We utilize the 3D point cloud and get a top-down-view camera model as shown in Figure 8.7. We then compute the homography matrix between each original camera and the top-down-view camera by:

$$H_{ab} = R_a R_b^T - \frac{(-R_a * R_b^T * t_b + t_a) n^T}{d} \tag{8.1}$$

where R_a, R_b and t_a, t_b are the rotation and translations of the two cameras. n is the normal

vector of the plane and d is the distance of the camera b to the plane. To check the accuracy of the homography transformation, we use the computed homography matrix and warp the original camera view to the top-down view, as shown in Figure 8.3 and Figure 8.6. The colored bounding boxes are for reference. The red circles on the top-down-view image, which are used for range reference, have a radius of 50 and 100, respectively, and they use the center of the red bounding box as their centers. As we can see, the homography matrices have reasonable accuracy. To get the final tracklets across different cameras, we first transform the center of each bounding box of the tracklets to the ground-plane and then compare different tracklet's (within a certain spatial and temporal tolerance) similarity using the aforementioned re-identification models. The tracklet association is done with the Hungarian algorithm.

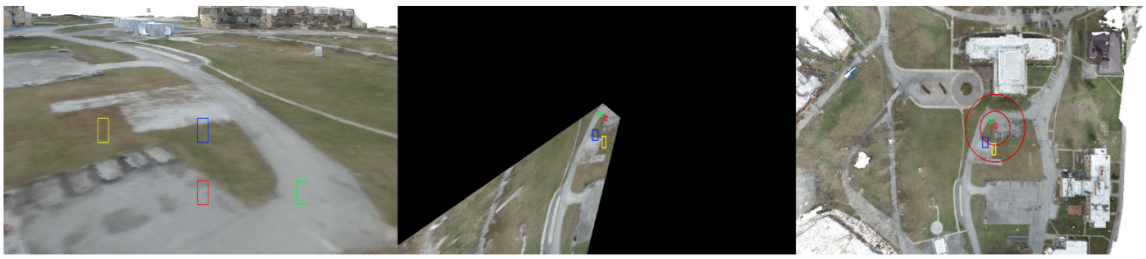

Figure 8.3: Visualization of homography transformation between original camera and top-down-view camera.

Manual annotation of global tracklets. After we get the global tracklets, we conduct a manual annotation process to reject global tracks that are not correct. In Figure 8.4, we show an example of the accepted global tracks and an rejected one. The numbers on top of the red bounding boxes are global track ID as well as local track ID. The original MEVA activity annotations are also visualized in the figure. They are assigned to the tracks based on the bounding box overlap between the tracklets and the original activity annotations.

Trajectory smoothing. Finally, since the bounding boxes of the tracklets might be unstable due to occlusions and miss-detections, we conduct trajectory smoothing to get a global trajectory for each global track that is suitable for the trajectory prediction task. For each global tracklet, multiple local trajectories (computed using the bottom center of the bounding boxes) from each camera are smoothed using moving average with a window size of 200 frames (the fps is 30). We show two examples of the smoothing results in Figure 8.5. The red trajectories are the final trajectory after smoothing and the rest are local trajectories. The activity annotations are also shown in the figure. As we see, the left example shows a reasonable trajectory of a

Figure 8.4: Visualization of accepted and rejected global tracks during the human annotation process. See text for details.

person walking from the parking lot to the building along the path.

Figure 8.5: Overview of the MEVA dataset.

8.2.2 Data Characteristics

Our MEVA-Trajectory dataset is the first publicly available dataset composed of multi-view long-term pedestrian trajectories with rich activity annotations. Compared to previous common pedstrian trajectory benchmarks like ETH, UCY, Stanford Drone and KITTI, our new dataset contains a few characteristics. With the focus of long-term trajectories and enhanced vi-

Datasets	ETH,UCY [118, 162]	SDD [184]	KITTI [59]	ActEV [158]	Ours
HD Resolution	-	-	✓	partial	✓
Multi-View	-	-	-	-	✓
Extended Length	-	-	✓	✓	✓
Event/Goal-Driven	-	-	-	partial	✓
Traffic Scene	-	partial	✓	✓	✓
Activity Annotation	-	-	-	✓	✓

Table 8.2: Comparison of representative trajectory prediction datasets and our new dataset called MEVA-Trajectory. The pedestrians in our dataset are often associated with a clear intent, whether it is going to the bus station to get on a bus or entering a building for a class, thanks to the scripted events from the original MEVA dataset.See text for details.

sual understanding, our dataset consists of extended length high resolution videos. The pedestrians in our dataset are often associated with a clear intent, whether it is going to the bus station to get on a bus or entering a building for a class, thanks to the scripted events from the original MEVA dataset. Compared to the ActEV [158] dataset used in previous chapter 7, MEVA-Trajectory has the advantage of longer tracks with synchronized multi-view data. Full data characteristics comparison is summarized in Table 8.2.

In addition, we compare trajectory statistics between our dataset with the ETH/UCY and the ActEV dataset in Table 8.3. As we see, our dataset has more number of trajectories. Notably, in terms of median trajectory length, our MEVA-Trajectory dataset is 48.3 seconds, which is more than five times longer compared to the 8.8 seconds of the ETH/UCY dataset. This shows that our dataset is suitable for the long-term trajectory prediction task.

Quality of the Dataset. Here we briefly discuss the quality of our collected dataset. For single video object detection and tracking, we utilize a well-tuned system for surveillance-type videos that achieves 0.83 average precision for persons and 0.98 for vehicles on VIRAT dataset [158]. See Table 2.2 in chapter 2. We then utilize state-of-the-art vehicle and person re-identification model for cross-view tracklet association. After the manual annotation effort, we have reduced the automatic global person tracklets from 2549 to 864, with a rejection rate of 66%. In this way, we have ensured the precision of the global person tracklets so that they could be reliably utilized for long-term trajectory prediction. Meanwhile, with high-quality single-video object tracks, we could extract object features for each global track from the videos accurately.

	ETH, UCY	ActEV	Ours
#Cameras	4	5	10
Resolutions	640x480,720x576	1920x1080, 1280x720	1920x1080
FPS	25	30	30
Annotation FPS	2.5	30	30
Total Traj. Length	4:59:05	12:14:44	15:36:17
#Traj.	1535	1073	2060 / 864*
Median Traj. Length	8.8	28.8	48.3
Median #Camera	1	1	2
Annotations	Person coordinates	Person+object bounding boxes,activities	Person+object bounding boxes,activities

Table 8.3: Statistical comparison to commonly used person trajectory benchmark datasets. "*" means the number of global tracks as opposed to local tracks in a video. The unit of the trajectory length is seconds. See text for details.

8.3 Summary

In this chapter, we describe our new dataset, called MEVA-Trajectory, for long-term trajectory prediction in urban traffic scenes. We show in details our process of collecting and annotating the dataset. In the next chapter, we discuss the experiments we have conducted on this dataset.

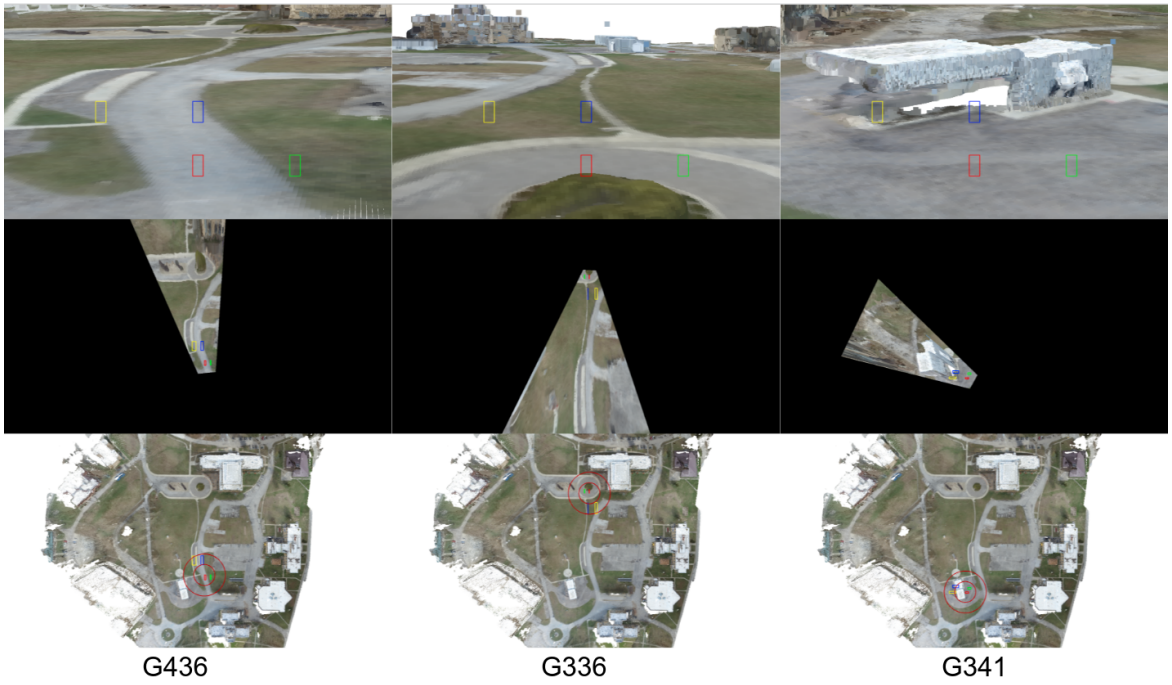

Figure 8.6: Overview of the MEVA dataset.

Figure 8.7: Visualization of top-down-view camera.

Figure 8.8: MEVA-Trajectory dataset's unit is in feet. On the left is an screenshot of the satellite view of the dataset location, Muscatatuck Urban Training Center, and on the right is the top-down view of the 3D point cloud.

Chapter 9

Long-term Trajectory Prediction with Scene and Action Understanding

In previous chapters, we have laid the foundation of standalone action recognition models and future prediction models using scene semantics. In this chapter, we propose a new model, called Next-GAT, which utilizes graph attention with scene and action understanding module, for long-term trajectory prediction. We conduct comprehensive experiments on two datasets, ActEV and our proposed MEVA-Trajectory, and show the efficacy of our Next-GAT model in urban traffic scenes.

9.1 Motivation

As discussed in chapter 8, most previous works [67, 82, 131, 155] on pedestrian trajectory prediction only focus on predicting trajectories of the short-term future (within 5 seconds/12 timesteps). To better improve safety in applications like self-driving cars and socially-aware robots, long-term predictions are needed. In this work, we aim to increase the prediction horizon two folds compared to previous work to predicting 12 seconds (30 timesteps) into the future. The term long-term is consistent with recent published works [99, 225].

Long-term prediction is challenging compared to short-term forecasting. In applications like self-driving cars, long-term prediction can better ensure safety. Using only the trajectory history is not enough to achieve accurate long-term prediction, and semantic reasoning with scene constraints and agent behaviors is crucial. To tackle the this challenge, we build on top of the models proposed in previous chapters and propose the Next-Graph-Attention-Network (Next-GAT), which takes both scene semantic understanding and behavioral analysis

into account for trajectory prediction. We add graph attention network (GAT), inspired by a recent successful work that used GAT for trajectory prediction [82], to model agent interactions. In terms of evaluation benchmarks, we choose both the ActEV dataset [158] and our MEVA-Trajectory dataset (chapter 8) for a comprehensive analysis in urban traffic scenarios.

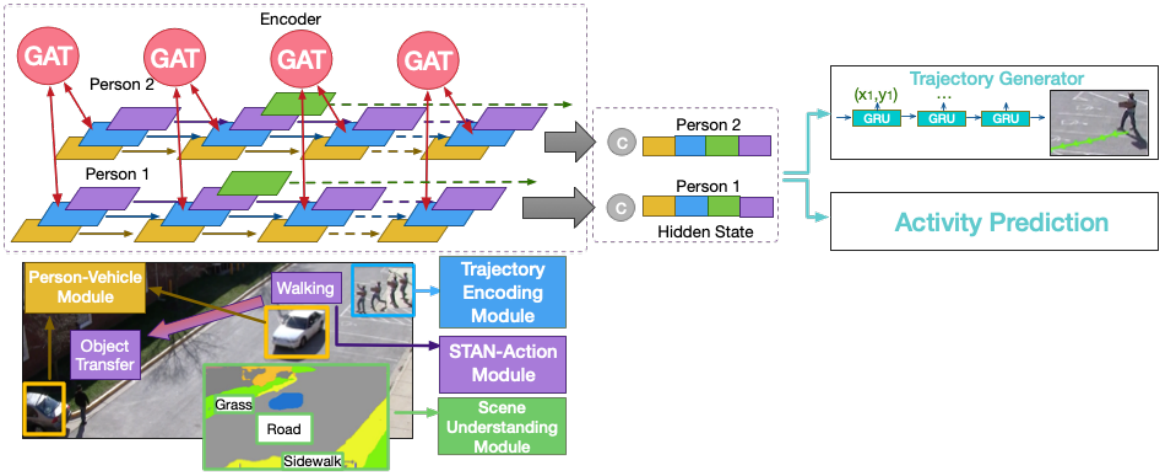

Figure 9.1: Overview of the Next-GAT model.

9.2 The Next-GAT Model

Similar to the Next model proposed in chapter 7, we utilize multi-task learning with trajectory prediction and activity prediction. Following the same problem formulation proposed in chapter 7, we simplify the Next model's person behavior module by removing the pose analysis part and adding the STAN module proposed in chapter 4, to allow the model encoder to learn viewpoint invariant representations that could help both activity and trajectory prediction. We also simplify the person interaction module, by only using one scene segmentation frames as the static scene semantics do not change a lot during the observation period. Finally, we add graph attention layer to the trajectory encoder to model the interactions between different agents in the scene.

9.2.1 Network Architecture

Fig. 9.1 shows the overall network architecture of our Next-GAT model. Our model focuses on modeling of both the scene semantics and the agent's behavior, by using viewpoint-invariant action representations and graph attention. Next-GAT has the following key components:

Trajectory encoding module takes the relative coordinates of the pedestrian's trajectories during the observation period and applies graph attention modeling [231] at each timestep between all pedestrians in the scene.
STAN-Action module takes the frame sequence of the target agent to model the behavioral cues for better action prediction and trajectory prediction.
Scene understanding module takes the scene semantic segmentation feature of the first video frame of the whole scene to encode the static scene information for both action and trajectory prediction.
Person-vehicle interaction module takes the person-vehicle geometric features from the frame sequence to encode the important interactions between pedestrians and vehicles in traffic scenes.
Trajectory generator uses the encoded visual features and predicts the future trajectory by a RNN decoder.
Activity prediction utilizes the concatenated encoder features to predict the future activity label. In the rest of this section, we will introduce the above modules and the learning objective in details.

