

Video Understanding: A Predictive Analytics Perspective

He Zhao

A Dissertation Submitted to the Faculty of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

Graduate Program in
Electrical Engineering and Computer Science
York University
Toronto, Ontario, Canada

September 2022

©He Zhao 2022

Abstract

This dissertation includes a detailed study of video predictive understanding, an emerging perspective on video-based computer vision research. This direction explores machine vision techniques to fill in missing spatiotemporal information in videos (e.g., predict the future), which is of great importance for understanding real world dynamics and benefits many applications. We investigate this direction with depth and breadth. Four emerging areas are considered and improved by our efforts: early action recognition, future activity prediction, trajectory prediction and procedure planning. For each, our research presents innovative solutions based on machine learning techniques (deep learning in particular) and meanwhile pays special attention to their interpretability, multi-modality and efficiency, which we consider as critical for next-generation Artificial Intelligence (AI). Finally, we conclude this dissertation by discussing current shortcomings as well as future directions.

*To my family,
for your love and support.
And, to myself,
for never giving up.*

Acknowledgments

First of all, I would like to express my deepest gratitude to my supervisor, Dr. Richard P. Wildes, for opening the door of the academic world for me. This amazing Ph.D. journey would not have been possible if he had said “No” when I was knocking on the door and asking to join the computer vision lab. And his impact on me goes way beyond that. It is his endless support, valuable guidance, great confidence and encouragement in me throughout the study that take me this far. He not only provided high-standard academic training for me, but also supported my growth as a person too. Of all the enjoyable moments in four years as a Ph.D. student, the one that I treasure most is when Rick said, and I quote, “***You have become an experienced writer***” when we were smithing a manuscript. A moment that I found my target fulfilled (i.e., becoming a better researcher) and started to believe my ambition is achievable. For everything, I am forever in his debt.

I was fortunate to have an amazing academic advisory committee, Dr. Michael S. Brown and Dr. Konstantinos G. Derpanis. I would like to thank them for treating me like one of their own students, being responsible, and providing great advice for my research career.

I would like to thank my examination committee members, Dr. Robert Allison, Dr. Patrick Cavanagh and Dr. Leonid Sigal, for their time and valuable feedback on this dissertation.

For my dear lab mates and friends, past and present, I would like to thank you all for the meaningful discussions when research work got stuck, the invaluable support when it comes to paper writing as well as presentation practicing, and the heartwarming comforts when there were pressure and hardship in life.

Finally, I would like to dedicate this work to my parents and wife, 赵恒玉（父），王存香（母），以及李莹琨（妻），for respecting my decision in pursuing the doctorate degree

and standing firmly by my side, and to my son, 趙胤吉吉（子）, for coming to our family safely in the final year of my study. They are the reason that I take on challenges and persist to the end!

Contents

Abstract	ii
Dedication	iii
Acknowledgements	iv
Table of Contents	vi
List of Figures	x
List of Tables	xii
1 Introduction	1
1.1 Motivation	1
1.2 Contributions	4
1.3 Dissertation Outline	6
2 Interpretable early action recognition	7
2.1 Motivation	7
2.2 Related work	9
2.2.1 Direct inference	9
2.2.2 Knowledge distillation	11
2.2.3 Feature propagation	13
2.3 Overall contribution: Learning to extrapolate features	13
2.3.1 General pipeline	14
2.3.2 Feature residuals	16
2.3.3 Residual generator network using dynamic filter	16
2.3.4 Revisiting Kalman filtering	18
2.3.5 Connection of Kalman filtering to RNNs	22
2.3.6 Unified model for all observations ratios	23
2.4 Empirical evaluation	23
2.4.1 Evaluation protocol	23
2.4.2 Implementation details	24

2.4.3	Comparisons to alternative approaches	26
2.4.4	Ablation studies	29
2.5	Interpretability	30
2.5.1	Intermediate layer features	31
2.5.2	Learned motion kernels	33
2.5.3	Learned Kalman gain	41
2.6	Conclusion	45
3	Future activity prediction	47
3.1	Motivation	47
3.2	Related work	48
3.2.1	Activity singleton or sequence prediction	49
3.2.2	Joint prediction of activity and occurrence time	51
3.2.3	Egocentric action prediction	55
3.3	Overall contribution: Diversified discrete GAN for joint activity and time prediction	58
3.3.1	General pipeline	58
3.3.2	Gumbel-Softmax for differentiable discrete variable sampling	59
3.3.3	Mode collapse in discrete GANs	61
3.3.4	Dual token embedding	63
3.3.5	Sequence generation	65
3.3.6	Diversified discrete adversarial learning	67
3.3.7	Ensembled discriminator	68
3.4	Empirical evaluation	69
3.4.1	Evaluation protocol	69
3.4.2	Implementation details	70
3.4.3	Comparisons to alternative approaches	71
3.4.4	Ablation studies	75
3.4.5	Visualization of diversity and quality	77
3.5	Conclusion	80
4	Video trajectory prediction	81
4.1	Motivation	81
4.2	Related work	82
4.2.1	Collective dynamic modelling on pedestrians trajectories	82
4.2.2	Modelling physical constraints	82
4.2.3	Future distribution modelling	83

4.2.4	Goal conditioned prediction	83
4.2.5	Expert learning	83
4.3	Overall contribution: Dynamic trajectory prediction with expert goal examples	84
4.3.1	Problem formulation	84
4.3.2	Goal retrieval via dynamic time warping	85
4.3.3	Goal conditioned trajectory predictor	88
4.3.4	Learning scheme	91
4.4	Empirical evaluation	91
4.4.1	Evaluation protocol	91
4.4.2	Implementation details	92
4.4.3	Comparison to alternative approaches	93
4.4.4	Feasibility of expert examples	95
4.4.5	Ablation studies	96
4.4.6	Goal and prediction visualization	99
4.5	Conclusion	102
5	Weakly supervised procedure planning in instructional videos	103
5.1	Motivation	103
5.2	Related work	105
5.2.1	Procedure planning	105
5.2.2	Supervision with natural language	106
5.2.3	Sequence modeling with transformers	106
5.2.4	Future prediction	106
5.3	Overall contribution: Probabilistic procedure planning via weak natural lan- guage supervision	107
5.3.1	Problem formulation	107
5.3.2	Memory augmented transformer decoder	108
5.3.3	Adversarial generative modeling	110
5.3.4	Training	111
5.3.5	Inference	113
5.4	Empirical evaluation	114
5.4.1	Implementation details	114
5.4.2	Evaluation protocol	115
5.4.3	Comparison to alternative approaches	117
5.4.4	Evaluation of probabilistic modeling	119

5.4.5	Ablation studies	122
5.4.6	Plan visualizations	128
5.5	Conclusion	128
6	Conclusion and future work	130
6.1	Summary and overall discussion	130
6.2	Societal impact	132
6.3	Future work	134
	Bibliography	138

List of Figures

1.1	Taxonomy of video understanding.	2
1.2	Illustration of three types of video predictive understanding.	3
2.1	Representative applications of early action recognition.	8
2.2	Summary of related work on early action recognition.	9
2.3	Overview of Proposed Feature Residual Propagation Approach to Action Prediction.	15
2.4	Temporal Extrapolation Residual Generator Network (RGN) for Predicting Next Time Step Residual.	17
2.5	Depiction of Kalman update procedure.	19
2.6	Action Prediction Results on the UCF101 and BIT Datasets.	28
2.7	Action Prediction Results on the JHMDB-21 Datasets.	29
2.8	Visualization of Features and Residuals.	32
2.9	Sparsity Comparison of Features and Residuals.	33
2.10	Visualization of Learned Motion Kernels.	34
2.11	Illustration of spatial shifting via convolution using 2D unit impulse kernels.	35
2.12	Discretization of flow magnitude for conversion to motion kernel format.	35
2.13	Pipeline for computing the match between learned motion kernels and optical flow kernels.	37
2.14	Box plots of correlation between learned and optical flow kernels.	38
2.15	Visualization of correlation on three distinctive feature channels.	39
2.16	Plots of kernels for videos that have larger motion magnitudes.	40
2.17	Visualization of failure modes on various learned kernels.	41
2.18	Visual examples for action videos.	42
2.19	Illustration of learned Kalman gain values on the Baseball swing video.	43
2.20	Visualization of learned Kalman gain for dart and handclapping videos.	44