9.2.2 Trajectory Encoding Module

Given the observation trajectory of each pedestrian, we first encode the location coordinates by

$$e_t = RNN_1(MLP[x_t, y_t]) \in \mathbb{R}^d, \tag{9.1}$$

where $[x_t, y_t]$ is the trajectory prediction of time $t-1$, and MLP is a single fully-connected layer to embed the coordinates into higher dimensions. We therefore get a individual motion-encoded hidden state tensor of $E \in \mathbb{R}^{T_{obs} \times d}$, where d is the hidden size of the RNN. We use GRU in our implementation. Then at each timestep, we utilize graph attention network [231] and another recurrent network, to refine each pedestrian's hidden states with all other pedestrians in the scene. It is computed by

$$H_t = GAT(RNN_2(E_t, H_{t-1})) \in \mathbb{R}^{K \times d}, \tag{9.2}$$

where K is the number of pedestrians in the scene. Let h_i be the corresponding output of GAT for the i-th person at time t. It is computed using GAT by

$$h_i = \sum_{j \in \mathcal{K}} f_e([v_i, v_j]) v_j \tag{9.3}$$

where $\mathcal{K}$ represents all pedestrian in the scene and node v_i is represented as $v_i = [\tilde{h_i}]$, the outputs of RNN_2. f_e is some edge function (implemented as an MLP in our experiments) that computes the attention/edge weights and then softmax.

9.2.3 STAN-Action Module

Given the frame sequence of each target agent $F \in \mathbb{R}^{H \times W \times T_{obs} \times c}$, we utilize a light-weighted version of the STAN model proposed in chapter 4 to extract a vector representation of the agent's actions. Specifically, the frame sequence is processed through a Resnet3D group and a STAN layer, before feeding into an MLP layer to get a fix-length vector representation. Please refer to chapter 4 for more details.

9.2.4 Scene Understanding Module

To encode the static scene semantic information, we first use a pre-trained scene segmentation model [32] to extract pixel-level scene semantic classes for the first video frame. We use totally $N_s = 10$ common scene classes, such as roads, sidewalks, etc., same as in chapter 7. The scene semantic features are integers (class indexes) of the size $h \times w$, where h, w are the spatial resolution. We first transform the integer tensor into N_s binary masks (one mask for each class). This results in N_s real-valued masks, each of the size of $h \times w$. We apply two convolutional layers on the mask feature with a stride of 2 and then an MLP to get a fix-length vector representation.

9.2.5 Person-Vehicle Interaction Module

We use the same person-object interaction module from chapter 7 and explicitly model the *geometric relation* of all the vehicles and pedestrians in the scene. At any time instant, given the observed box of a person (x_b, y_b, w_b, h_b) and K other vehicles in the scene $\{(x_k, y_k, w_k, h_k)|k \in [1, K]\}$, we encode the geometric relation into $\mathcal{G} \in \mathbb{R}^{K \times 4}$, the k-th row of which equals to:

$$\mathcal{G}_k = [\log(\frac{|x_b - x_k|}{w_b}), \log(\frac{|y_b - y_k|}{h_b}), \log(\frac{w_k}{w_b}), \log(\frac{h_k}{h_b})] \tag{9.4}$$

This encoding computes the geometric relation in terms of the geometric distance and the fraction box size. We use a logarithmic function to reflect our observation that human trajectories are more likely to be affected by close-by objects. This encoding has been proven effective in object detection [78]. We then embed the geometric features at the current time into d_e-dimensional vectors and feed the embedded features into an GRU encoder to obtain the final feature in $\mathbb{R}^{T_{obs} \times d}$.

	Short-term Trajectory Prediction			Long-term Trajectory Prediction		
	Act	Single-Future	Multi-Future	Act	Single-Future	Multi-Future
NN	-	1.79/3.12	-	-	3.47/6.5	-
Const. Vel.	-	1.17/2.25	-	-	2.78/5.74	-
Human Perf.*	-	-	-	-	2.60/5.49	-
SGAN	-	1.21/2.25	0.88/1.63	-	3.37/6.66	2.69/5.29
STGAT	-	1.43/2.75	0.88/1.68	-	4.05/7.78	2.27/4.63
STGCNN	-	1.48/2.57	1.08/1.93	-	3.46/6.51	2.78/5.46
Next	0.192	1.06/2.03	0.87/1.79	0.211	2.22/4.56	1.97/4.05
Next-GAT	**0.236**	**0.84/1.57**	**0.76/1.42**	**0.267**	**1.94/4.05**	**1.63/3.36**

Table 9.1: Main results on the ActEV dataset. The numbers are ADE/FDE respectively for trajectories, and mAP for activity ("Act"). "*": estimated performance. See evaluation metrics for details.

9.2.6 Prediction Module

We concatenate the fix-length representation of scene semantics, person-vehicle interaction and agent action with the last timestep's output of the trajectory encoding module, and feed into our multi-task learning framework, which includes the trajectory generator and activity prediction module. Similar to chapter 7, the multi-task training objective include a cross-entropy loss the activity prediction and L2 loss for trajectory prediction.

9.3 Experiments

Dataset & Setups. We compare our model and baselines on both short-term and long-term trajectory prediction for a comprehensive analysis of model performance. We evaluate the proposed Next-GAT model on two benchmarks, ActEV and MEVA-Trajectory, since both datasets satisfy the required track length for long-term trajectory prediction (see Table 8.3 in chapter 8). For short-term trajectory prediction task, we follow the common protocol [3, 67, 82, 155, 191] of observing 8 timesteps (3.2 seconds) and predicting 12 timesteps (4.8 seconds). For long-term trajectory prediction, we set the criteria to observe 12 timesteps (4.8 seconds) and predict 30 timesteps (12 seconds), which is slightly further into the future compared to a recent long-term trajectory prediction work [225].

Evaluation Metrics. Different from previous works [3, 67, 82, 155, 191] that only evaluate on the multi-modality of the results, we are also interested in how the model performs given a single best prediction, so that in real-world application, users can better optimize their system according to the computation-accuracy trade-off. We evaluate all models with both single-future metric ($minADE_1/minFDE_1$) and multi-future metric ($minADE_{20}/minFDE_{20}$). For detailed formulations of the metrics, please refer to Section 5.4 in chapter 5. In terms of activity prediction, we use Mean Average Precision (mAP), same as in the previous chapter.

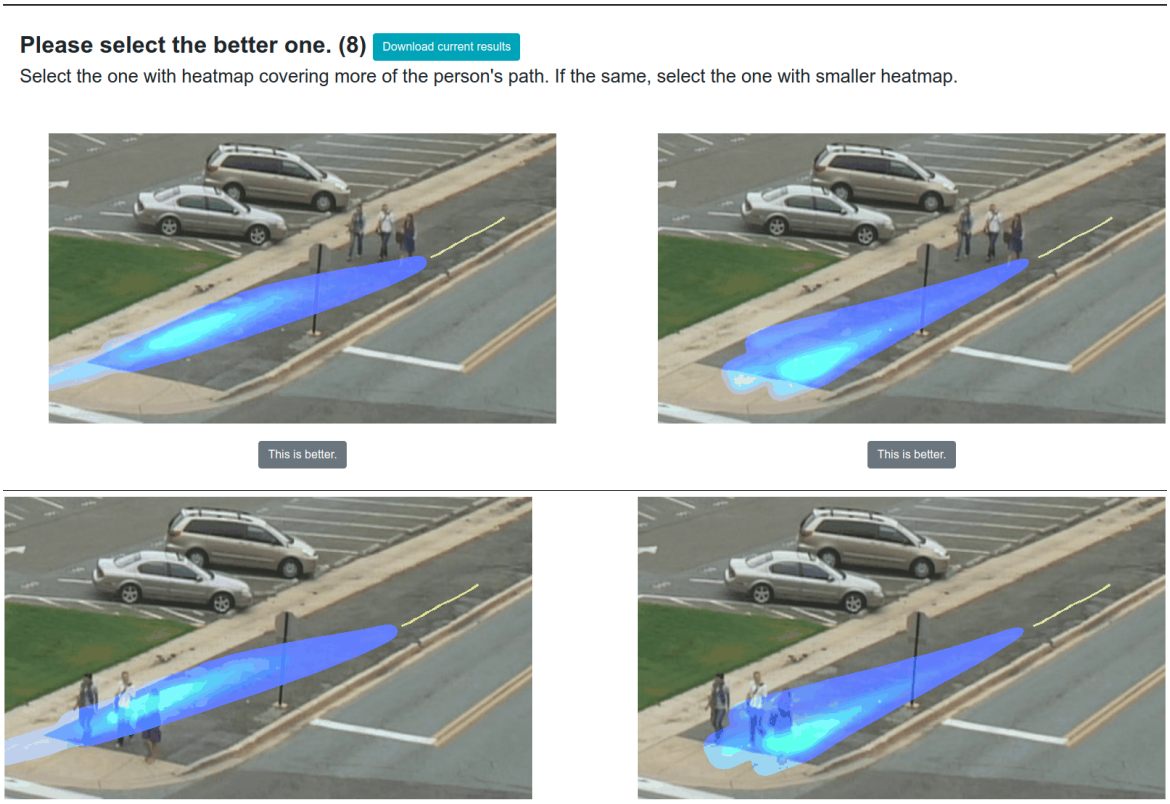

Figure 9.2: On the top is the user study web interface. The video frames are shown in the format of GIFs. The bottom image shows the last video frame. In this example, the women on the right is the target agent and from the two images we can see that the model of the right image is better since the predicted heatmap covers all of the woman's future path.

Baseline Methods. We compare the proposed Next-GAT model with recent representative, open-sourced baselines as well as classic baselines, which includes: (1) Nearest Neighbor (NN) is a simple and intuitive baseline in which we stitch the most similar trajectories observed in the training set during inference time. (2) Constant Velocity [193] (Const. Vel.) is another simple non-parametric baseline in which we use the velocity at the last observation timestep

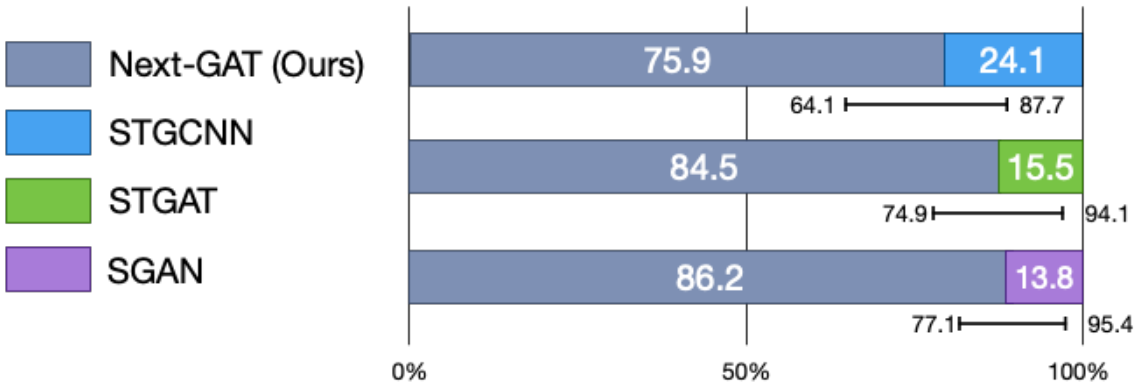

Figure 9.3: User study of trajectory prediction results. Both the mean and 95% confidence interval are shown.

to generate future trajectories. (3) The *SGAN* [67] model consists of a generator for trajectory prediction and a discriminator which outputs the predicted trajectory with real or fake label. (4) The *STGAT* [82] model uses graph attention network to model agent interactions. This is a popular method that uses recurrent neural nets. (5) The *STGCNN* [155] model substitutes the need of aggregation methods by modeling the interactions as a graph. This is a recent representative and popular method that use convolutional neural nets. *Next* model is described in chapter 7. *Next-GAT* is our proposed method in this chapter.

Implementation Details. For ActEV dataset, we transform the trajectory from pixels to the ground plane using the provided homography matrices. For MEVA-Trajectory dataset, the global trajectory in world coordinates is used. The visual features of the longest local tracks are used as inputs to our model. We use GRU cell for both the encoder and decoder. No teacher forcing is used during training of the decoder. The embedding size d_e is set to 128, and the hidden sizes d of encoder and decoder are both 256. The scene semantic segmentation features are resized to (64, 36) and the scene convolution layers are set to have a kernel size of 3, a stride of 2 and the channel dimension is 64. The activation function is $tanh$ if not stated otherwise and we do not use any normalization. For training, we use Adam optimizer [259] with an initial learning rate of 0.0001 and the dropout value is 0.3. For multi-future inference, we train 20 identical models with different random initialization. We also tune the hyper-parameters of other baselines if the reported ones do not perform well in their papers.

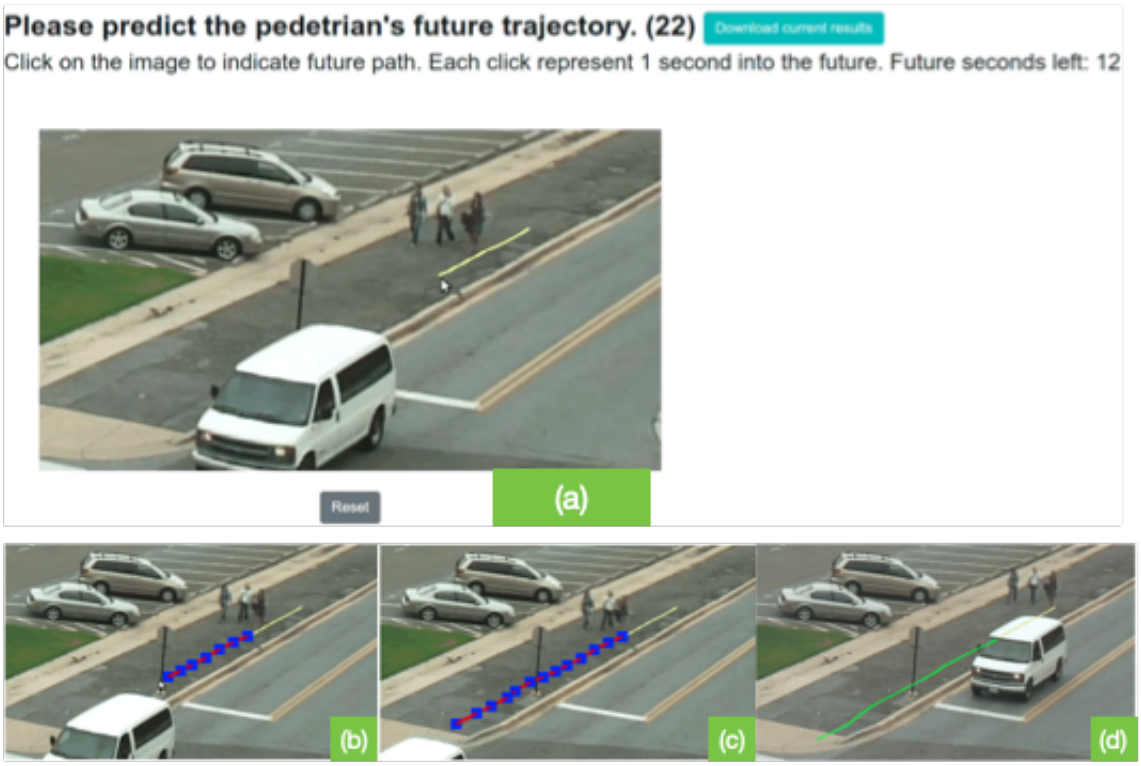

Figure 9.4: Human performance experiment. See text for details

9.3.1 ActEV

Main Results. The quantitative results are shown in Table 9.1, where both short-term trajectory prediction and long-term trajectory prediction results are shown. The numbers are ADE/FDE respectively for trajectories, and mAP for activity ("Act"). The unit of the numbers are meters. As reported by [193], constant velocity can already achieve relatively good results for both short-term and long-term prediction, in terms of single-future ($minADE_1$/$minFDE_1$) metrics. Comparing STGAT and STGCNN, we find that STGCNN is slightly better on single-future metrics, but significantly worse on multi-future metrics. Comparing SGAN and STGAT, SGAN is slightly better for short-term trajectory prediction, but STGAT is significantly better for multi-future long-term trajectory prediction. Our Next-GAT achieves significantly better results across all metrics and tasks. It is able to outperform the Next model thanks to the graph attention modeling and better action representations. Notably, Next-GAT achieves a significant 28% improvement over the second best method that are not in this thesis (STGAT) on $minADE_{20}$ for long-term trajectory prediction task, while only 17% improvement for short-term prediction. This validates the efficacy of our proposed method for long-term trajectory

prediction.

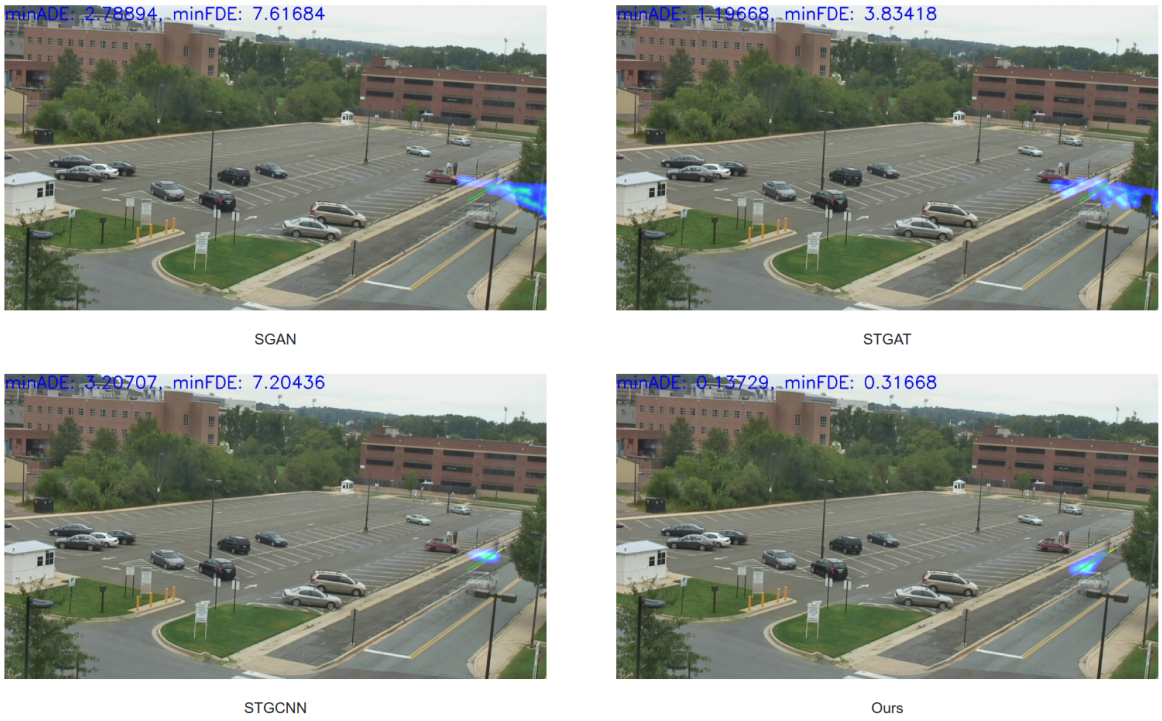

Figure 9.5: Qualitative analysis. The yellow trajectories are the history trajectory and the green trajectories are the ground truth future trajectories. The predicted trajectories are in green-blue heatmaps. See text for analysis.

User Studies. We are interested to find out whether the ADE/FDE differences in Table 9.1 are conceivable by humans. We conduct a user study to evaluate the multi-future long-term trajectory prediction performance between different baselines. For each baseline, we show the baseline's visualization and our method side by side (the order is randomized) and ask the annotator to select the better one. The visualization includes the yellow path, which is the observation trajectory, and the heatmaps, which are the multi-future trajectory prediction results of the corresponding method. The video frames of the future period are shown in the format of GIFs. Note that the ground truth future trajectory is not shown on purpose so that the human annotator would not ignore the video frames and directly compare the ground truth trajectory with the prediction results. The annotation instructions are as follows: Select the one with heatmap covering more of the person's path. If the same, select the one with smaller heatmap. See Figure 9.2 for the user interface. In total, results from 6 annotators with 100 pairs of samples each are recorded. On average, annotators have to look at each sample 3-4 times to decide. Sometimes due to the fact that two predictions are similar in human eyes, the selection could

Figure 9.6: Qualitative analysis. The yellow trajectories are the history trajectory and the green trajectories are the ground truth future trajectories. The predicted trajectories are in green-blue heatmaps. See text for analysis.

be close to a random choice between the two. The final results are shown in Figure 9.3. Both the mean and 95% confidence interval are shown in the figure. As we see, the results show that our method is significantly better than all the baselines and it is clearly perceivable by humans. However, different from the ranking in Table 9.1, the second best method is STGCNN rather than STGAT, suggesting a difference of 0.4 in terms of $minADE_{20}$ might not be noticeable by humans. To the best of our knowledge, we are the first to conduct such kind of user study to look into how ADE/FDE are perceived by humans. Note that inconceivable ADE/FDE differences may still be meaningful in many applications that rely on better trajectory prediction module.

Human Performance. In this section, we conduct a human performance experiment to see how well human can predict the pedestrian future trajectories. We randomly select 200 samples from the test set and put them into a web interface for human to annotate as shown in Figure 9.4. Fig. 9.4 (a) shows the interface with a GIF image showing the video frames of the observation period. The annotator is asked to use the pointer to click on the image where the pedestrian is going to appear at a one-second interval. The total length they are asked to predict is 12

seconds, the same as our long-term setting. Fig. 9.4 (b) and Fig. 9.4 (c) show what the image looks like after the annotator has clicked multiple locations. In Fig. 9.4 (d) we show the ground truth future trajectory as a reference. As we see, there is already about a half meter difference between the final location of the annotators' compared to the ground truth's. Table 9.1 shows the results. As we see, the numbers are only close to the constant velocity baseline, which is expected since in our experience, it is actually quite hard for human to predict exactly where the pedestrian is going to be in a fixed future time-frame.

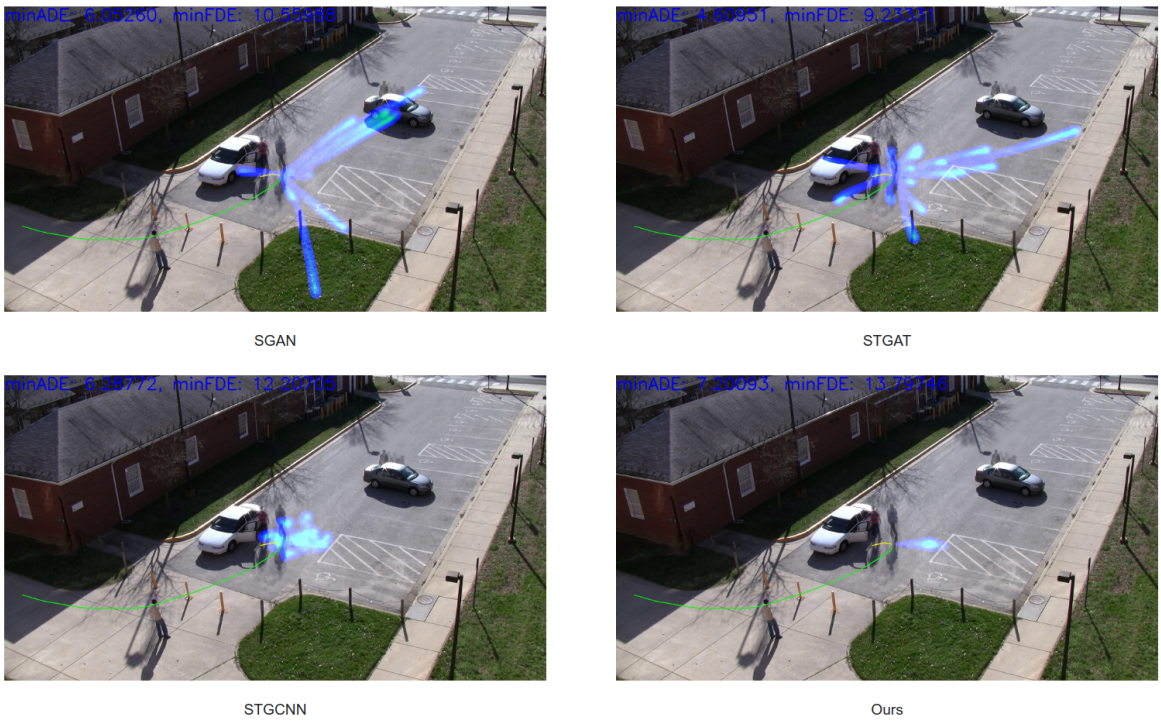

Figure 9.7: Error analysis. See text for details

Qualitative Analysis. We visualize some outputs of the 4 representative methods in Figure 9.5 and Figure 9.6. These are multi-future trajectories for long-term prediction. In each image, the yellow trajectories are the history trajectory and the green trajectories are the ground truth future trajectories. The predicted trajectories are in green-blue heatmaps. On the top-left corner of each image, $minADE_{20}$ and $minFDE_{20}$ of this sample are shown. In Figure 9.5, an example of a pedestrian that is walking is compared. In Figure 9.6, an example of a pedestrian that is static is compared. As we see, our model correctly generally puts probability mass where the ground truth is, and does not "waste" probability mass where there is no data. Looking at the other methods, STGAT has a more "spred-out" prediction pattern while STGCNN has a more "focused" probability mass. SGAN is closer to STGAT but its outputs have a dominated

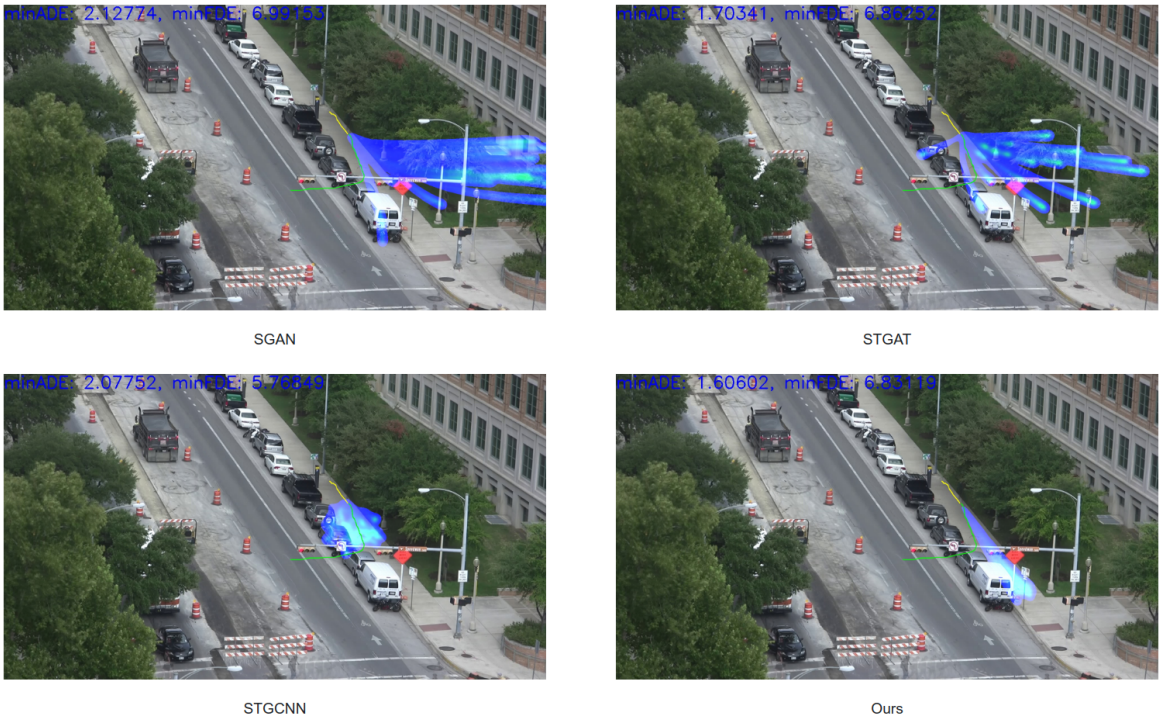

Figure 9.8: Error analysis. See text for details

direction. In Figure 9.6, STGAT basically has a "360" prediction direction.

Error Analysis. We analyze the most difficult test samples by ranking them based on average errors made by all four methods (three baselines and ours). We show some typical hard cases where most methods including ours fail in Figure 9.7, in Figure 9.8 and Figure 9.9. In Figure 9.7, the target person slowly moves near the white car while talking to the other person that enters the car, and then during the prediction period, the person starts moving to the left side. None of the methods is able to capture that, with STGAT outputing one closest path. This is a difficult case as there is no clear indication of the person's future direction. After looking through the video, we think that this would only be possible for any model to predict if all previous video history can be utilized, in which it shows that the target person goes from the left side to the white car. In Figure 9.8, the target person is walking along the sidewalk but in the middle of the long-term prediction period, the target person suddenly turns and decides to go across the street. None of the methods captures this turn. During the observation period there is no indication that the person is trying to cross the street. However, for a human driver seeing this pedestrian, it is common sense for them to assume that they have to be prepared for the pedestrian to cross/jay-walk. In Figure 9.9, the target person is riding a bike and a vehicle is coming towards him. This is a hard sample since it involves person-vehicle negotiation. In the

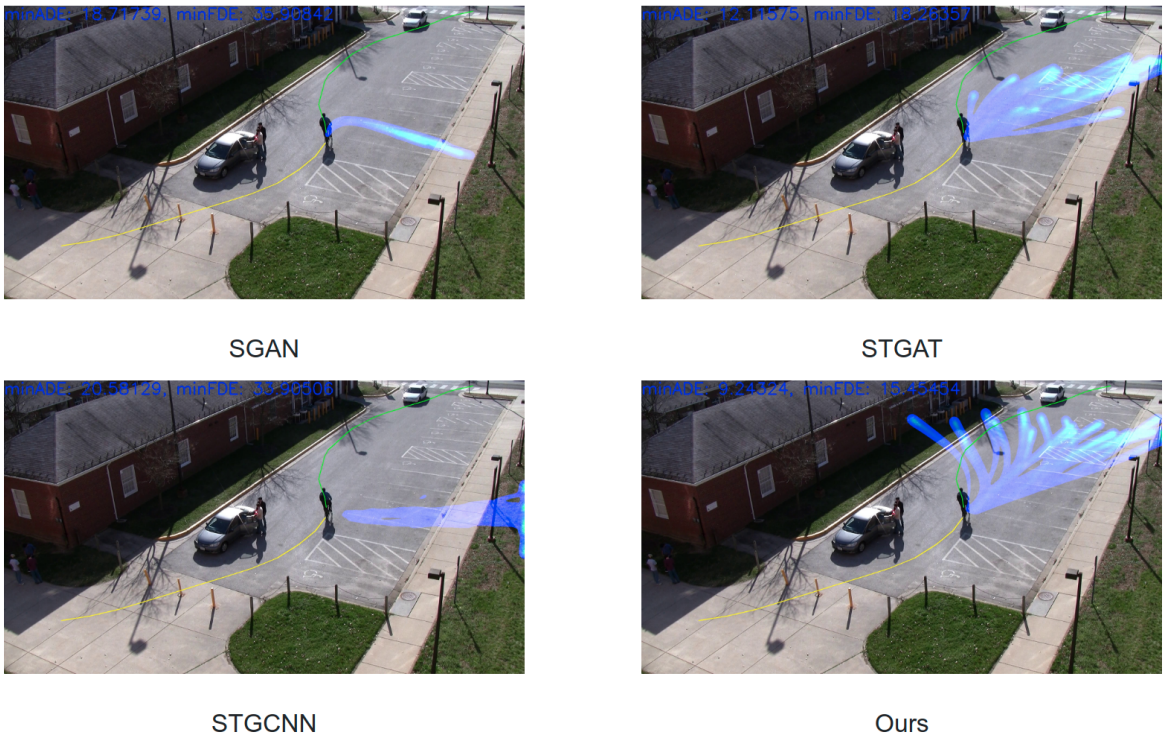

Figure 9.9: Error analysis. See text for details

last observation frame, we can see that the person has decided to turn left and then later get around the incoming vehicle to exit the parking lot. This kind of corner-cases in current dataset would need a high-level reasoning model to predict agent negotiations as well as common sense understanding (people on the sidewalk is possible to suddenly cross the road) to solve. We leave this to future work.