2.21	Visualization of changed early action recognition accuracies.	45
3.1	Dissimilarity between future activity prediction and early action recognition. .	48
3.2	Tree graph action prediction.	50
3.3	Predicting future activity labels and time durations.	52
3.4	Poisson distribution.	53
3.5	Multi-task model for joint prediction of action and time.	55
3.6	Examples of egocentric activity footage.	55
3.7	Proposed Activity Anticipation Process Overview.	59
3.8	Impact of the temperature hyper-parameter on Gumbel-Softmax function. . .	60
3.9	Simplified demonstration of the quality vs diversity trade-off.	62
3.10	Illustration of the dual token embedding process.	63
3.11	Depiction of our diversity improved discrete sequential GAN architecture. . .	64
3.12	Comparison table for the Epic-Kitchen dataset.	75
3.13	Visualization of multimodal samples.	77
3.14	Visualization of multimodal samples.	78
3.15	Visualization of multimodal samples.	79
4.1	Overview of the Query-Retrieval based framework.	85
4.2	Pipeline of our goal-retrieval based trajectory prediction algorithm.	87
4.3	Illustration of goal retrieval quality by plotting the retrieval error.	88
4.4	Illustration of the feasibility quality of our results on the SDD Hyang4 scene.	95
4.5	Plotting of evaluation results from the presented approach.	101
5.1	Illustration of weak language supervision for procedure planning.	104
5.2	Overview of our pipeline for procedure planing.	108
5.3	Illustration of the transformer block with self-attention and cross-attention with memory.	110
5.4	Sample <i>plausible</i> plans 1.	127
5.5	Sample <i>plausible</i> plans 2.	127
5.6	Sample <i>plausible</i> plans 3.	128

List of Tables

2.1	Action Prediction Results on JHMDB21.	30
2.2	Accuracy results for different temporal propagation approaches on JHMDB21.	31
2.3	Prediction Accuracy (%) at Various Intermediate Feature Stages.	32
3.1	Breakfast Dataset Protocol 1 results with dense anticipation mean over classes (MoC) accuracy.	72
3.2	Breakfast Dataset Protocol 2 results.	73
3.3	Results for the 50Salads dataset.	74
3.4	Influence of various number of samples for Breakfast Dataset.	76
3.5	Comparison across various temperatures as well as impact of the normalized distance regularizer.	76
4.1	Evaluation results on the ETH and UCY datasets.	93
4.2	Evaluation results on the SDD dataset for the next 12 timesteps trajectory prediction.	95
4.3	Ablation studies for accuracy and search speed.	97
4.4	Ablation results on our predictor w/o goal.	98
4.5	Ablation study on number of goal retrievals and samples.	98
4.6	Goal retrieval quality versus various number of repository search candidates.	99
4.7	Ablation study on few-shot goal retrieval.	99
5.1	Evaluation of procedure planning results on CrossTask.	117
5.2	Success Rate evaluation results of long-term performance CrossTask.	119
5.3	Procedure planning results on NIV and COIN.	119
5.4	Evaluation of the plan distributions produced by our probabilistic approach versus the deterministic variant on CrossTask.	120
5.5	Evaluation of diversity and accuracy of our planned procedures.	120

5.6	Evaluation of the plan distributions on COIN.	121
5.7	Evaluation of the plan distributions on NIV.	121
5.8	The average number of unique paths across three datasets.	122
5.9	Performance of our model trained with different loss functions on three datasets.	123
5.10	Ablation study on the impact of external memory sizes for prediction horizon $T = 3$	124
5.11	The effect of diversity regularization loss on the produced plans.	124
5.12	The effect of estimating transition matrix and emission matrix of <i>Viterbi</i> post- processing.	125
5.13	Ablation study of the number of samples for procedure planning.	125
5.14	Ablation study on the number of layers and heads for prediction horizon $T = 3$.	126
5.15	Alternative uncertainty baselines.	126

Chapter 1

Introduction

1.1 Motivation

This dissertation examines a topic that has received increased attention in recent years, **video understanding**. The fact that spatiotemporal video input data contains richer information than its static, single image counterpart (e.g., zero motion) makes it appealing. Many confusing problems in a static world become much easier to handle in videos, e.g., the difference between *sitting down on* and *standing up from* the same chair becomes much clearer, as the upward and downward motion stand in stark contrast to each other. Video data provides a fresh visual outlook on how events might develop as time goes on, and it is a more revealing information source, considering that our world is highly dynamic and never stops changing. Video understanding research largely can be categorized, into three subareas, namely recognition, motion analysis and predictive modeling, as outlined in Figure 1.1. The focus of this dissertation is on prediction; however, to set the stage, the next three paragraphs briefly introduces each of the three sub-areas in turn.

To help machine vision gain insightful understanding on world dynamics, great effort has been spent on video-based recognition. Some representative tasks include: *Action recognition* [48, 47, 237], *dynamic texture recognition* [75, 76, 38], *activity boundary segmentation* (localization) [196] and *event detection* [46]. The former two target labeling the categories of actions or dynamic textures across entire videos, without localization, whereas the latter two focus on estimating and/or grounding frame-wise recognition. Indeed, recognition is

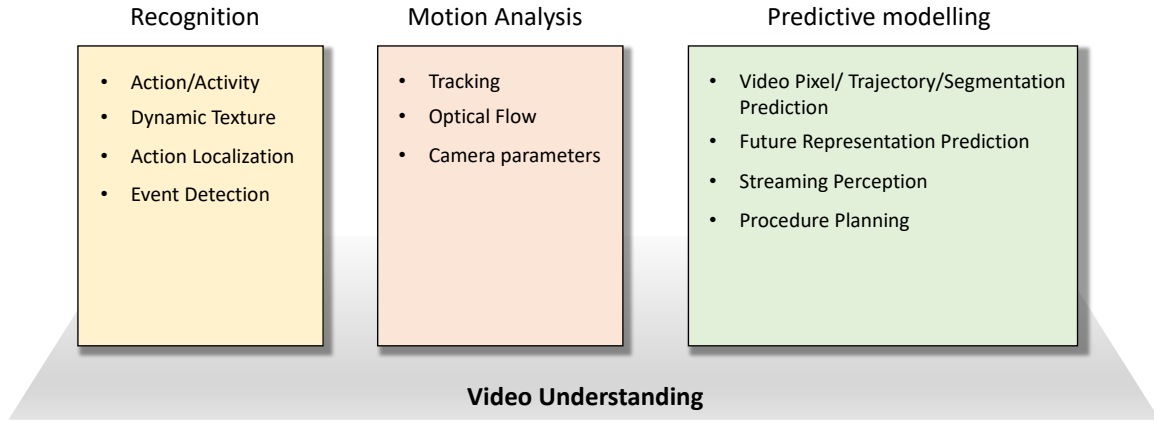


Figure 1.1: Taxonomy of video understanding.

fundamental in interpreting a viewed scene and studying these tasks often leads to a good representation of information captured in a video.

Another perspective views video data as a collection of moving objects, e.g., walking pedestrians, running athletes and driving vehicles, where locomotions are captured by pixel displacements in image sequences. Thereby, motion analysis is of great interest, including consideration of both certain visual regions (e.g., objects) as well as intrinsic parameters of motion-capturing devices (e.g., camera egomotion estimation [270]). Related vision tasks include visual *odometry* estimation [157], *tracking* [255, 19] and *optical flow* estimation [218]. The outputs from such analyses are found valuable to other areas, e.g., modern action recognition work has observed huge success from taking optical flow and/or trajectories as components of their input (e.g., in addition to raw frames) [98].

Different from the approaches just outlined, in this dissertation, we approach video understanding from a predictive perspective, which studies inferring unseen information from some observed videos. Videos often have smooth flow of information and are semantically consistent. For example: Soccer plays may proceed by player running, kicking the ball and scoring a goal; it is neither acceptable to perform the actions in the opposite order nor to skip the middle step. This temporal coherence is an inherent attribute of typical video activity events and could be of great value in its own interpretation, e.g., deducing missing parts of videos from the extant.