Prediction Diversity. In self-driving applications, we may consider the diversity of the prediction models so that they could cover all possible future trajectories to ensure safety. Previous metrics like $minADE_k$ do not measure diversity. In chapter 5, with the Forking Path dataset (with multi-modal ground truth trajectories) and a explicit probabilistic model, we could use the

	$minADE_{10}/minFDE_{10}$	$minASD_{10}/minFSD_{10}$
SGAN	2.76/5.42	0.353/0.642
STGAT	2.46/4.98	1.327/2.295
STGCNN	2.90/5.63	0.761/1.120
Next-GAT	1.71/3.54	0.192/0.375

Table 9.2: Prediction diversity measurements. See text for details

	Short-term Trajectory Prediction			long-term Trajectory Prediction		
	Act	Single-Future	Multi-Future	Act	Single-Future	Multi-Future
NN	-	7.32/13.54	-	-	15.29/30.00	-
Const. Vel.	-	2.76/5.76	-	-	8.35/17.89	-
SGAN	-	3.41/7.21	1.92/4.04	-	8.77/18.11	7.24/14.98
STGAT	-	5.05/10.43	2.00/4.15	-	14.75/29.51	7.71/15.68
STGCNN	-	4.79/8.56	3.36/6.33	-	14.60/27.42	11.54/22.63
Next	0.257	2.14/5.04	1.95/4.55	0.176	7.62/18.20	6.98/16.60
Next-GAT	**0.328**	**1.91/4.33**	**1.63/3.75**	**0.299**	**6.51/14.67**	**5.60/12.82**

Table 9.3: Main results on the MEVA-Trajectory dataset. The unit is in feet. See text for details.

Negative-Log-Likelihood (NLL) metric to measure both diversity, precision and recall. However in real-world video dataset like ActEV, only one ground truth trajectory is available. And in this chapter we are using generative model, which does not output probability for each trajectory. So to measure prediction diversity, we follow the work of Yuan et. al. [258]. They proposed to use minimum Average Self Distance (minASD) and minimum Final Self Distance (minFSD), which measures average L2 distance over all time steps or the final ones between a predicted sample and its closest neighbor prediction. In this way, repeated samples will be penalized. We hope that this metric can help users better understand the characteristics of each method. We show the results in Table 9.2. Following [258] we use K=10 per test sample. Unsuprisingly, Next-GAT model has the lowest self distances as our multi-future prediction models consist of the same models trained using different random initialization. We focus on the performance of single-future trajectory prediction for this model. We can also see that STGAT has the best diversity according to the metric, which is consistent with what we see in the qualitative examples in Fig. 9.5. In real-world deployment, such property (diversity) should be taken into account.

9.3.2 MEVA-Trajectory

Main Results. We show the main results in Table 9.3, where both short-term trajectory prediction and long-term trajectory prediction results are shown. The numbers are ADE/FDE respectively for trajectories, and mAP for activity ("Act"). The unit of the numbers are feet (0.3 meters). See Figure 8.8. From the results, we can see that our method is also able to

achieve the best among all metrics. Looking at the numbers of recent methods, we can see that STGCNN perform badly on this dataset in terms of multi-future long-term trajectory prediction, showing that this dataset is more difficult compared to ActEV. STGAT also performs badly on single-future long-term trajectory prediction, but its multi-modality makes up for multi-future prediction. Comparing our Next-GAT with Next, Next-GAT improves signifcantly on activity prediction and in turn it helps trajectory prediction as well.

Qualitative Analysis. We visualize the results and show them in Figure 9.10 and Figure 9.11. In each figure, the top four images show visualization of the trajectories from SGAN, STGAT, STGCNN and ours, respectively, and below shows the video frame example of this person. We also show the $minADE_{20}$ and $minFDE_{20}$, as well as the ground truth trajectory's length at the top-left corner of each visualization image. In Figure 9.10, we can see that Next-GAT outputs the most accurate path prediction. In Figure 9.11, both Next-GAT and STGAT correctly predict that the person is going to turn.

Ablation Experiments. We test various ablations of our Next-GAT model on single-future trajectory prediction to substantiate our design decisions. Results are presented in Table 9.4, where activity prediction mAP and minADE_1/minFDE_1 are shown. We verify three of our key designs by leaving the module out from the full model. (1) *graph attention:* Our model is built on top of graph attention to model the agents' interaction. We compare Next-GAT with Next, which does not use graph attention. As we see, the performance drops on trajectory prediction. (2) *STAN-Action module:* We test our model without the STAN-Action module and replace it with a single ResNet group. As we see, the significant performance drops on activity prediction verify the efficacy of this new module proposed in our study. (3) *Scene modeling:* We train Next-GAT without the scene understanding module. We see that performance is slightly worse if we do not have static scene modeling. Finally, we compare our model with a simple GRU-based Encoder-Decoder model with only trajectory inputs. We can see that motion information alone cannot bring good performance.

9.4 Summary

In this chapter, we present our Next-GAT model for long-term trajectory prediction in urban traffic scenes. Next-GAT utilizes graph attention with scene and action understanding module, which is crucial for long-term prediction. We show our method's efficacy on two major benchmarks, ActEV and our MEVA-Trajectory dataset.

	long-term Trajectory Prediction		
	Activity	$minADE_1$	$minFDE_1$
Next-GAT	0.299	6.51	14.67
Next	0.176	7.62	18.2
Next-GAT-ResNet	0.253	7.02	15.55
Next-GAT-noScene	0.280	6.88	15.78
GRU-EncodeDecode	-	9.69	20.97

Table 9.4: Ablation experiments on MEVA-Trajectory dataset. See text for details.

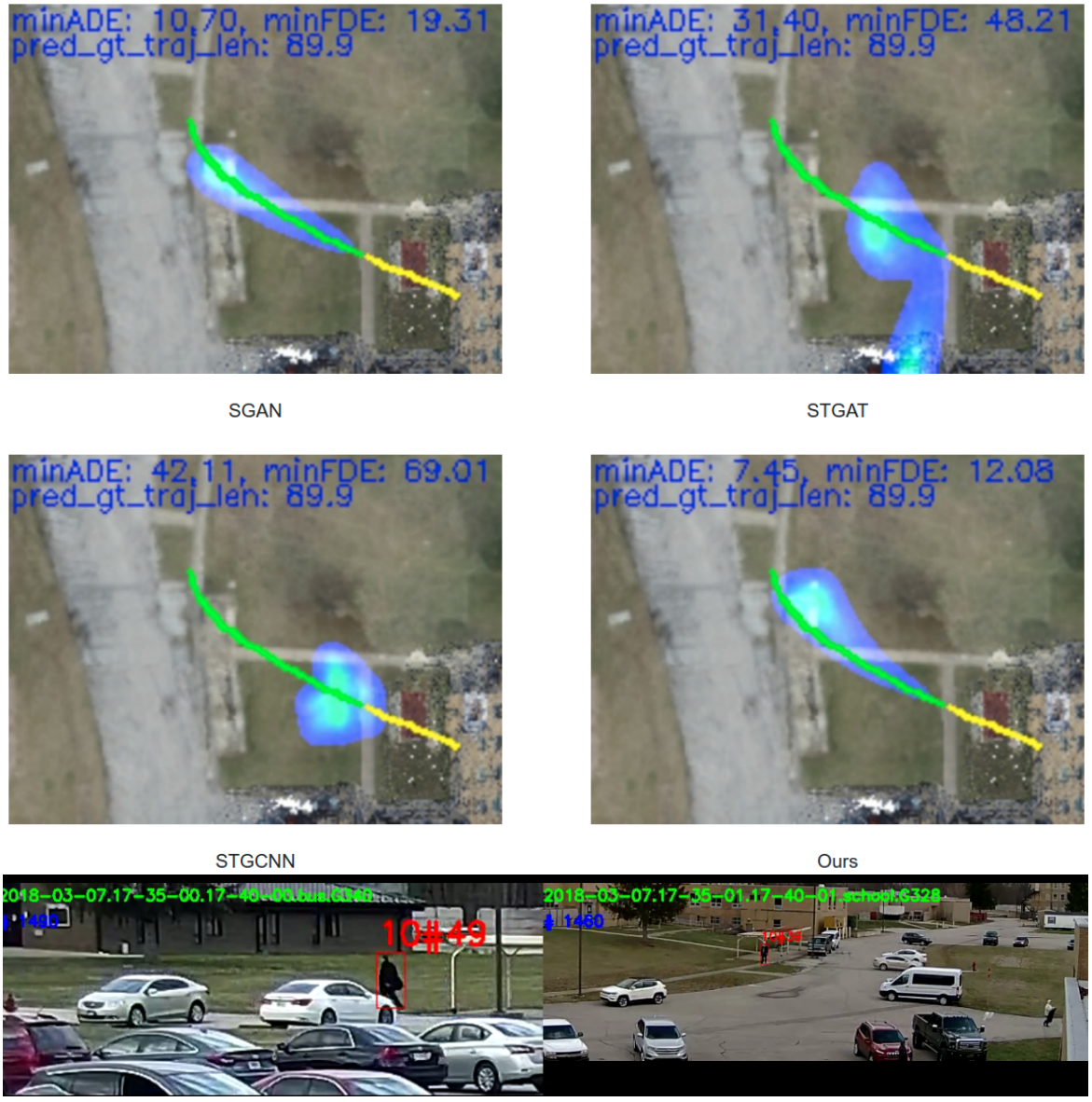

Figure 9.10: Qualitative analysis. The top four images show visualization of the trajectories from four methods and below shows the video frame example of the global track. See text for details

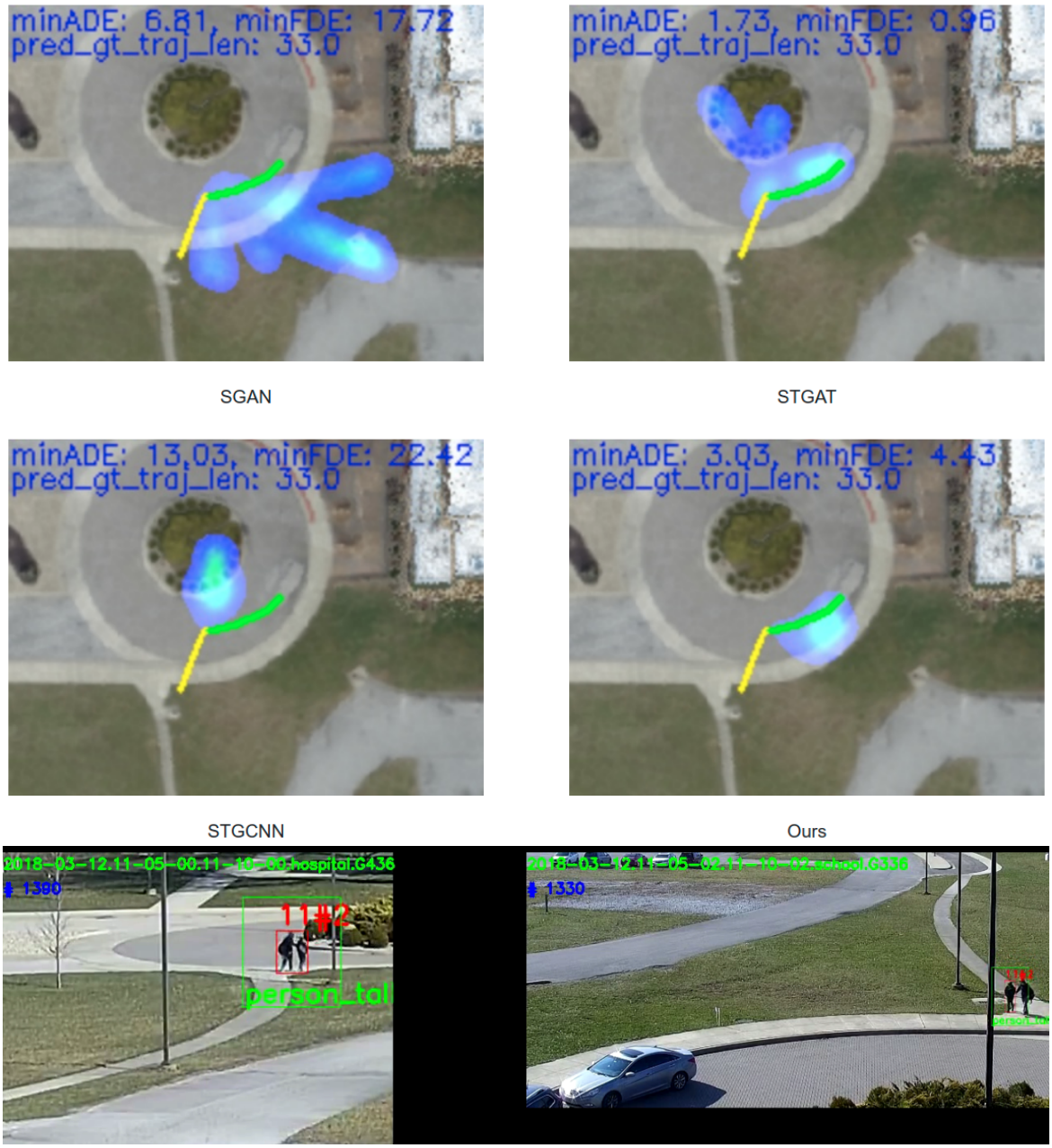

Figure 9.11: Qualitative analysis. The top four images show visualization of the trajectories from four methods and below shows the video frame example of the global track. See text for details

Part IV

Conclusions and Future Directions

Chapter 10

Conclusions

This thesis explores the problem of pedestrian future trajectory prediction. Our goal is to promote human safety in applications such as socially-aware robotics or autonomous driving. Specifically, throughout this thesis, we develop novel models and new datasets for three closely related tasks: *human action analysis, pedestrian trajectory prediction with scene semantics, and joint analysis of human actions and trajectory prediction.* In this thesis, we have demonstrated that, with scene understanding and enhanced behavioral representations, one can achieve accurate state-of-the-art performance in the challenging long-term trajectory prediction task in urban traffic scenes. Below, we summarize the contributions of each part of our work (please refer to section 1.5 for our overall impact). We then provide our key insights based on all the works we have done. Finally, we provide our recommendations for real-world applications using our models.