Modelling the predictive nature of videos can bring substantial understanding of the viewed world. As an example, an automatic action prediction model mounted on surveillance cameras

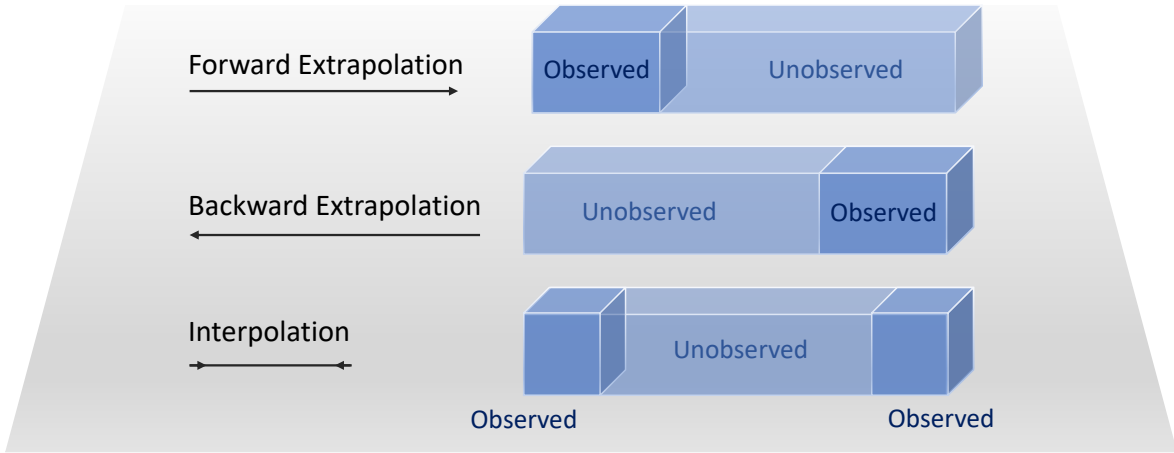


Figure 1.2: Illustration of three types of video predictive understanding.

can help foresee the next step of pedestrians in traffic intersections – a technique that is much needed for safe travelling and anomaly detection. Conversely, humans also can deduce the beginning part of video content after first observing the end, such as hypothesizing a player driving past a defender given a dunk on a basketball court. Moreover, given the start and end, planning the intermediate content is typical in our lives, e.g., conceptualizing a plan to make a pepperoni pizza (end) out of a stack of ingredients (start). These three examples correspond to three types of temporal deduction, i.e., **forward**-extrapolation, **backward**-extrapolation and interpolation; see Figure 1.2. To equal or exceed human visual abilities, artificial intelligence (AI) research needs to study them in video-based computer vision.

In recent academic research, understanding videos from the predictive modelling perspective has been investigated along several lines. Some representative examples include: (1) video pixel prediction that inputs the past image observations and outputs the unobserved future pixels (e.g., images) [125, 251], (2) video segmentation prediction that produces scene segmentation of the next few unseen seconds [148], (3) future visual representation/semantic prediction, which can be used for future action/activity anticipations or self-supervised representation learning [232, 205, 266], (4) streaming perception that concerns reducing latency time of online video processing algorithms (i.e., video-based online object detection for autonomous vehicles) [134], (5) human pedestrian trajectory (e.g., 2D coordinates) or motion (e.g., body key-points) future prediction [72, 179, 88, 198, 268] and (6) procedure planning in instructional videos that studies how to plan multi-step solutions for daily tasks via watching videos [25, 13, 265]. All these efforts have applied either the extrapolation or the interpola-

tion idea within their domains, and they all ask the same core question: How can one deduce a part of video (mostly future parts) from other extant parts?

1.2 Contributions

In this dissertation, we investigate video predictive understanding with a **depth** and **breadth**. Our study considers the following four emerging areas: early action recognition, future activity prediction, video trajectory prediction and procedural planning in instructional videos. For each, we have made novel technical contributions and improvements on performance compared to the previous state of the art.

1. Early action recognition, which studies recognizing actions using only limited frames from the beginning of videos, is an exemplar application of the aforementioned future visual representation prediction (i.e., example (3) in Section 1.1). For this task, we investigate classical sequential modelling techniques for visual feature propagation and present a learnable Kalman filtering model with the following three key characteristics. First, it makes use of video feature residuals across time, which allows for a compact representation that reduces redundancy while retaining essential information. Second, it extends a recurrent neural network (RNN) with adaptive schedule sampling (Kalman gain) to extrapolate the feature residuals forward in image spacetime, which effectively assimilates new observations and is more stable in training than a standard RNN. Third, our system admits to detailed interpretability, instead of being a black-box. We demonstrate the superiority of our model compared to alternatives by achieving state-of-the-art accuracy on three public benchmarks as well as a unified model for various prediction horizons. Our approach to early action recognition has appeared previously [266]. Our code and data are publicly available at <https://github.com/JoeHEZHAO/Spatiotemporal-Residual-Propagation>.

2. Future activity prediction in long-term videos focuses on predicting not just the current action at the earliest possible time, but rather on predicting a subsequent sequence of actions from initial observations. This research direction can be categorized as future semantic prediction (i.e., example (3) in Section 1.1). Here, our work concentrates on modelling uncertainty. Existing models lack diversity in their predictions: Indeed, many are designed to be deterministic. In the real world, however, the same initial actions can very possibly lead

to diverse outcomes. Take kitchen cooking scenarios as an example: Many things can follow *washing vegetables*, such as *cutting vegetables*, *put vegetables to pot*, *etc.* The difficulty for learning resides in the fact that there typically are no multiple outcome labels intentionally annotated for this purpose; therefore, each iteration of parameter update (e.g., stochastic gradient descent [66]) will only push the model toward one ground truth. In response to this challenge, we adopt the recent advance of deep generative networks that can encode uncertainty with random latent codes. Furthermore, we observe that previous approaches have difficulty capturing diverse modes, i.e., models tend to ignore the variants in the latent code and downgrade to a deterministic model. We solve this matter with an efficient latent code regularizer that promotes diversity in a non-trivial fashion. Again, we demonstrate high quality results over multiple datasets. Our approach to future activity prediction has appeared previously [268]. Our code and data are publicly available at <https://github.com/JoeHEZHAO/DAA>.

3. For the video trajectory prediction work, where researchers are interested in forecasting the trajectories of human pedestrians in future frames (i.e., example (5) in Section 1.1), we switch to the study of predictive modelling from the interpolation perspective, with a special attention on data efficiency. Rather than directly predicting future pedestrian trajectories, our work targets bridging the initially observed trajectories and the final goal positions, which essentially is interpolation. The key question is how to obtain such goals. A non-parametric goal searching method is presented in our work, which requires zero learning and no extra parameters. The entire search procedure can be done through the training set, which needs no extra efforts on data augmentation and can run with GPU acceleration for real-time inference. New state-of-the-art results on video-based pedestrian trajectory predication are witnessed from our approach. Our approach to video trajectory prediction has appeared previously [267]. Our code and data are publicly available at https://github.com/JoeHEZHAO/expert_traj.

4. Finally, for the procedure planning task, we also study interpolation but with instructional videos. This task tackles filling in the missing transitional actions given the start and goal observations (i.e., example (6) in Section 1.1). The question we answer is how to make a plan containing several procedural steps connect start and end image observations. Here, the instructional videos can vary widely, e.g., across cooking, carpentry and car repair. There are three parts to this problem. The first is finding an expressive sequential model for procedure inpainting and the second is learning efficiency. Instructional videos are minutes long, and thus providing detailed annotations for each training sequence is human labor intensive.

Yet, model training needs good annotations. We tackle the first problem using transformers, a “one-shot” attention model that runs in parallel for every time step. Especially, we augment the vanilla transformer with external memory slots shared across all layers for better accuracy. For the second problem, we reduce the amount of visual annotations using natural language supervision. The label descriptions coming freely with many datasets can be viewed as language information and can provide useful supervision. The third problem is uncertainty modelling. Similar to future activity prediction, there typically are several different procedures that can bridge initial and goal observations. For example, given the initial observation of some produce items (e.g., lettuce, tomatoes, etc.) and the final/goal observation of a salad, there are many ways in which one might slice, dice and otherwise combine the ingredients. We use the same technique adopted in future activity prediction, i.e., generative modeling with diversity regularization, to handle this matter. Once again, empirical results demonstrate the superiority of our approach to currently available alternatives. Our approach to procedural planning has appeared previously [265]. Our code and data are publicly available at <https://github.com/SamsungLabs/procedure-planning>.