10.1 Contributions

10.1.1 Part I: Human Action Analysis

- We are one of the early works that study how we could better utilize weakly-supervised video data from the Internet. Our algorithm won several TRECVID challenges.
- We explore viewpoint-invariant representation for action recognition and detection, which is also one of the early works to tackle this problem in video.

10.1.2 Part II: Pedestrian Trajectory Prediction with Scene Semantics

- We develop a new model for multi-modal trajectory prediction and propose the first human-annotated benchmark for multi-future trajectory prediction.
- We propose a novel algorithm to build robust trajectory prediction models that could be transfer to different domains, which is an important under-explored problem in this research field.

10.1.3 Part III: Joint Analysis of Human Actions and Trajectory Prediction

- We propose a new human-annotated long-term trajectory prediction benchmark with multi-view video data that includes person-vehicle interactions and rich human activities in urban traffic scenes.
- We develop the first joint action and trajectory prediction model utilizes both scene understanding and viewpoint-invariant action representation for the challenging long-term trajectory prediction problem.

10.2 Key Insights

10.2.1 Limitations of datasets

Based on our experience working on datasets like ETH/UCY, ActEV, MEVA-Trajectory (chapter 8), we have noticed that these datasets based on outdoor scenes may not include enough critical corner cases (See Fig. 9.8, Fig. 9.9, Fig. 10.1) that are important for safety. This could be supported by the fact that in many experiments (chapter 7, chapter 9), simple constant velocity method can already achieve good results. We also observe that after removing person pose features (from the Next model in chapter 7 to the Next-GAT model in chapter 9), the performance is not affected, which is counter-intuitive as drivers when driving in parking lot they do utilize pedestrians' poses to read their intentions, etc. The main reason for this might be due to the fact that the datasets we use only has a very small portion of the trajectories that require the models to understand multi-agent negotiations. For example, in Fig. 10.1 (a), we can see that in this scenario, the target person is riding a bike and a car is coming to the person's direction. The ground truth future trajectory suggests that the person has eventually moved to the left to

get around the car's expected future path. Such person-vehicle negotiation might be intuitive for humans but it is hard for models to learn, especially when this kind of data is rare. See Fig. 9.9 in chapter 9 to see how other recent baselines perform in this case. One way to mitigate this is to find out these corner cases and use them for both training and evaluation. We have explored this by looking at samples that all baselines fail (see error analysis in subsection 9.3.1). However, more work needs to be done and datasets with rich multi-agent negotiations need to be collected to solve these corner/long-tail situations in traffic scenes. The PIE dataset [174] is a step towards this direction but the number of scenes and viewpoints are limited.

10.2.2 Limitations of training schemes

In all of our model training, we use only the positive examples to train. Unreasonable trajectory predictions, e.g. predicting person walking through vehicles, are not specially penalized. We could add hand-crafted heuristic based on the semantic segmentation classes like agent cannot walk through vehicles and water. For a more long-term vision, physical constraint in 3D simulator could be built into the training process for more general cases. On the other hand, we have only tried training with real-world video samples, where only one ground truth trajectory for each scenario is provided. We have not tried training with the Forking Paths dataset, where multiple positive samples are provided at a time. Using this dataset for training could allow the model to generalize better on possible predictions.

10.2.3 Limitations of models using scene semantics

During our examination of hard cases (see error analysis in subsection 9.3.1), we have also noticed error cases where the model does not utilize the scene semantics as expected to predict the correct trajectory. As we see in Fig. 10.1 (b), the person walks straight towards a crosswalk but he is actually going to turn left onto the sidewalk. Our model (and other baselines as well) predicts the trajectory solely based on the observation trajectory and does not use sidewalk semantic information. Although we have kept the spatial dimension of scene semantic segmentation features when inputting into our model (method like SoPhie [191] uses segmentation model's fully connected layer output, which could be even harder to learn), the model seems to not fully capture the correlations. We think that it may be worthwhile to revisit simple Hidden Markov Model as proposed in [103] to better associate directly scene classes and trajectories.

10.2.4 Evaluation metrics

Throughout this thesis, we have utilized different metrics to evaluate models capabilities for single-future prediction and multi-future prediction (section 9.3). To ensure safety, diversity of multi-modal prediction model should also be measure. Metrics like $minADE_k$ do not measure diversity. In chapter 5, with the Forking Path dataset (with multi-modal ground truth trajectories) and a explicit probabilistic model, we could use the Negative-Log-Likelihood (NLL) metric to measure both diversity, precision and recall. However in real-world video dataset like ActEV, only one ground truth trajectory is available. And many generative models, which do not output probability for each trajectory, cannot be evaluated using NLL metric. So following the work of Yuan et. al. [258] to measure prediction diversity, we use minimum Average Self Distance (minASD) and minimum Final Self Distance (minFSD), which measures average L2 distance over all time steps or the final ones between a predicted sample and its closest neighbor prediction, in subsection 9.3.1. In this way, repeated samples will be penalized. In conclusion, we believe that NLL is a better metric to use when possible, and minADE and minASD should both be considered in other situations to have a more comprehensive understanding of the model characteristics, which could be crucial during real-world deployment.

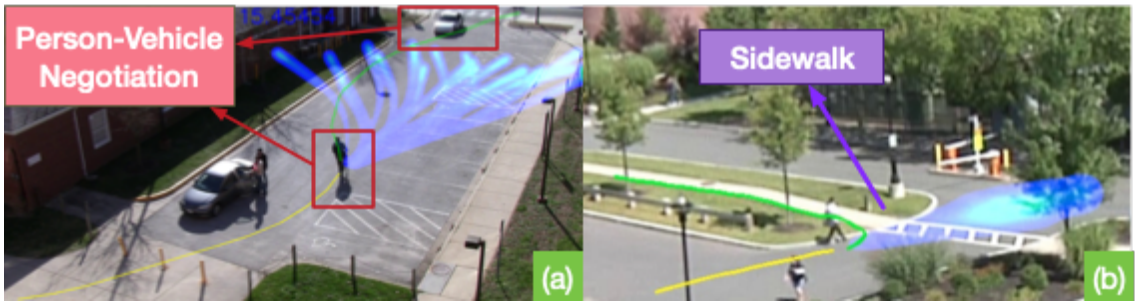

Figure 10.1: Important corner cases examples. See text for details.

10.3 Recommendations for Real-World Applications

In this section we briefly discuss recommendations for applying our models to an unseen dataset and the possibilities of achieving real-time prediction during test time.

Training and testing on an unseen dataset. We recommend using our SimAug-trained (chapter 6) model when trying to predict pedestrian trajectories on an unseen dataset. Our

SimAug algorithm is specially designed to optimize the domain transfer ability and our experiments show that it works well on unseen cameras like top-down view and ego-centric view. When extra training data could be available, we recommend using at many annotations as the ActEV dataset with 12 hours of trajectories, with which models can achieve within one meter average errors for 5-second prediction and within two meters for 12-second prediction (see subsection 9.3.1). The training time is estimated to be 8 hours with a middle-to-high-end GPU like GTX 1080 TI. When human annotations are not readily available, we recommend follow our instructions when building the MEVA-Trajectory dataset (chapter 8), in which we have minimized manual effort with state-of-the-art tracking and re-identification systems.

Real-Time Processing. As shown in chapter 2, our object detection and tracking system can achieves real-time speed if we process about 5 video frames per second (Table 2.4) given a relatively new 4-GPU machine listed in Table 2.3. Recall that in this work we only need 2.5 frame-per-second observation features to achieve the good results we have shown thoughout this thesis. The prediction inference time is about 10x faster than real-time, which means that with our efficient object detection and tracking system, one can easily build a real-time end-to-end system with our prediction models. One of our open-source project [1] already provides an example code base using the Next model (chapter 7).

[1] https://github.com/JunweiLiang/social-distancing-prediction

Chapter 11

Vision and Future Directions

In this thesis, we have propose novel deep learning models that utilize enhanced contextual cues like scene semantics and human actions for trajectory prediction in urban traffic scenes. Our research goal is to promote human safety in applications such as robotics or autonomous driving. We have explored multi-modal trajectory prediction and robust learning algorithms using simulation-augmentation. These are important directions as they provides better traffic safety if used in diverse environment. To push the boundary even further, we have proposed a long-term trajectory prediction dataset and models for joint action and trajectory prediction in urban traffic scenes. In summary, we have answered the key research question of how to build a robust trajectory prediction system with enhanced semantic context understanding for urban traffic scenes. Below, we outline several important future research directions that we have not studied, including some short-term directions related to applications of trajectory prediction and long-term directions that require novel model and algorithm to solve.

Short-Term Future Directions - Applications

First-person View. We have not explored trajectory prediction in first-person view videos. This may be difficult and different from our problem setting as first-person view videos may often include ego-motions that the models have to take into account when extracting behavioral representations for prediction. In future applications like social-aware robots that could navigate among humans, first-person-view trajectory prediction is crucial.

Long-tail Cases. We have not specifically tackled the challenging long-tail cases exist in traffic scenes [147]. Rare events like traffic accidents and even terrorist attack may significantly alter the behaviors of pedestrians, causing current models to fail in such scenarios. This is difficult

to solve as data collection may not be possible. It is important for prediction models to handle such rare cases to ensure traffic safety at all times.

Computation-accuracy Trade-off. We have not systematically studied how to balance computation cost and accuracy with our models. It is difficult because there are a lot of alternative models with different level of performance for object detection and tracking, scene semantic segmentation and action feature representations. We believe this is important as in applications like self-driving cars and social robots, on-board computing resources are limited. A clear computation-accuracy trade-off would be beneficial in real-world model deployment.

Trajectory Prediction in Sports. We have not looked into trajectory prediction in sports. For example, trajectory prediction in basketball or soccer can be used to help coaches with their decision makings. Trajectory prediction in sports require specific in-domain prior knowledge when building an effective model. It would be interesting to explore whether complex prediction algorithms can be generalized to different sports.

Crowd Dynamics Estimation for Public Safety Monitoring. Trajectory prediction can enable analysis and control of crowd flow in populated areas such as malls and airports by combining other crowd dynamics analysis tools like crowd counting [38] (see this news story [1] from the Washington Post how our crowd dynamics analysis has been used in real-world scenario). With crowd dynamics prediction and estimation, public safety events like stampede and riot could be alerted to the authorities and actions could be taken early to save lives.

Long-Term Future Directions - Model & Algorithm

Unifying Long-term and Short-term Trajectory Prediction. In chapter 9 we study long-term trajectory prediction with a fixed-length horizon in the experimental setup. With the grow of datasets and better algorithm, the length of the horizon is likely to expand. Meanwhile some long-term cues could be useful for short-term prediction. We believe that a multi-task learning framework of long-term and short-term trajectory prediction could benefit from different kinds of behavioral cues. Models with flexible prediction horizon can also be beneficial to real-world applications as well.

Behaviors of Different Population. We have not looked into the difference of behaviors of different population. For example, traffic scenes in India may be distinctively different from

[1] https://www.washingtonpost.com/investigations/interactive/2021/dc-police-records-capitol-riot/

the U.S, making models trained from one place difficult to transfer to another. This would be important because as self-driving cars become more and more common, on-board safety systems like trajectory prediction should be able to maintain the same safety standard around the world.

Unifying Vehicle and Pedestrian Trajectory Prediction. Currently in most self-driving systems pedestrian trajectory and vehicle trajectory prediction are done separately with different models. It is difficult to use one trajectory prediction for another task as vehicle and pedestrians behave differently. For example, vehicles have lane constrains while pedestrians do not. A unify model that takes into account these differences may be beneficial for system efficiency and better prediction accuracy as these two types of traffic actors have frequent interactions in real-world situation.

Common Sense Reasoning for Long-term Future Prediction. We have not explored common sense reasoning with high-level scene attributes for trajectory prediction. For example, if the scene is a school zone, we can expect the vehicles to be slow and students are present. If the scene is near a stadium and there is a game going on, we can expect people with sports jersey to be generally heading the stadium's direction. This is important information to better improve trajectory prediction as human drivers can comprehend such situation effortlessly. Ultimately, we believe it is possible to predict pedestrian trajectories minutes into the future with high-level reasoning, as shown in the Example of Figure 11.1.

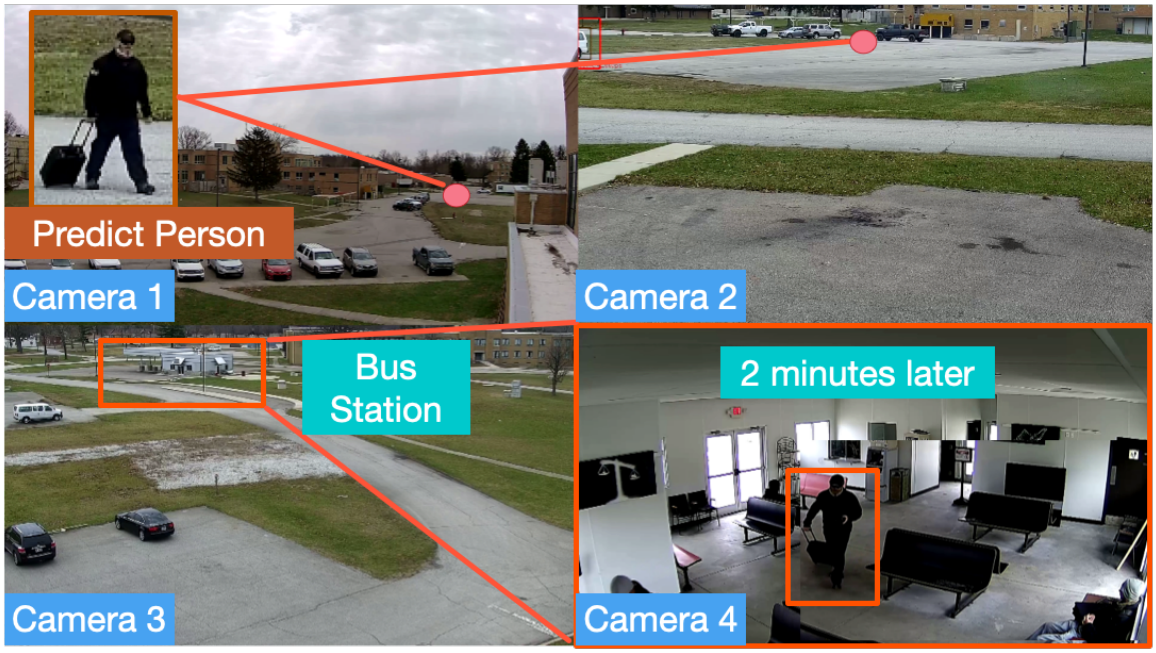

Figure 11.1: Example of future trajectory prediction using high-level reasoning. In two of the cameras we can see a person with a luggage, and knowing there is a bus station near the scene, the model should be equipped with a prior predicting that the person is going to appear in that location.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 2

[2] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016. 3.1

[3] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *CVPR*, 2016. 1.1, 5.1, 5.3, 5.4.1, **??**, **??**, 5.5, 6.1, 6.4.1, 6.4.2, 6.4.3, 6.2, 6.4.4, 6.3, 6.5, 7.1, 7.1, 7.2, 7.2.1, 7.2.3, 7.3.1, 7.3.1, **??**, 7.3.3, 7.4, 9.3

[4] Mohammad Sadegh Aliakbarian, Fatemehsadat Saleh, Mathieu Salzmann, Basura Fernando, Lars Petersson, and Lars Andersson. Encouraging lstms to anticipate actions very early. 2017. 7.4

[5] Javad Amirian, Jean-Bernard Hayet, and Julien Pettré. Social ways: Learning multi-modal distributions of pedestrian trajectories with gans. In *CVPRW*, 2019. 5.5

[6] Javad Amirian, Bingqing Zhang, Francisco Valente Castro, Juan Jose Baldelomar, Jean-Bernard Hayet, and Julien Pettre. Opentraj: Assessing prediction complexity in human trajectories datasets. In *ACCV*, 2020. 1.5

[7] George Awad, Asad Butt, Keith Curtis, Jonathan Fiscus, Afzal Godil, Alan F. Smeaton, Yvette Graham, Wessel Kraaij, Georges Quénot, Joao Magalhaes, David Semedo, and Saverio Blasi. Trecvid 2018: Benchmarking video activity detection, video captioning and matching, video storytelling linking and video search. In *TRECVID*, 2018. 5.1, 5.3, 5.4, 5.4.2, 6.1, 6.4, 7.1, 7.3, 7.3.1

[8] George Awad, Asad A Butt, Keith Curtis, Yooyoung Lee, Jonathan Fiscus, Afzal Godil, Andrew Delgado, Jesse Zhang, Eliot Godard, Lukas Diduch, et al. Trecvid 2019: An evaluation campaign to benchmark video activity detection, video captioning and matching, and video search & retrieval. *arXiv preprint arXiv:2009.09984*, 2020. 2, 2.1, 2.3

[9] Inhwan Bae and Hae-Gon Jeon. Disentangled multi-relational graph convolutional network for pedestrian trajectory prediction. In *AAAI*, 2021. 1.5

[10] Slawomir Bak, Peter Carr, and Jean-Francois Lalonde. Domain adaptation through synthesis for unsupervised person re-identification. In *ECCV*, 2018. 6.5

[11] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018. 5.1, 5.5, 6.1, 6.4.2, 6.5

[12] Mokhtar S Bazaraa, Hanif D Sherali, and Chitharanjan M Shetty. *Nonlinear programming: theory and algorithms*. John Wiley & Sons, 2013. 3.2.3

[13] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, 2009. 3.1, 3.4

[14] Alessia Bertugli, Simone Calderara, Pasquale Coscia, Lamberto Ballan, and Rita Cucchiara. Ac-vrnn: Attentive conditional-vrnn for multi-future trajectory prediction. *arXiv preprint arXiv:2005.08307*, 2020. 1.5

[15] Huikun Bi, Zhong Fang, Tianlu Mao, Zhaoqi Wang, and Zhigang Deng. Joint prediction for kinematic trajectories in vehicle-pedestrian-mixed scenes. In *ICCV*, 2019. 1.5

[16] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003. 3.2.2

[17] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *CVPR*, 2017. 6.5

[18] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu,

Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019. 5.3, 5.5, 6.5, 8.1

[19] Yingfeng Cai, Lei Dai, Hai Wang, Long Chen, Yicheng Li, Miguel Angel Sotelo, and Zhixiong Li. Pedestrian motion trajectory prediction in intelligent driving from far shot first-person perspective video. *IEEE Transactions on Intelligent Transportation Systems*, 2021. 1.5

[20] Liangliang Cao, Zicheng Liu, and Thomas S. Huang. Cross-dataset action detection. In *CVPR*, 2010. 4.4

[21] João Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In *CVPR*, 2017. 4.1, 4.4

[22] Joao Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. A short note on the kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987*, 2019. 4.3.1, 4.3.1

[23] Sergio Casas, Abbas Sadat, and Raquel Urtasun. Mp3: A unified model to map, perceive, predict and plan. In *CVPR*, 2021. 1.5

[24] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*, 2019. 1.1, 5.1, 5.2.3, 5.3, 5.4, 5.4, 5.5, 6.1, 6.4.2

[25] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *CVPR*, 2019. 5.5, 6.1, 6.4, 6.5

[26] Yu-Wei Chao, Sudheendra Vijayanarasimhan, Bryan Seybold, David A Ross, Jia Deng, and Rahul Sukthankar. Rethinking the faster r-cnn architecture for temporal action localization. In *CVPR*, 2018. 4.4

[27] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014. 3.2.1

[28] Bo Chen, Decai Li, Yuqing He, and Chunsheng Hua. Scr-graph: Spatial-causal relationships based graph reasoning network for human action prediction. *arXiv preprint arXiv:1912.05003*, 2019. 1.5

[29] Hanting Chen, Yunhe Wang, Chang Xu, Zhaohui Yang, Chuanjian Liu, Boxin Shi, Chunjing Xu, Chao Xu, and Qi Tian. Data-free learning of student networks. In *ICCV*, 2019. 6.5

[30] Jia Chen, Junwei Liang, Han Lu, Shoou-I Yu, and Alexander G Hauptmann. Videos from the 2013 boston marathon: An event reconstruction dataset for synchronization and localization. 2016. 1.2.3

[31] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 5.2.1, 5.4.1, 6.3.2

[32] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 1.1, 7.2.3, 7.3.2, 9.2.4

[33] Xinlei Chen and Abhinav Gupta. Webly supervised learning of convolutional networks. In *ICCV*, 2015. 3.1, 3.4

[34] Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. Neil: Extracting visual knowledge from web data. In *Proceedings of the IEEE International Conference on Computer Vision*, 2013. 3.4

[35] Yunpeng Chen, Marcus Rohrbach, Zhicheng Yan, Yan Shuicheng, Jiashi Feng, and Yannis Kalantidis. Graph-based global reasoning networks. In *CVPR*, 2019. 4.4

[36] Yong Cheng, Lu Jiang, and Wolfgang Macherey. Robust neural machine translation with doubly adversarial inputs. *ACL*, 2019. 6.5

[37] Yong Cheng, Lu Jiang, Wolfgang Macherey, and Jacob Eisenstein. Advaug: Robust data augmentation for neural machine translation. In *ACL*, 2020. 6.1, 6.5

[38] Zhi-Qi Cheng, Jun-Xiu Li, Qi Dai, Xiao Wu, and Alexander G Hauptmann. Learning spatial awareness to improve crowd counting. In *ICCV*, 2019. 1.2.3, 11

[39] Wongun Choi and Silvio Savarese. Understanding collective activitiesof people from videos. *IEEE transactions on pattern analysis and machine intelligence*, 36(6):1242–1257, 2014. 7.4

[40] Kellie Corona, Katie Osterdahl, Roderic Collins, and Anthony Hoogs. Meva: A large-scale multiview, multimodal video dataset for activity detection. In *WACV*, 2021. 4.3, 4.3.2, 8.1

[41] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *NeurIPS*, 2016. 2

[42] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei.

Deformable convolutional networks. In *ICCV*, 2017. 4.4

[43] Xiyang Dai, Bharat Singh, Guyue Zhang, Larry S Davis, and Yan Qiu Chen. Temporal context network for activity localization in videos. In *ICCV*, 2017. 2.1

[44] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *CVPRW*, 2018. 5.5

[45] César Roberto de Souza, Adrien Gaidon, Yohann Cabon, and Antonio Manuel López. Procedural generation of videos to train deep action recognition networks. In *CVPR*, 2017. 5.5, 6.1, 6.5

[46] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 3.1

[47] Nachiket Deo and Mohan M Trivedi. Trajectory forecasts in unknown environments conditioned on grid-based plans. *arXiv preprint arXiv:2001.00735*, 2020. 6.1, 6.3.2, 6.4.2, 6.4.3, 6.3, 6.5

[48] Ali Diba, Vivek Sharma, Luc Van Gool, and Rainer Stiefelhagen. Dynamonet: Dynamic action and motion network. In *CVPR*, 2019. ??

[49] Santosh K Divvala, Alireza Farhadi, and Carlos Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *CVPR*, 2014. 3.4

[50] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. *arXiv preprint arXiv:1711.03938*, 2017. 5.1, 5.3, 5.5, 6.1, 6.3.2, 6.5

[51] Lixin Duan, Dong Xu, IW-H Tsang, and Jiebo Luo. Visual event recognition in videos by learning from web data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(9):1667–1680, 2012. 3.4

[52] Kevin Duarte, Yogesh Rawat, and Mubarak Shah. Videocapsulenet: A simplified network for action detection. In *NeurIPS*, 2018. 4.4

[53] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 2

[54] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. RMPE: Regional multi-person pose estimation. In *ICCV*, 2017. 7.2.2, 7.3.1

[55] Christoph Feichtenhofer, Axel Pinz, and Richard P. Wildes. Spatiotemporal residual net-

works for video action recognition. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *NeurIPS*, 2016. 4.1

[56] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, 2019. 4.1, 4.1, 4.2.5, 4.2.6, 4.3.1, **??**, **??**, 4.3.1, 4.3.2, 4.3.2, **??**, 4.5, **??**, **??**, **??**, 4.4

[57] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*, 2016. 5.3, 5.5, 6.1, 6.3.2, 6.5

[58] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016. 6.5

[59] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 5.3, 8.1, **??**

[60] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Home: Heatmap output for future motion estimation. *arXiv preprint arXiv:2105.10968*, 2021. 1.5

[61] Joshua Gleason, Rajeev Ranjan, Steven Schwarcz, Carlos Castillo, Jun-Cheng Chen, and Rama Chellappa. A proposal-based solution to spatio-temporal action detection in untrimmed videos. In *WACV*, 2019. 2.1

[62] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011. 4.2.2

[63] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 5.5

[64] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 6.1, 6.4.2, 6.1, 6.5

[65] Jochen Gorski, Frank Pfeuffer, and Kathrin Klamroth. Biconvex sets and optimization with biconvex functions: a survey and extensions. *Mathematical Methods of Operations Research*, 66(3):373–407, 2007. 3.2.3

[66] Chunhui Gu, Chen Sun, David A Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, et al. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *CVPR*, 2018. 4.1, 4.2.6, 4.3, 4.3.1, 4.3.1

[67] Agrim Gupta, Justin Johnson, Silvio Savarese, Li Fei-Fei, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *CVPR*, 2018. 1.1, 5.1, 5.3, 5.4, 5.4.1, **??**, **??**, **??**, **??**, 5.5, 6.1, 6.4.2, 6.4.3, 6.2, 6.4.4, 6.3, 7.1, 7.1, 7.2, 7.2.1, 7.2.3, 7.2.4, 7.2.6, 7.3.1, 7.3.1, 7.3.1, **??**, **??**, 7.3.3, 7.3.3, 7.4, 8.1, 9.1, 9.3, 9.3

[68] Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947–951, 2000. 4.2.2

[69] Xintong Han, Bharat Singh, Vlad I Morariu, and Larry S Davis. Fast automatic video retrieval using web images. *arXiv preprint arXiv:1512.03384*, 2015. 3.3.1, 3.2, 3.4

[70] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1.1, 2.2.1, 4.1, 4.3.1

[71] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 1.1, 2, 2.2.1, 4.2.6, 4.3.1, 7.2.2

[72] Nicolas Heess, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, Martin Riedmiller, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017. 5.5, 6.5

[73] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence*, 37(3):583–596, 2014. 2.2.2

[74] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 5.4.1

[75] Matthew Hoffman, Francis R Bach, and David M Blei. Online learning for latent dirichlet allocation. In *advances in neural information processing systems*, pages 856–864, 2010. 3.2.2

[76] Joey Hong, Benjamin Sapp, and James Philbin. Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions. In *CVPR*, 2019. 5.5, 6.4.2, 6.5

[77] Rui Hou, Chen Chen, and Mubarak Shah. Tube convolutional neural network (t-cnn) for action detection in videos. In *ICCV*, 2017. 2.1, 4.4

[78] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *CVPR*, 2018. 7.2.3, 9.2.5

[79] Gary B. Huang, Marwan A. Mattar, Honglak Lee, and Erik G. Learned-Miller. Learning to align from scratch. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *NeurIPS*, 2012. 4.2, 4.4

[80] Linjiang Huang, Yan Huang, Wanli Ouyang, and Liang Wang. Part-aligned pose-guided recurrent network for action recognition. *Pattern Recognition*, 92:165–176, 2019. 4.4

[81] Po-Yao Huang, Junwei Liang, Jean-Baptiste Lamare, and Alexander G Hauptmann. Multimodal filtering of social media for temporal monitoring and event analysis. In *ICMR*, 2018. 1.2.3

[82] Yingfan Huang, Huikun Bi, Zhaoxin Li, Tianlu Mao, and Zhaoqi Wang. Stgat: Modeling spatial-temporal interactions for human trajectory prediction. In *ICCV*, 2019. 9.1, 9.3, 9.3

[83] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 4.2.2

[84] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *NeurIPS*, 2015. 4.2, 4.2.3, 4.4

[85] Ashesh Jain, Hema S Koppula, Bharad Raghavan, Shane Soh, and Ashutosh Saxena. Car that knows before you do: Anticipating maneuvers via learning temporal driving models. In *CVPR*, 2015. 7.4

[86] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *CVPR*, 2016. 7.4

[87] Nikita Jaipuria, Golnaz Habibi, and Jonathan P How. A transferable pedestrian motion prediction model for intersections with different geometries. *arXiv preprint arXiv:1806.09444*, 2018. 7.4

[88] Hueihan Jhuang, Juergen Gall, Silvia Zuffi, Cordelia Schmid, and Michael J Black. Towards understanding action recognition. In *ICCV*, 2013. 4.4

[89] Lu Jiang, Deyu Meng, Teruko Mitamura, and Alexander G Hauptmann. Easy samples first: Self-paced reranking for zero-example multimedia search. In *MM*, 2014. 3.4

[90] Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, Shiguang Shan, and Alexander Haupt-

mann. Self-paced learning with diversity. In *NIPS*, 2014. 3.4

[91] Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G Hauptmann. Self-paced curriculum learning. In *AAAI*, 2015. 1.4.1, 3.1, 3.2.2, 3.2.2, 3.2.3, 6.5

[92] Lu Jiang, Shoou-I Yu, Deyu Meng, Yi Yang, Teruko Mitamura, and Alexander G Hauptmann. Fast and accurate content-based semantic search in 100m internet videos. In *Proceedings of the 23rd ACM international conference on Multimedia*, 2015. 3.1

[93] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. *ICML*, 2018. 6.1, 6.3.3, 6.5

[94] Lu Jiang, Di Huang, Mason Liu, and Weilong Yang. Beyond synthetic noise: Deep learning on controlled noisy labels. In *ICML*, 2020. 6.1, 6.3.3, 6.5