1.3 Dissertation Outline

We organize the content of this dissertation in the following fashion. This first chapter has served to place video prediction in the larger context of video understanding as well as to introduce our major contributions. Subsequent chapters will present our contributions in detail, in the same order as introduced above: Chapter 2 presents early action recognition; Chapter 3 presents future activity prediction; Chapter 4 presents trajectory prediction; Chapter 5 presents procedure planning. In each chapter, we introduce motivations and problem definition, review extant approaches (including discussions of pros and cons), introduce our novel contributions and finally show results of empirical evaluation. In the final chapter, we summarize our technical findings, societal implications and provide outlines of several future directions.

Chapter 2

Interpretable early action recognition

2.1 Motivation

Action recognition has received much research effort in the past decade [98, 108]. A recent trend has been to focus on the ability to recognize an action as early as possible, i.e., before the execution has entirely finished [111, 192]. In this dissertation, this approach is called **early action recognition**. Why is this problem considered important? The reason can be easily seen when having video action recognition models as real-world deployable systems: Full action executions take elongated time spans and running the recognition algorithm after the action is completed would naturally lose the capability to (pre)react in real-time, which, however, in many conditions is essential. Figure 2.1 provides three applicable examples. One working scenario is using recognition models on edge devices that suffer from limited computational resources, such as CPU-only smart phones. Early recognition equals to using only small amounts of initial frames, which is much favored. Also, it can be a valuable asset to interactive environments, such as human-robot interactions and self-driving cars, where **fast response** is needed for real-time processing. Another applicable situation is traffic scenes, where harmful behaviors, e.g., running across the red light, needs to be recognized as early as possible.

Problem definition. Following early work on this topic (e.g., [111]), the task is formally defined and evaluated in terms of *observation ratio*, i.e., the ratio of the observed frames on which recognition is based to the total number of frames in a video of interest, with the goal being recognition at smaller ratios. More precisely, a full video, $y_{1:T}$, is divided uniformly

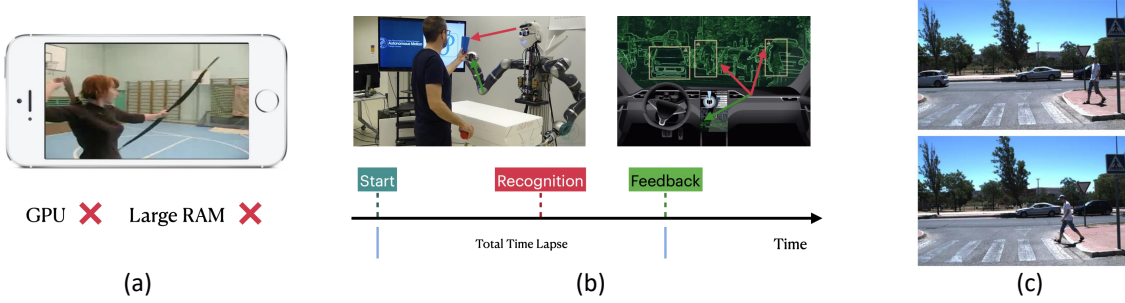


Figure 2.1: Representative applications of early action recognition. Sub-figures (a)-(b) depict the use of early action recognition on reducing the computational budget of portable devices (e.g., mobile phones typically lack GPU and memory space to process elongated videos) and on diminishing the response gap for fast feedback systems (e.g., human-robot interactions and self-driving vehicles). Sub-figure (c) illustrates the application for early recognition of pedestrian behaviors at traffic intersections.

into K segments

$$y_{1:T} = \sum_k^K y\left[\frac{T}{K}(k-1) + 1, \frac{T}{K}(k)\right], \quad (2.1)$$

where $k = 1, \dots, K$ is the index of the segment, K is the number of segments and T is the number of video frames. The length of each segment is T/K and a partial observed video sequence can be described as $y[1, \frac{T}{K}(k)]$. Following existing work, the number of segments, K , is set to 10 and the observation ratio is defined as the ratio of k to K , i.e., $\rho = \frac{k}{K}, \in [0.1, \dots, 1]$ and performance is evaluated on recognition accuracy as a function of ρ .

This direction of research holds a basic assumption: The full action sequence supports superior recognition compared to observation of only an initial segment; otherwise, the problem is uninteresting. This assumption was validated by the original definer of the problem [192]. Experiments were conducted that evaluated recognition algorithms on various partial action clip segments (i.e., $y_{1:t}, t \in (1, \dots, T)$) and demonstrated that insufficient observations in action temporal scope resulted in much inferior recognition accuracy. Nevertheless, results were promising enough to motivate pursuit of early action recognition as a research topic.

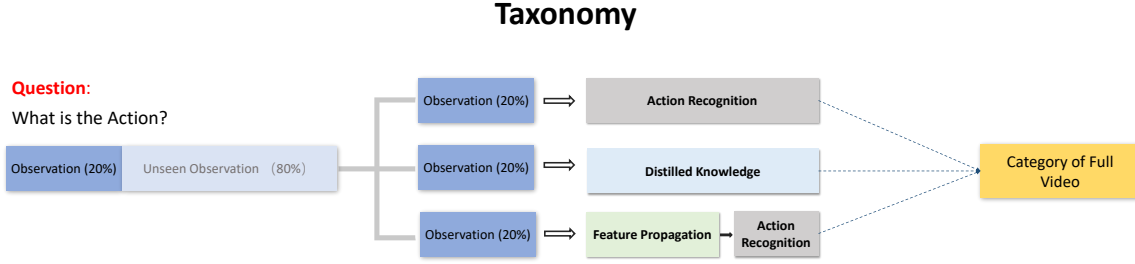


Figure 2.2: Summary of related work on early action recognition. Existing approaches can be grouped into three sub-categories. They are direct inference at the top row, knowledge distillation at the middle row and feature propagation at the bottom row.

2.2 Related work

In the last decade, researchers have approached the task of early action recognition from various perspectives and we group them into three major divisions (Figure 2.2): (1) Directly infer the the ground-truth labels of full observations from partial observations; (2) distilling knowledge from full observations into partial observations prior to making inferences; (3) propagating the partial observations into the future along the temporal axis and then making inferences on both the initial and extrapolated information. In the following sections, we review all recent research according to these divisions.

2.2.1 Direct inference

Early on, researchers believed that a partial observation of an action video provides sufficient information in and of itself to define the appropriate overall action class regardless of the unobserved clip portions. What appears to be the first work [192] made use of sequential **feature matching** with Integral Bag-of-Word (IBoW) representations [207]. The intuition being that the feature distributions of partial and complete action clips should be statistically similar. They initially extract spatiotemporal Interest Point features (STIP [123]) from training videos, re-organize them into visual word histograms (IBoW) and use them as matching templates. After that, a Gaussian kernel is used to generate a probability score representing the likelihood between partial and whole observations. Finally, the selected class label is that of the full action that produces the largest likelihood. Similar work improved the feature representation as

well as selection with sparse coding but shared the overall structure [20].

While the IBoW approach supports fast inference and is learning free, it operates under two restrictive assumptions. First, the feature extractor, STIP, yields an appearance based feature descriptor (indeed STIP comes from spacetime Harris-corner detection [80]), which is sensitive to view changes, illumination and other nuisance details of image acquisition. Thus, targets from the same action category are assumed to be similar in appearance. Second, the scheme to matching feature histograms between partial and full actions is not always practical since actions can evolve dramatically, to the extent that the beginning appears totally unlike the ending.

Subsequent work following essentially the same framework improved on the original by employing stronger features as well as more sophisticated machine learning-based classifiers [122]. As an example, the feature representation was enhanced by merging motion descriptors (iDT) with appearance descriptors (HOG) [33, 235]. Others focused on extracting features from hierarchical temporal scales and assign labels using Support Vector Machines (SVMs) [29] that were trained on features collected across the hierarchy to make the decision jointly [122].

Some contemporary work tried using fine-grained information (e.g., actor regions), rather than the whole scene, for this task. They thought that actions can only associate with behaviors of human actors (i.e., bounding box coordinates of detected actors). One such effort made use of object/human detectors pretrained on extensive datasets (e.g., SSD [144]). They proposed to jointly perform action localization as well as early action recognition [206]. Furthermore, some work felt human active regions are still very coarse; they suggested using human body structure (i.e., key-points) and selecting the salient body features for early action recognition, with a reinforcement learning policy function [164] for performing the selection [27]. Along similar lines, research using 3D body skeleton analysis for early action recognition has focused on adaptively selecting the best historical information window for predictions in a data-driven fashion [141]. Besides actor-centric methods, a recent effort has instead considered object-centric human actions (e.g., fetching objects or moving something to somewhere) [271]

Relational modelling is proposed very recently for this task. Some work exploited spatial relations of elements within a single frame [217]. The relation between multiple actors is modeled as nodes and edges in a graph [200] and a Graph Neural Network (GNN) [78] is adopted

to diffuse information among nodes. Following this direction, a multiscale spatiotemporal graph based method further explores fine-grained human-to-human and human-to-object interactions for this task [245].

Finally, some other work considers early action prediction as a special case of action recognition in that only a few action clips that are easily distinguishable under full observations become difficult when incomplete. To deal with such difficult cases, they equipped the regular recognition framework with an extra memory module to log hard examples explicitly [109].

2.2.2 Knowledge distillation

Methods in the previous section mainly focus on reasoning about the future directly from the past. The unobserved information remains negligible, except the overall action label being used for training. Here, it is worth remembering that previous work has shown that full observation almost always excels over the partial and no one has forbidden the use of unobserved information in training [192]. The more urgent question is *how*. Hereby, we discuss research that attempts to lend power from unobserved data in training to either enrich the feature representation of partial data or encourage the classifiers to recognize partial data with ease.

Initial work trained video segment classifiers that can align the temporal order of partial observations to training data [111, 107]. They achieved this result by using a Structured SVM (SSVM) [226]. More specifically, they evenly divided action videos into equal-length segments, extracted features from each segment, integrated segments into one chunk, such as $y_{1:k}$, and trained SSVM classifiers according to the following constraints (without loss of generality, two progress levels $(k-1, k)$ are given):

$$\alpha_{k-1}^T \psi_1(y_{1:k-1}, x) \leq \alpha_k^T \psi_1(y_{1:k}, x), \quad (2.2)$$

where $y_{1:k}$ is the video segment feature accumulated from the very beginning till the k_{th} level, x is the ground-truth label, α_k^T is the score matrix for progress level k and $\psi_1(y_{1:k}, x)$ is a joint feature map that represents the spatiotemporal features of action label, x , given a partial video $y_{1:k}$. The authors assumed 10 progress levels, such that $k \in (1, 10)$.

The above formulation emphasized temporal label consistency: Besides assigning a cor-

rect class label to given input videos, classifiers also need to capture the temporal evolutionary order by monotonically increasing the score for segments across time (also seen in [85]). The intuition coincides with daily human experience: As we see more, we should know better than before. Some other work following a similar intuition used ranking loss [150], global human intentions [193], temporal saliency [121] and poselet [178]. Finally, an effort focused on refining the classification confidence score at each time-step in an incremental manner [87]. That work relied on an attention mechanism to provoke high confidence on early observations.

Besides SSVM based methods, another approach was proposed to infer complete future features from partial observations [112]. Their work supposes that we have features of partial observation, $g(y_{1:t})$, and full observation, $g(y_{1:T})$ ($t \in (1, \dots, T)$), in the training stage, and proposed to reconstruct the full from partial according to:

$$\operatorname{argmin}_{W, \theta} \|g(y_{1:T}, \theta) - Wg(y_{1:t}, \theta)\|^2, \quad (2.3)$$

where W is the feature transformation matrix and $g(y, \theta)$ is the feature extraction function with parameter θ . Then state-of-the-art 3D CNN video features (i.e., C3D [225]) were employed for g and the transformation matrix, W , was learned through Stochastic Gradient Descent [66].

A more recent upgrade on this work [113] converts the deterministic feature transformation process to a conditional generative process, with an adversarial loss [67] to promote realism. Similarly, parallel work suggested transforming features into a low dimension binary code, as seen in Hashing encoding theory [18], and operating the feature matching on the binary encoded counterpart.

Very recently, neural network based knowledge distillation [84] has been applied to transfer fully-observed information into partial observations [238]. Two neural networks were employed: One acts as a teacher that gets access to full observations; the second acts as a student with only partial access. The student’s representation is driven toward that of the teacher by minimizing the Mean Square Error (MSE) between the latent features and the Maximum Mean Discrepancy (MMD) between the predicted classification probabilities. Some follow on efforts emphasized both the feature reconstruction consistency and semantic knowledge consistency via integrating multiple loss functions (e.g., feature distance loss, adversarial loss and classification loss) for a joint optimization [252, 234]. Indeed, it seems that the distance metric plays a critical role in building the connection between video parts. Further, one recent

work has found significant improvements via adopting the Jaccard similarity metric to learn the transition between partial and full observations [50].

2.2.3 Feature propagation

Another way to exploit future information is by propagating the partial observation into the future. Typically, this type of approach first extrapolates the visual representations along the time-axis and then applies an existing classifier. Mathematically, propagation based approaches start from any incomplete input $y_{1:t}$, assuming $t < T$, and produce the subsequent information $(y_{t+1}, y_{t+2}, \dots, y_T)$; thus, they can be considered as temporal extrapolation.

An early exemplar of the propagation approach operated by working with relatively high-level visual representations so that they can be readily processed by extant recognition tools [233]. In practice, they formulated a deep regression network, consisting of multiple convolutional and fully connected layers, to temporally regress the future high-level deep feature (i.e., fc7 activation output from AlexNet [201]) to improve the nonlinearity of predictions [205]. Yet other work upgraded the deep regression component with a Markov Decision Process (MDP) as well as an RNNs [261], but shared the same intuition. Other work combined the idea of feature propagation with a reinforcement reward function to encourage early recognition [60]. Yet other work propagated both the RGB and motion (i.e., optical flow) features from deep networks and adopted adversarial learning to boost the realism of their outputs [59]. Our work presented in the following section is closely related to this category.

2.3 Overall contribution: Learning to extrapolate features

With the background introduced in the previous section, here we present our research contribution on early action recognition: A deep learning based Kalman filtering network that propagates current states to the future. Our study provides solutions to the following technical difficulties: (1) compound error, where errors from first few predictions are carried along to later stages, (2) definition of good features to propagate, given many feature representations available, and (3) interpreting what has been learned in the devised Kalman filtering deep networks which are trained for extrapolating features. We will introduce the overall pipeline and

elaborate how our work addresses these challenges in the sections below.

2.3.1 General pipeline

When consider propagating an observed video sample into the future, we revisit classic techniques of sequential modelling (e.g., Kalman filter in state space models) as well as signal processing (e.g., video residual compression). Especially, we study applying models on deep feature representations. That means, rather than predict future frames per se, we instead predict intermediate layer features of a ConvNet trained for action recognition. Features at such layers are of high dimensionality, and thus enjoy a level of abstraction from the raw data that focuses on action relevant components, even while preserving spatial layout to capture relations between action components as well as scene context. For illustrative purposes, we use the Temporal Segment Network (TSN) [237] for initial feature extraction and final classification, because of its generally strong performance on action recognition [237]. While we use the TSN features and classifier, our prediction framework does not rely on the specifics of that approach and therefore should be more widely applicable to action prediction.

We decouple the prediction process into two steps: (1) feature residual propagation and (2) reconstruction. Residual information can play an important role in processing redundant data even while capturing subtle differences. MPEG-4 compression is a representative example of such processing [223], as is more general coarse-to-fine motion estimation (e.g., [7]). Moreover, recent work that exploits residual processing has considered optical-flow estimation [177], image denoising [93], video artifact removal [146] and action recognition [242]. Here, we advance the application of residual extraction and processing in the domain of ConvNet features to yield a novel framework for action prediction.