[95] Yu-Gang Jiang, Zuxuan Wu, Jun Wang, Xiangyang Xue, and Shih-Fu Chang. Exploiting feature and class relationships in video categorization with regularized deep neural networks. *arXiv preprint arXiv:1502.07209*, 2015. 3.1, 3.3, 3.3.1, 3.4

[96] RE Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME, D*, 82:35–44, 1960. 5.1

[97] Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. In *CVPR*, 2019. 6.5

[98] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-sim: Learning to generate synthetic datasets. In *ICCV*, 2019. 6.5

[99] Vasiliy Karasev, Alper Ayvaci, Bernd Heisele, and Stefano Soatto. Intent-aware long-term prediction of pedestrian motion. In *ICRA*, 2016. 1.2.1, 1.6, 9.1

[100] Andrej Karpathy, George Toderici, Sachin Shetty, Tommy Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 3.1

[101] Kavindie Katuwandeniya, Stefan H Kiss, Lei Shi, and Jaime Valls Miro. Multi-modal scene-compliant user intention estimation for navigation. *arXiv preprint arXiv:2106.06920*, 2021. 1.5

[102] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics

human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 1.6, 4.3, 4.3.2, ??, 4.4

[103] Kris M Kitani, Brian D Ziebart, James Andrew Bagnell, and Martial Hebert. Activity forecasting. In *ECCV*, 2012. 1.1, 5.1, 5.5, 6.1, 7.1, 7.1, 7.2.1, 7.4, 10.2.3

[104] Alexander Kläser, Marcin Marszalek, and Cordelia Schmid. A spatio-temporal descriptor based on 3d-gradients. In Mark Everingham, Chris J. Needham, and Roberto Fraile, editors, *BMVC*, 2008. 4.1

[105] Julian Francisco Pieter Kooij, Nicolas Schneider, Fabian Flohr, and Dariu M Gavrila. Context-based pedestrian path prediction. In *ECCV*, 2014. 5.5, 6.5, 7.4

[106] Hema S Koppula and Ashutosh Saxena. Anticipating human activities using object affordances for reactive robotic response. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):14–29, 2016. 7.4

[107] Adam Kosiorek, Sara Sabour, Yee Whye Teh, and Geoffrey E Hinton. Stacked capsule autoencoders. In *NeurIPS*, 2019. 4.1, 4.2.2, 4.4

[108] Iuliia Kotseruba, Amir Rasouli, and John K Tsotsos. Benchmark for evaluating pedestrian action prediction. In *WACV*, 2021. 1.5

[109] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 3.3.1

[110] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011. 4.4

[111] M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *NIPS*, 2010. 3.1, 3.2.3, 3.3.1, 3.4

[112] M Pawan Kumar, Haithem Turki, Dan Preston, and Daphne Koller. Learning specific-class segmentation from diverse data. In *ICCV*, 2011. 3.2, 3.4, 3.6, 3.4

[113] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *ICLR*, 2017. 6.1, 6.3.3

[114] John Lambert, Ozan Sener, and Silvio Savarese. Deep learning under privileged information using heteroscedastic dropout. In *CVPR*, 2018. 6.5

[115] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008. 4.1

[116] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial

pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 5.2.1

[117] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *CVPR*, 2017. 1.1, 5.1, 5.5, 6.1, 6.4.2, 6.4.3, 6.3, 6.5

[118] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by example. In *Computer Graphics Forum*, pages 655–664. Wiley Online Library, 2007. 5.1, 5.3, 6.1, 6.5, 7.1, 7.3, 7.3.3, 8.1, **??**

[119] Ang Li, Meghana Thotakuri, David A Ross, João Carreira, Alexander Vostrikov, and Andrew Zisserman. The ava-kinetics localized human actions video dataset. *arXiv preprint arXiv:2005.00214*, 2020. 2.2.1, 4.3, 4.3.1, **??**, 4.3.1, 4.4

[120] Jiwei Li, Will Monroe, and Dan Jurafsky. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562*, 2016. 5.2.5, 5.4.1

[121] Yuke Li. Which way are you going? imitative decision learning for path forecasting in dynamic scenes. In *CVPR*, 2019. 1.1, 5.1, 5.3, 5.5, 6.1, 6.4.2, 6.4.3, 6.3, 6.5

[122] Junwei Liang, Qin Jin, Xixi He, Gang Yang, Jieping Xu, and Xirong Li. Semantic concept annotation of consumer videos at frame-level using audio. In *Pacific Rim Conference on Multimedia*, pages 113–122. Springer, 2014. 3.1

[123] Junwei Liang, Lu Jiang, Deyu Meng, and Alexander G Hauptmann. Learning to detect concepts from webly-labeled video data. In *IJCAI*, 2016. 3, 3.1, 3.2.2, 3.2.2, 6.5

[124] Junwei Liang, Desai Fan, Han Lu, Poyao Huang, Jia Chen, Lu Jiang, and Alexander Hauptmann. An event reconstruction tool for conflict monitoring using social media. In *AAAI*, 2017. 1.2.3, 5.5, 6.5

[125] Junwei Liang, Lu Jiang, and Alexander Hauptmann. Temporal localization of audio events for conflict monitoring in social media. In *ICASSP*, 2017. 1.2.3

[126] Junwei Liang, Lu Jiang, and Alexander Hauptmann. Webly-supervised learning of multimodal video detectors. In *AAAI*, 2017. 3

[127] Junwei Liang, Lu Jiang, Deyu Meng, and Alexander Hauptmann. Leveraging multi-modal prior knowledge for large-scale concept learning in noisy web data. In *ICMR*, 2017. 3

[128] Junwei Liang, Lu Jiang, Liangliang Cao, Li-Jia Li, and Alexander G Hauptmann. Focal visual-text attention for visual question answering. In *CVPR*, 2018. 5.5, 7.2.1, 7.2.4

[129] Junwei Liang, Jay D Aronson, and Alexander Hauptmann. Technical report of the video

event reconstruction and analysis (vera) system–shooter localization, models, interface, and beyond. *arXiv preprint arXiv:1905.13313*, 2019. 1.2.3

[130] Junwei Liang, Lu Jiang, Liangliang Cao, Yannis Kalantidis, Li-Jia Li, and Alexander G Hauptmann. Focal visual-text attention for memex question answering. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1893–1908, 2019. 5.5, 6.1, 7.2.1

[131] Junwei Liang, Lu Jiang, Juan Carlos Niebles, Alexander G Hauptmann, and Li Fei-Fei. Peeking into the future: Predicting future person activities and locations in videos. In *CVPR*, 2019. 1.1, 1.2.1, 5.1, 5.3, 5.4.1, **??**, 5.4.2, **??**, 5.5, 6.1, 6.3.2, 6.3.4, 6.4.2, 6.2, 6.4.4, 6.5, 7, 8.1, 9.1

[132] Junwei Liang, Lu Jiang, Kevin Murphy, Ting Yu, and Alexander Hauptmann. The garden of forking paths: Towards multi-future trajectory prediction. In *CVPR*, 2020. 5, 6.1, 6.3, 6.3.2, 6.3.3, 6.3.3, 6.3.4, 6.3.4, 6.4.1, 6.4.2, 6.1, 6.2, 6.4.4, 6.3, 6.5

[133] Xiaodan Liang, Si Liu, Yunchao Wei, Luoqi Liu, Liang Lin, and Shuicheng Yan. Towards computational baby learning: A weakly-supervised approach for object detection. In *ICCV*, 2015. 3.1, 3.3.1, 3.2, 3.4, 3.6, 3.4

[134] Chen-Hsuan Lin and Simon Lucey. Inverse compositional spatial transformer networks. In *CVPR*, 2017. 4.2, 4.2, 4.2.2, 4.4

[135] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *ICCV*, 2019. 4.4

[136] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 1.4.1, 2, 2.2.1, 4.3.1

[137] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 2.2.1, 5.2.1, 7.3.1

[138] Wenhe Liu, Guoliang Kang, Po-Yao Huang, Xiaojun Chang, Yijun Qian, Junwei Liang, Liangke Gui, Jing Wen, and Peng Chen. Argus: Efficient activity detection system for extended video analysis. In *WACVW*, 2020. 2, 2.3

[139] David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643*, 2015. 6.5

[140] Matthias Luber, Johannes A Stork, Gian Diego Tipaldi, and Kai O Arras. People tracking

with human motion predictions from social forces. In *ICRA*, 2010. 1.1, 5.1, 6.1, 6.5, 7.1

[141] Zelun Luo, Jun-Ting Hsieh, Lu Jiang, Juan Carlos Niebles, and Li Fei-Fei. Graph distillation for action detection with privileged modalities. In *ECCV*, 2018. 6.5

[142] Shugao Ma, Leonid Sigal, and Stan Sclaroff. Learning activity progression in lstms for activity detection and early detection. In *CVPR*, 2016. 7.4

[143] Wei-Chiu Ma, De-An Huang, Namhoon Lee, and Kris M Kitani. Forecasting interactive dynamics of pedestrians with fictitious play. In *CVPR*, 2017. 5.5

[144] Yecheng Jason Ma, Jeevana Priya Inala, Dinesh Jayaraman, and Osbert Bastani. Diverse sampling for normalizing flow based trajectory forecasting. *arXiv preprint arXiv:2011.15084*, 2020. 1.5

[145] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 6.4.2, 6.1, 6.5

[146] Osama Makansi, Eddy Ilg, Ozgun Cicek, and Thomas Brox. Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction. In *CVPR*, 2019. 1.1, 5.1, 5.3, 5.4, 5.4.1, 5.5, 6.4.2, 6.5

[147] Osama Makansi, Özgün Cicek, Yassine Marrakchi, and Thomas Brox. On exposing the challenging long tail in future prediction of traffic actors. *arXiv preprint arXiv:2103.12474*, 2021. 1.5, 11

[148] Karttikeya Mangalam, Ehsan Adeli, Kuan-Hui Lee, Adrien Gaidon, and Juan Carlos Niebles. Disentangling human dynamics for pedestrian locomotion forecasting with noisy supervision. *arXiv preprint arXiv:1911.01138*, 2019. 6.1, 6.5

[149] Huynh Manh and Gita Alaghband. Scene-lstm: A model for human trajectory prediction. *arXiv preprint arXiv:1808.04018*, 2018. 5.5, 7.1, 7.2.1, 7.2.5, 7.2.6, 7.3.3, 7.4

[150] Christopher D Manning, Prabhakar Raghavan, et al. *Introduction to information retrieval*, volume 1. 3.2.2

[151] Michael McCandless, Erik Hatcher, and Otis Gospodnetic. *Lucene in Action: Covers Apache Lucene 3.0*. Manning Publications Co., 2010. 3.2.2

[152] Deyu Meng and Qian Zhao. What objective does self-paced learning indeed optimize? *arXiv preprint arXiv:1511.06049*, 2015. 3.2.2, 3.2.3

[153] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed

representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013. 3.2.2

[154] T Mitchell, W Cohen, E Hruschka, P Talukdar, J Betteridge, A Carlson, B Dalvi, M Gardner, B Kisiel, J Krishnamurthy, et al. Never-ending learning. In *AAAI*, 2015. 3.4

[155] Abduallah Mohamed, Kun Qian, Mohamed Elhoseiny, and Christian Claudel. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *CVPR*, 2020. 9.1, 9.3, 9.3

[156] Milind Naphade, Zheng Tang, Ming-Ching Chang, David C Anastasiu, Anuj Sharma, Rama Chellappa, Shuo Wang, Pranamesh Chakraborty, Tingting Huang, Jenq-Neng Hwang, et al. The 2019 ai city challenge. In *CVPRW*, 2019. 8.2.1

[157] Curtis G Northcutt, Lu Jiang, and Isaac L Chuang. Confident learning: Estimating uncertainty in dataset labels. *arXiv preprint arXiv:1911.00068*, 2019. 6.5

[158] Sangmin Oh, Anthony Hoogs, Amitha Perera, Naresh Cuntoor, Chia-Chih Chen, Jong Taek Lee, Saurajit Mukherjee, JK Aggarwal, Hyungtae Lee, Larry Davis, et al. A large-scale benchmark dataset for event recognition in surveillance video. In *CVPR*, 2011. 1.6, 2, 2.1, 2.2.1, 5.1, 5.3, 5.4, 5.4.2, 6.1, 6.4, 6.5, 7.1, 7.3, 7.3.1, 8.1, **??**, 8.2.2, 8.2.2, 9.1

[159] Paul Over, Jon Fiscus, Greg Sanders, David Joy, Martial Michel, George Awad, Alan Smeaton, Wessel Kraaij, and Georges Quénot. Trecvid 2014–an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *Proceedings of TRECVID*, page 52, 2014. 3.1, 3.3.1

[160] Junting Pan, Siyu Chen, Zheng Shou, Jing Shao, and Hongsheng Li. Actor-context-actor relation network for spatio-temporal action localization. *arXiv preprint arXiv:2006.07976*, 2020. 4.2.5

[161] Bo Pang, Tianyang Zhao, Xu Xie, and Ying Nian Wu. Trajectory prediction with latent belief energy-based model. In *CVPR*, 2021. 1.5

[162] Stefano Pellegrini, Andreas Ess, and Luc Van Gool. Improving data association by joint modeling of pedestrian trajectories and groupings. In *ECCV*, 2012. 5.3, 7.1, 7.3, 7.3.3, 8.1, **??**

[163] Xiaojiang Peng, Limin Wang, Xingxing Wang, and Yu Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *Comput. Vis. Image Underst.*, 150:109–125, 2016. 4.4

[164] Tobias Plötz and Stefan Roth. Neural nearest neighbors networks. In *NeurIPS*, 2018. 5.4.1

[165] Atanas Poibrenski, Matthias Klusch, Igor Vozniak, and Christian Müller. M2p3: multimodal multi-pedestrian path prediction by self-driving cars with egocentric vision. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, pages 190–197, 2020. 1.5

[166] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukáš Burget, Ondřej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlíček, Yanmin Qian, Petr Schwarz, et al. The kaldi speech recognition toolkit. 2011. 3.2.1, 3.2.2

[167] Maria Priisalu, Ciprian Paduraru, Aleksis Pirinen, and Cristian Sminchisescu. Semantic synthesis of pedestrian locomotion. In *ACCV*, 2020. 1.5

[168] Siyuan Qi, Wenguan Wang, Baoxiong Jia, Jianbing Shen, and Song-Chun Zhu. Learning human-object interactions by graph parsing neural networks. In *ECCV*, 2018. 4.4

[169] William Qi. *Learning Representations for Safe Autonomous Movement*. PhD thesis, Carnegie Mellon University Pittsburgh, PA, 2020. 1.5

[170] Yijun Qian, Lijun Yu, Wenhe Liu, and Alexander G Hauptmann. Electricity: An efficient multi-camera vehicle tracking system for intelligent city. In *CVPRW*, 2020. 8.2.1

[171] Weichao Qiu, Fangwei Zhong, Yi Zhang, Siyuan Qiao, Zihao Xiao, Tae Soo Kim, and Yizhou Wang. Unrealcv: Virtual worlds for computer vision. In *ACM Multimedia*, 2017. 5.5, 6.5

[172] Ruijie Quan, Linchao Zhu, Yu Wu, and Yi Yang. Holistic lstm for pedestrian trajectory prediction. *IEEE transactions on image processing*, 30:3229–3239, 2021. 1.5