To combat error accumulation, we need a method to instantly update wrong predictions with correct observations. This procedure is often dubbed **data association**. Recent approaches to sequence generation prefer autoregressive architectures, which decompose multi-step prediction into single-step prediction for training and apply the same model autoregressively for testing. Owing to error accumulation, such approaches often lead to quality degeneration as the sequence becomes longer. Current time-series optimization methods (e.g., BackPropagation Through Time (BPTT)) lack the ability to inject mid-stage supervision during optimization; thus, errors in initial stages negatively impact the following results. To

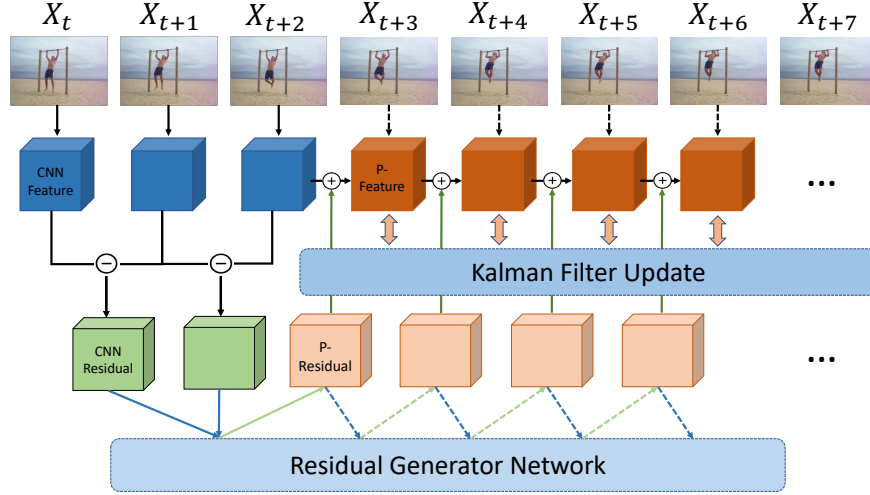


Figure 2.3: Overview of Proposed Feature Residual Propagation Approach to Action Prediction. Intermediate layer ConvNet features are extracted from an initial set of input frames; in the depicted example, these are given as $[X_t, X_{t+1}, X_{t+2}]$; subsequent frames (e.g., $[X_{t+3}, \dots, X_{t+7}]$) are not seen by the system during testing (although they are during training) and are shown here merely for context. Initial feature residuals, CNN Residuals, are extracted via pointwise differencing of temporally adjacent feature maps. A generative model, Residual Generator Network (RGN), then recursively estimates future residuals, P-Residuals. Predicted features, P-Features, are recovered via addition of residuals to the initial reference feature map. A Kalman Filter serves to minimize error accumulation across time. The Kalman Filter operates across an entire video sequence during training, but only across the initially observed partial sequence during testing. Final action classification (not shown in figure) is performed with reference to both the initially observed and predicted features. Dotted lines represent unobserved future information that is either inferred by our approach (i.e., P-Residuals) or provided as training signals (i.e., real measurements).

avoid such scenarios, we incorporate a learnable Kalman filter design [97] into our approach, cf. [30, 146]. To generate the final action label, the partially observed and propagated features are concatenated together as input to pre-trained classifiers. An overview of our Kalman feature prediction process can be viewed in Figure 2.3 and technical details are provided starting from Section 2.3.2.

2.3.2 Feature residuals

Given a partially observed video with a set of frames $X_{1:k}$, let (mid-level) features extracted at time t be denoted as $\mathbf{d}_t \in \mathbb{R}^{C \times W \times H}$, with C the number of feature channels, W the feature map width and H the feature map height. Temporal feature residuals at time t are then calculated via pointwise differencing along each channel

$$\mathbf{r}_t|_c = \mathbf{d}_t|_c - \mathbf{d}_{t-1}|_c, \quad 2 \leq t \leq k, 1 \leq c \leq C, \quad (2.4)$$

where $|_c$ indicates application to channel c , i.e., the value at spatial position (w, h) in channel c at time $t - 1$ is subtracted from the value at time t and assigned to the residual, $\mathbf{r}_t \in \mathbb{R}^{C \times W \times H}$, at the same spatial position and channel. Owing to the differencing operation, the cardinality of the set of calculated residuals, $\{\mathbf{r}_{2:k}\}$, is one less than the set of features, $\{\mathbf{d}_{1:k}\}$.

From the limited feature set $\{\mathbf{d}_{1:k}\}$ and their residuals set $\{\mathbf{r}_{2:k}\}$, we seek to recover the feature representation of $\{\mathbf{d}_{k+1:K}\}$. To achieve this result, we proceed in two steps. First, we recursively generate feature residuals $\{\mathbf{r}_{k+1:K}\}$ via appeal to a feature Residual Generator Network (RGN). Second, we sequentially add the residuals to the features that have been observed or generated so far to reconstruct features into the future according to

$$\mathbf{d}_{t+1} = \mathbf{d}_t + \mathbf{r}_{t+1}, \quad k \leq t \leq K - 1. \quad (2.5)$$

In Figure 2.3, P-Residuals and P-Features are used to distinguish predicted residuals and features, resp. In the next subsection, we define our feature residual generator.

2.3.3 Residual generator network using dynamic filter

Our Residual Generator Network (RGN) is schematized in Figure 2.4. At its core is a kernel motion transformation, $\mathcal{G}$. Given a set of stacked temporal observations, $\mathcal{G}$ produces a set of kernels, $\{K_n\}$, that can be convolved with the most recent residual input to predict the next (unobserved) result. We choose the kernel motion transformation because it has proven useful in synthesis of future intensity frames [51, 183], can be applied with various kernel sizes, $n \times n$, to capture multiple motion scales and has lower complexity than its deep regression counterpart [233].

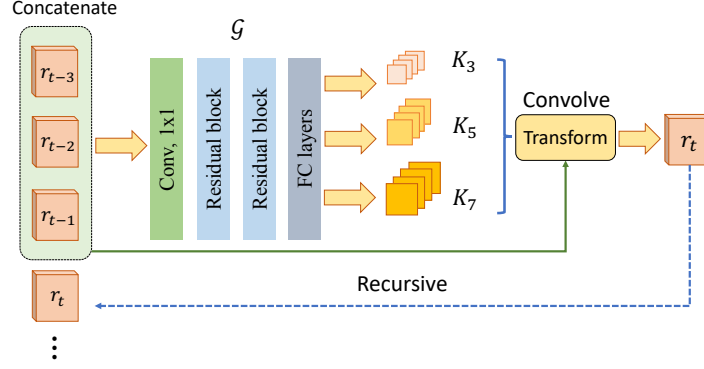


Figure 2.4: Temporal Extrapolation Residual Generator Network (RGN) for Predicting Next Time Step Residual. Our model recursively generates motion kernels, K_n , using a ConvNet, $\mathcal{G}$, based on a short historical temporal window, $m = 3$, and performs transformations in a convolutional fashion on the most recent residual. The newly generated residual joins the ongoing prediction sequence until the end of the desired sequence. Subscripts n specify kernel size (i.e., $n \times n$). Convolutions on the residuals are performed on a channel-by-channel basis; so, multiple kernels are depicted for each n .

We generate motion kernels for each channel, c , with multiple sizes, $n \times n$, according to

$$K_n = \mathcal{G}(\mathbf{r}_t, \mathbf{r}_{t-1}, \dots, \mathbf{r}_{t-m} \mid \mathbf{r}_{t-m-1}, \dots, \mathbf{r}_2; \theta_f) \big|_c, \quad (2.6)$$

where $\mathcal{G}$ is a ConvNet with learnable parameters, θ_f , that inputs residuals over its current observation window, m , but through its recurrent application depends on the entire history of residuals and thereby follows the Markov-Chain conditional distribution. Each of these kernels is normalized via the l_2 norm to avoid abnormal temporal changing, cf. [51]. The architecture of $\mathcal{G}$ is depicted in Figure 2.4, with implementation details provided in Section 2.4.1. Subsequent to kernel generation, for each channel, c , we apply the kernels to the current residual $\mathbf{r}_t$ and average the results to predict the next time step residual

$$\mathbf{r}_{t+1} \big|_c = \frac{1}{N} \sum_{n=1}^N K_n \otimes \mathbf{r}_t \big|_c, \quad (2.7)$$

where $\otimes$ stands for convolution. Based on preliminary experiments we use $N = 3$, with $n \in \{3, 5, 7\}$.

2.3.4 Revisiting Kalman filtering

Kalman filter. The Kalman filter was created as a combination of Bayesian filters and state space functions [97, 43]. It models the dynamics between the system state (e.g., vehicle dynamic systems), $\mathbf{h}_t$, and the noisy measurement (e.g., GPS tracking signals), $\mathbf{x}_t$. Typically, they are framed as conditional probability distributions that follow a Markov process, i.e., $p(\mathbf{h}_t|\mathbf{h}_{t-1})$ and $p(\mathbf{x}_t|\mathbf{h}_t)$. The classic Kalman filter needs as a prerequisite the state transition function, $\mathbf{F}$, for $p(\mathbf{h}_t|\mathbf{h}_{t-1})$, and measurement emission function, $\mathbf{H}$, for $p(\mathbf{x}_t|\mathbf{h}_t)$. Once decided, both functions, $\mathbf{F}$ and $\mathbf{H}$, remain static throughout the filtering process. One easy realization for both functions is the linear equation, e.g., $\mathbf{x}_{t+1} = \mathbf{H} \mathbf{x}_{t+1} + \mathbf{q}$, with $\mathbf{q}$ the noise. To associate with new-incoming data, the Kalman gain term, $\mathbf{K}_t$, is adopted to balance between new predictions and real, albeit noisy, measurements. Its value varies from 0 (high prediction error) to 1 (low prediction error), and is dynamically determined from the {prediction, measurement} pair at each step.

Learnable Kalman filter. Our re-design of the Kalman filter differs with the aforementioned prototype in several ways. (1) The Markov process functions, $\mathbf{F}$ and $\mathbf{H}$, are modeled as neural network modules, which can model more complicated transitions (i.e., more than just modelling linear equations) and can be easily applied to high dimensional representations (i.e., raw RGB images, in general, and deep representations, in particular). (2) The state and measurement variables, $\mathbf{h}_y$ and $\mathbf{x}_t$, are simplified as deterministic variables, which are free from Gaussian uncertainty constraints of the standard Kalman filter and can be easily determined through the learning process. (3) The online corrective term, Kalman gain $\mathbf{K}_t$, is modeled as the output of a Long Short-Term Memory (LSTM) module [86], which considers the correlation between prior predictions and measurements. Its memorizing and forgetting gate vectors help capture the long-term temporal evolution of the Kalman gain.

The Kalman filter recursively estimates an internal state from a time series of measurements via alternating *Predict* and *Update* steps along the temporal axis. In our case, the internal state corresponds to the features recovered from the predicted residuals according to (2.5), while *Predict* is formulated as the RGN defined in Section 2.3.3 and *Update* is formulated as

$$\hat{\mathbf{d}}_t^+ = \hat{\mathbf{d}}_t^- + \Gamma_t(Z_t - \hat{\mathbf{d}}_t^-), \quad (2.8)$$

where $\mathbf{d}_t^+$ is the updated posterior estimation, Z_t is the real measurement at time t (ground

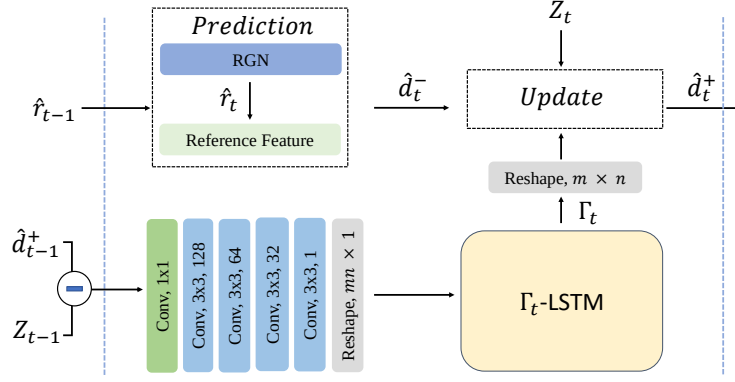


Figure 2.5: Depiction of Kalman update procedure. Prior estimation of feature $\hat{\mathbf{d}}_t^-$ is updated with Kalman gain Γ_t . The transition of Γ_t is modeled by a ConvNet with LSTM (Γ_t -LSTM) across time. At each time step, Γ_t corrects $\hat{\mathbf{d}}_t^-$ with observed measurement Z_t and produces posterior $\hat{\mathbf{d}}_t^+$ for next time step inference.

truth framewise feature), $\mathbf{d}_t^-$ is the prior estimate, (2.5), and Γ_t is the Kalman Gain. We use Γ to denote our Kalman gain to distinguish it from the typical classical notation, $\mathbf{K}$, used with the standard definition. The **Update** step corrects the current prediction by balancing the observed measurement, Z_t , and prior estimate, $\hat{\mathbf{d}}_t^-$, and the gain, Γ_t , defined as

$$\Gamma_t = \psi(Z_{t-1} - \hat{\mathbf{d}}_{t-1}^+; \theta_z). \quad (2.9)$$

In the classic definition, the Kalman gain is estimated from a ratio of the process and measurement noise, both of which are pre-defined by prior knowledge of physical aspects of the system being modelled (e.g., environmental factors, like air resistance). Such noise factors can be considered as the source of prediction errors and are readily modelled under Gaussian assumptions.

However, this design is not feasible in our work, as the size of the action states is too large, i.e., every feature point is viewed as an individual state variable and the employed features are large in size, (28, 28, 192). Further, it is difficult to obtain useful prior knowledge of errors when using deep networks. Instead, we treat the Kalman gain as the output of a nonlinear function, ψ , of the *difference* between the real and predicted features, $Z_{(t-1)} - \hat{d}_{(t-1)}^+$, which naturally reflects the residual error. We realize ψ as a ConvNet with an LSTM and learnable parameters, θ_z , cf. [30]. The architecture is depicted in Figure 2.5, with details provided in Section 2.4.1. To validate this design choice, in our empirical studies, Sec 2.4.3, we also

conduct a complimentary experiment where rather than input the difference, $Z_{(t-1)} - \hat{d}_{(t-1)}^+$, we simply directly input both $Z_{(t-1)}$ and $\hat{d}_{(t-1)}^+$ without differencing, to the ConvNet and thereby more poorly captured the desire for the update to be conditioned on the prediction error. This former Kalman gain is named **KF-2** and the later one **KF** in the following.

We explicitly incorporate the Kalman filter *Update* step into the training of the RGN, where correction happens after the estimate of $\hat{\mathbf{d}}_t^-$ is obtained, as depicted in Figure 2.5. The corrected feature $\hat{\mathbf{d}}_t^+$ is subsequently used for $t + 1$ prediction and loss computation thereafter. During training, the Kalman filter has access to true observations, Z_t throughout the video. In testing, however, the Kalman filter only has access to true observations up through the final input partial observation, X_k , and is only applied through that point, as detailed in Section 2.3.1. We find that the instantaneous correction offered by the Kalman filter helps stabilize long-term inference, as documented in Section 2.5.3.

Learning Scheme. In our approach, there are two sets of trainable parameters, θ_f and θ_z , that are associated with the kernel motion generator, $\mathcal{G}$, of the residual generative network and the Kalman gain transition, ψ , resp.

Both sets of parameters are trained using backpropagation [66] to minimize loss objective functions. We adopt a two stage training strategy that initially learns the θ_f values and subsequently learns the θ_z values, while also refining the θ_f values. We first train θ_f because it is more central to our overall approach in performing the essential prediction, rather than the correction. This design choice conforms to the standard Kalman filter paradigm that presupposes a sensible transition module and a corrective module built on rational prior estimates [97]. Nevertheless, ultimately the prediction and correction must work together; so, θ_f and θ_z are trained jointly in our second stage.