[173] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015. 5.2.4

[174] Amir Rasouli, Iuliia Kotseruba, Toni Kunic, and John K Tsotsos. Pie: A large-scale dataset and models for pedestrian intention estimation and trajectory prediction. In *ICCV*, 2019. 10.2.1

[175] Amir Rasouli, Mohsen Rohani, and Jun Luo. Pedestrian behavior prediction via multitask learning and categorical interaction modeling. *arXiv preprint arXiv:2012.03298*, 2020. 1.5

[176] Esteban Real, Jonathon Shlens, Stefano Mazzocchi, Xin Pan, and Vincent Vanhoucke. Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In *CVPR*, 2017. 2

[177] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *ICML*, 2018. 6.5

[178] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 2.1, 5.1, 5.2.3, 5.2.4, 7.2.5

[179] Nicholas Rhinehart and Kris M Kitani. First-person activity forecasting with online inverse reinforcement learning. In *ICCV*, 2017. 1.1, 5.1, 6.1

[180] Nicholas Rhinehart, Kris M Kitani, and Paul Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *ECCV*, 2018. 5.1, 5.3, 5.4, 5.5, 6.1, 6.5

[181] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. *arXiv preprint arXiv:1905.01296*, 2019. 5.3, 5.4, 5.5

[182] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *ECCV*, 2016. 5.5, 6.1, 6.5

[183] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *ECCV*, 2016. 8.1

[184] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *ECCV*, 2016. 5.3, 6.1, 6.3.2, 6.4, 6.5, 8.1, **??**

[185] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016. 5.3, 5.5, 6.3.2, 6.5

[186] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila, and Kai O Arras. Human motion trajectory prediction: A survey. *The International Journal of Robotics Research*, 39(8):895–935, 2020. 1.1, 1.1, 1.4.1, 1.4.2, 1.4.3, 1.6, 5.5, 6.5, 7.4

[187] Nataniel Ruiz, Samuel Schulter, and Manmohan Chandraker. Learning to simulate. *ICLR*, 2018. 1.1, 6.1

[188] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and

Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 3.2.2

[189] Abbas Sadat, Sergio Casas, Mengye Ren, Xinyu Wu, Pranaab Dhawan, and Raquel Urtasun. Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. In *ECCV*, 2020. 1.5

[190] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In *ICCV*, 2017. 1.1, 5.1, 7.4

[191] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, and Silvio Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. *arXiv preprint arXiv:1806.01482*, 2018. 5.3, 5.4.1, 5.5, 6.1, 6.3.2, 6.4.2, 6.4.3, 6.3, 6.5, 7.1, 7.2, 7.2.1, 7.2.6, 7.3.1, **??**, 7.3.3, 7.3.3, 7.4, 9.3, 10.2.3

[192] Amir Sadeghian, Ferdinand Legros, Maxime Voisin, Ricky Vesel, Alexandre Alahi, and Silvio Savarese. Car-net: Clairvoyant attentive recurrent network. In *ECCV*, 2018. 5.5, 6.5

[193] Christoph Schöller, Vincent Aravantinos, Florian Lay, and Alois Knoll. The simpler the better: Constant velocity for pedestrian motion prediction. *arXiv preprint arXiv:1903.07933*, 2019. 9.3, 9.3.1

[194] Christian Schüldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: A local SVM approach. In *ICPR*, 2004. 4.4

[195] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*, pages 621–635. Springer, 2018. 5.5, 6.5

[196] Tianmin Shu, Dan Xie, Brandon Rothrock, Sinisa Todorovic, and Song Chun Zhu. Joint inference of groups, events and human roles in aerial videos. In *CVPR*, 2015. 7.4

[197] Tianmin Shu, Sinisa Todorovic, and Song-Chun Zhu. Cern: confidence-energy recurrent network for group activity recognition. In *CVPR*, 2017. 7.4

[198] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV*, 2016. 4.3, 4.3.2

[199] Gunnar A Sigurdsson, Santosh Divvala, Ali Farhadi, and Abhinav Gupta. Asynchronous temporal fields for action recognition. In *Proceedings of the IEEE Conference on Computer*

Vision and Pattern Recognition, pages 585–594, 2017. 4.1

[200] Gunnar A Sigurdsson, Abhinav Gupta, Cordelia Schmid, Ali Farhadi, and Karteek Alahari. Charades-ego: A large-scale dataset of paired third and first person videos. *arXiv preprint arXiv:1804.09626*, 2018. 4.3, 4.3.2, **??**

[201] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3.2.2, 3.3.1

[202] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *NeurIPS*, 2014. 4.1, 4.4

[203] Ray Smith. An overview of the tesseract ocr engine. In *icdar*, pages 629–633. IEEE, 2007. 3.2.1, 3.2.2

[204] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 4.4

[205] Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. Baby steps: How "less is more" in unsupervised dependency parsing. *NIPS GRLL*, 2009. 3.4

[206] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *ICML*, 2015. 7.4

[207] Oily Styles, Arun Ross, and Victor Sanchez. Forecasting pedestrian trajectory with machine-annotated training data. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 716–721. IEEE, 2019. 6.1, 6.5

[208] Olly Styles, Tanaya Guha, and Victor Sanchez. Multiple object forecasting: Predicting future object locations in diverse environments. *arXiv preprint arXiv:1909.11944*, 2019. 6.1, 6.5

[209] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*, 2014. 3.3.1, 3.3.3, 3.5

[210] Chen Sun, Abhinav Shrivastava, Carl Vondrick, Kevin Murphy, Rahul Sukthankar, and Cordelia Schmid. Actor-centric relation network. In *ECCV*, 2018. 4.2.6, 4.3.1

[211] Chen Sun, Per Karlsson, Jiajun Wu, Joshua B Tenenbaum, and Kevin Murphy. Stochastic prediction of multi-agent interactions from partial observations. *arXiv preprint arXiv:1902.09641*, 2019. 5.5, 6.5

[212] Jianhua Sun, Qinhong Jiang, and Cewu Lu. Recursive social behavior graph for trajectory prediction. In *CVPR*, 2020. 1.5

[213] Shao-Hua Sun, Minyoung Huh, Yuan-Hong Liao, Ning Zhang, and Joseph J Lim. Multi-view to novel view: Synthesizing novel views with self-learned confidence. In *ECCV*, 2018. 6.5

[214] James Steven Supancic and Deva Ramanan. Self-paced learning for long-term tracking. In *CVPR*, 2013. 3.4

[215] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *CVPR*, 2020. 2, 2.2.1

[216] Kevin Tang, Vignesh Ramanathan, Li Fei-Fei, and Daphne Koller. Shifting weights: Adapting object detectors from image to video. In *NIPS*, 2012. 3.4

[217] Yichuan Charlie Tang and Ruslan Salakhutdinov. Multiple futures prediction. *arXiv preprint arXiv:1911.00997*, 2019. 1.1, 5.1, 5.5

[218] Graham W. Taylor, Rob Fergus, Yann LeCun, and Christoph Bregler. Convolutional learning of spatio-temporal features. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *ECCV*, 2010. 4.1

[219] Luca Anthony Thiede and Pratik Prabhanjan Brahma. Analyzing the variety loss in the context of probabilistic trajectory prediction. *arXiv preprint arXiv:1907.10178*, 2019. 1.1, 5.1, 5.3, 5.5

[220] Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. The new data and new challenges in multimedia research. *arXiv preprint arXiv:1503.01817*, 2015. 3.3, 3.3.1

[221] Yingli Tian, Liangliang Cao, Zicheng Liu, and Zhengyou Zhang. Hierarchical filtered motion for action recognition in crowded videos. *IEEE Trans. Syst. Man Cybern. Part C*, 42(3):313–323, 2012. 4.1

[222] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017. 1.1, 6.5

[223] Du Tran, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 4.1, 4.4

[224] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri.

A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 2018. 4.2, 4.2.2, ??, 4.4

[225] Hung Tran, Vuong Le, and Truyen Tran. Goal-driven long-term trajectory prediction. In *WACV*, 2021. 1.6, 9.1, 9.3

[226] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, 2017. 6.5

[227] Vladimir Vapnik and Rauf Izmailov. Learning using privileged information: similarity control and knowledge transfer. *Journal of machine learning research*, 16(2023-2049):2, 2015. 6.5

[228] Balakrishnan Varadarajan, George Toderici, Sudheendra Vijayanarasimhan, and Apostol Natsev. Efficient large scale video classification. *arXiv preprint arXiv:1505.06250*, 2015. 3.2.1, 3.2.2, 3.3.1, 3.2, 3.4, 3.6, 3.4

[229] Gül Varol, Ivan Laptev, Cordelia Schmid, and Andrew Zisserman. Synthetic humans for action recognition from unseen viewpoints. *arXiv preprint arXiv:1912.04070*, 2019. 6.1

[230] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 4.2.1

[231] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017. 1.1, 5.2.2, 9.2.1, 9.2.2

[232] Chengxin Wang, Shaofeng Cai, and Gary Tan. Graphtcn: Spatio-temporal interaction modeling for human trajectory prediction. In *WACV*, 2021. 1.5

[233] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 4.1

[234] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *CVPR*, 2011. 4.4

[235] Heng Wang, Du Tran, Lorenzo Torresani, and Matt Feiszli. Video modeling with correlation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 352–361, 2020. 4.2.1

[236] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *ECCV*, 2018. 4.4

[237] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 4.1, 4.2.5, 4.3.2, **??**

[238] Yunbo Wang, Lu Jiang, Ming-Hsuan Yang, Li-Jia Li, Mingsheng Long, and Li Fei-Fei. Eidetic 3d lstm: A model for video prediction and beyond. In *ICLR*, 2019. 5.2.1, 6.3.4

[239] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *ICIP*, 2017. 2.2.2

[240] Conghao Wong, Heng Li, Shiming Chen, Qinmu Peng, Xinge You, et al. Bgm: Building a dynamic guidance map without visual images for trajectory prediction. *arXiv preprint arXiv:2010.03897*, 2020. 1.5

[241] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015. 1.1, 7.3.1

[242] Yu Wu, Lu Jiang, and Yi Yang. Revisiting embodiedqa: A simple baseline and beyond. *arXiv preprint arXiv:1904.04166*, 2019. 5.5, 6.5

[243] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 3.3.1, 3.3.3, 3.5, 3.4

[244] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *CVPR*, 2019. 6.4.2, 6.1, 6.5

[245] Dan Xie, Tianmin Shu, Sinisa Todorovic, and Song-Chun Zhu. Learning and inferring "dark matter" and predicting human intents and trajectories in videos. *IEEE transactions on pattern analysis and machine intelligence*, 40(7):1639–1652, 2018. 7.1, 7.2.1, 7.4

[246] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *NeurIPS*, 2015. 5.1, 5.2.1, 5.4.1, 6.3.4

[247] Xuehan Xiong and Fernando De la Torre. Supervised descent method and its applications to face alignment. In *CVPR*, 2013. 4.4

[248] Huijuan Xu, Abir Das, and Kate Saenko. R-c3d: Region convolutional 3d network for temporal activity detection. In *ICCV*, 2017. 2.1, 4.4

[249] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015. 4.2.2

[250] Yanyu Xu, Zhixin Piao, and Shenghua Gao. Encoding crowd interaction with deep neural network for pedestrian trajectory prediction. In *CVPR*, 2018. 7.4

[251] Hao Xue, Du Q Huynh, and Mark Reynolds. Ss-lstm: A hierarchical lstm model for pedestrian trajectory prediction. In *WACV*, 2018. 5.5, 6.5

[252] Hao Xue, Du Q Huynh, and Mark Reynolds. A location-velocity-temporal attention lstm model for pedestrian trajectory prediction. *IEEE Access*, 8:44576–44589, 2020. 1.5

[253] Takuma Yagi, Karttikeya Mangalam, Ryo Yonetani, and Yoichi Sato. Future person localization in first-person videos. In *CVPR*, 2018. 1.1, 5.5, 6.1, 6.4.2, 6.5, 7.3.1, 7.3.1, 7.4

[254] Ceyuan Yang, Yinghao Xu, Jianping Shi, Bo Dai, and Bolei Zhou. Temporal pyramid network for action recognition. In *CVPR*, 2020. 4.4

[255] Xitong Yang, Xiaodong Yang, Ming-Yu Liu, Fanyi Xiao, Larry S Davis, and Jan Kautz. Step: Spatio-temporal progressive learning for video action detection. In *CVPR*, 2019. 4.4

[256] Shuai Yi, Hongsheng Li, and Xiaogang Wang. Pedestrian behavior understanding and prediction with deep neural networks. In *ECCV*, 2016. 7.4

[257] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving video database with scalable annotation tooling. *arXiv preprint arXiv:1805.04687*, 2018. 6.5

[258] Ye Yuan and Kris Kitani. Diverse trajectory forecasting with determinantal point processes. *ICLR*, 2020. 9.3.1, 10.2.4

[259] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012. 5.4.1, 6.4.2, 7.3.1, 9.3

[260] Xiaohui Zeng, Chenxi Liu, Yu-Siang Wang, Weichao Qiu, Lingxi Xie, Yu-Wing Tai, Chi-Keung Tang, and Alan L Yuille. Adversarial attacks beyond the image space. In *CVPR*, 2019. 6.5

[261] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 6.1, 6.3.3, 6.3.3, 6.4.5, 6.5

[262] Pu Zhang, Wanli Ouyang, Pengfei Zhang, Jianru Xue, and Nanning Zheng. Sr-lstm: State refinement for lstm towards pedestrian trajectory prediction. In *CVPR*, 2019. 5.3, 5.5, 6.5

[263] Yi Zhang, Xinyue Wei, Weichao Qiu, Zihao Xiao, Gregory D Hager, and Alan Yuille. Rsa: Randomized simulation as augmentation for robust human action recognition. *arXiv preprint arXiv:1912.01180*, 2019. 6.1, 6.3.2, 6.5

[264] Yiwei Zhang, Graham M Gibson, Rebecca Hay, Richard W Bowman, Miles J Padgett, and Matthew P Edgar. A fast 3d reconstruction system with a low-cost camera accessory. *Scientific reports*, 5:10909, 2015. 5.5, 6.5

[265] Yubo Zhang, Pavel Tokmakov, Martial Hebert, and Cordelia Schmid. A structured model for action detection. In *CVPR*, 2019. 4.4

[266] Tianyang Zhao, Yifei Xu, Mathew Monfort, Wongun Choi, Chris Baker, Yibiao Zhao, Yizhou Wang, and Ying Nian Wu. Multi-agent tensor fusion for contextual trajectory prediction. In *CVPR*, 2019. 5.3, 5.5, 6.5

[267] Yue Zhao, Yuanjun Xiong, and Dahua Lin. Trajectory convolution for action recognition. In *NeurIPS*, 2018. 4.4

[268] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *ICCV*, 2015. 8.2.1

[269] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017. 5.2.1, 5.4.1, 6.3.2

[270] Kaiyang Zhou, Yongxin Yang, Andrea Cavallaro, and Tao Xiang. Omni-scale feature learning for person re-identification. In *ICCV*, 2019. 8.2.1

[271] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, 2017. 5.5, 6.5

[272] Haosheng Zou, Hang Su, Shihong Song, and Jun Zhu. Understanding human behaviors in crowds by imitating the decision-making process. *arXiv preprint arXiv:1801.08391*, 2018. 7.4