The parameters θ_f are optimized with respect to four losses. The first loss pertains to the residuals

$$\mathcal{L}_2^{res}(\theta_f) = \|\mathbf{r}_t - RGN(\mathbf{r}_{t-1}, \mathbf{r}_{t-2}, \dots, \mathbf{r}_{t-m}; \theta_f)\|_2^2, \quad (2.10)$$

where m is the temporal window size. (In (2.10), note that $\mathcal{G}$ is embedded in RGN , but here we suppress the recursive dependence on all previous residuals beyond the current observation window that was given in (2.6) for the sake of compactness of notation.) The second loss

pertains to the features

$$\mathcal{L}_2^{feat}(\theta_f) = \|\mathbf{Z}_t - \hat{\mathbf{d}}_t^-\|_2^2 = \|\mathbf{d}_t - (\hat{\mathbf{d}}_{t-1} + \hat{\mathbf{r}}_t)\|_2^2. \quad (2.11)$$

As reported elsewhere [160, 229, 17], $\mathcal{L}_2$ works under the Gaussian assumption that data is draw from a single parameterized Gaussian distribution and thus produces blurry outcomes. To counter this shortcoming, we include an additional two losses by applying the *Gradient Difference Loss* [160], which emphasizes high frequency content, on both the features and residuals to yield

$$\mathcal{L}_{gdl}^{res}(\theta_f) = \left\| \frac{\partial}{\partial x} (\mathbf{r}_t - \hat{\mathbf{r}}_t) \right\|_2^2 + \left\| \frac{\partial}{\partial y} (\mathbf{r}_t - \hat{\mathbf{r}}_t) \right\|_2^2 \quad (2.12)$$

and

$$\mathcal{L}_{gdl}^{feat}(\theta_f) = \left\| \frac{\partial}{\partial x} (\mathbf{Z}_t - \hat{\mathbf{d}}_t^-) \right\|_2^2 + \left\| \frac{\partial}{\partial y} (\mathbf{Z}_t - \hat{\mathbf{d}}_t^-) \right\|_2^2, \quad (2.13)$$

where x and y indicate the horizontal and vertical image dimensions, respectively.

The overall objective function for $\mathcal{G}$ is defined as

$$\mathcal{L}_2^{\mathcal{G}}(\theta_f) = \lambda_1 \mathcal{L}_2^{res} + \lambda_2 \mathcal{L}_2^{feat} + \lambda_3 \mathcal{L}_{gdl}^{res} + \lambda_4 \mathcal{L}_{gdl}^{feat}, \quad (2.14)$$

with the λ_i scalar weighting factors specified in implementation details 2.4.1. Note that during the first stage of training, the Kalman filter would not be operating, as it has yet to be trained.

After training the RGN parameters, θ_f , the Kalman gain parameters, θ_z , are trained, while the θ_f parameters values are refined to yield a joint optimization. Now, there are only two losses, both pertaining to the features, $\mathbf{d}$, because that is where the Kalman filter operates. The losses are analogous to (2.11) and (2.13), except that they are calculated on the updated posterior $\hat{\mathbf{d}}_t^+$ according to

$$\mathcal{L}_2(\theta_f, \theta_z) = \alpha \mathcal{L}_2^{feat}(\hat{\mathbf{d}}_t^+; \theta_f, \theta_z) + \beta \mathcal{L}_{gdl}^{feat}(\hat{\mathbf{d}}_t^+; \theta_f, \theta_z), \quad (2.15)$$

with α and β scalar weighting factors.

2.3.5 Connection of Kalman filtering to RNNs

The centre of our contribution is a better alternative to classic Recurrent Neural Networks. Inspired by one recent work that systematically studied deep Bayesian filtering [69], hereby we provide further analysis on the subtle differences between our learnable Kalman filtering and RNNs, and thereby shed light on the intuition behind our design.

Recall that the RNN formula for propagating arbitrary sequential hidden states, $\mathbf{h}_t$, is defined as

$$\mathbf{h}_t = \sigma(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{ih}\mathbf{x}_t + b_h), \quad (2.16)$$

where σ is the activation function (e.g., ReLU), b_h is the bias term added on the hidden state, $\mathbf{x}_t$ is the new input and finally two learning parameters, $\{\mathbf{W}_{hh}, \mathbf{W}_{ih}\}$, serve to project the previous hidden state, $\mathbf{h}_{t-1}$, together with the input, $\mathbf{x}_t$, to the current timestamp.

Similarly, such a formula in the language of Kalman filtering can be given as

$$\begin{aligned} \mathbf{h}_t &= \mathbf{W}\mathbf{h}_{t-1} + \mathbf{K}_t(\mathbf{x}_t - \mathbf{V}\mathbf{h}_{t-1}) \\ &= (\mathbf{W} - \mathbf{K}_t\mathbf{V})\mathbf{h}_{t-1} + \mathbf{K}_t\mathbf{x}_t \\ &= \mathbf{W}_{kh}^t\mathbf{h}_{t-1} + \mathbf{W}_{ki}^t\mathbf{x}_t, \end{aligned} \quad (2.17)$$

where $\mathbf{W}$ and $\mathbf{V}$ denote the vanilla state transition and measurement emission matrices, $\mathbf{K}_t$ is the Kalman gain, $\mathbf{W}_{kh}$ is the projection matrix for hidden states comprised of $\mathbf{W}$, $\mathbf{V}$ and $\mathbf{K}_t$ and $\mathbf{W}_{ki}$ is rewrite of $\mathbf{K}_t$ for the sake of comparison to the RNN formulation. To further facilitate comparison, also consider simplifying the RNN Equation 2.16 to become

$$\mathbf{h}_t = \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{ih}\mathbf{x}_t. \quad (2.18)$$

Discussion. From Equation 2.17 and 2.18, it is seen that Kalman filtering and RNNs indeed bear much resemblance: Future states will depend on the previous states and the incoming measurement. Yet, they differ in the matrices that are responsible for transiting the hidden states and balancing new input. More specifically, in RNNs, once learning is done, the two projecting matrices, $\{\mathbf{W}_{hh}, \mathbf{W}_{ih}\}$, are fixed, whereas the same matrices of Kalman filtering, $\{\mathbf{W}_{kh}^t, \mathbf{W}_{ki}^t\}$, change over time. The Kalman gain term, $\mathbf{K}_t$, contributes to the temporal adaptive ability of Kalman filtering as it is inferred online. We believe that it is this key difference

causing our approach to be more effective than the RNN counterparts, as observed in empirical validation, Section 2.4.3.

2.3.6 Unified model for all observations ratios

Learning a separate model for each observation ratio is not applicable in the real world. To overcome this difficulty, we design a unified training and testing strategy, as follows.

Training. The RGN begins by inputting the very first batch of residuals $[\mathbf{r}_m, \mathbf{r}_{m-1}, \dots, \mathbf{r}_2]$ and recursively produces all the rest. In other words, our model is trained for predicting the whole sequence from the same starting point, thereby entirely ignoring observation ratios.

Testing. Our testing also is invariant to observation ratio by switching modes of the Kalman filter operation so that it only corrects the estimates while data observations are available according to the observation ratio, ρ . For example, when $\rho = 0.6$, the proposed approach still starts from the beginning observations and propagates to the end, but in two modes: While the observation ratio is not yet reached, i.e., $\rho \in [0.1, 0.6]$, we update predictions via reference to the observed true data by using the Kalman filter update step, (2.8). After entering $\rho \in [0.7, 1.0]$, only prediction is performed, (2.7).

This procedure resembles tracking objects under a Kalman filter: When objects are observed, the system corrects its estimated coordinates based on true observation measurements; however, while objects are occluded, the system extrapolates possible locations based on “up-to-now” system parameter values, i.e., only the prediction step is performed.

2.4 Empirical evaluation

2.4.1 Evaluation protocol

Datasets. To evaluate our approach, we choose three widely examined datasets, UCF101 [211], JHMDB21 [92] and BIT [110]. UCF101 consists of 13,320 videos of 101 action categories containing a wide range of activities (e.g., sports, music and others). JHMDB21, a subset of HMDB [118], contains 928 videos of 21 realistic, nuanced human action categories

(e.g., catching, throwing, picking). We use the provided RGB images rather than body joints of JHMDB21. BIT consists of eight classes of human interactions, with 50 videos per class. Different from the other datasets, BIT has similar behaviors of people in the initial stage of different actions (e.g., they tend to be standing still) [113], which leads to challenges from limited discriminatory information.

Protocols. For all datasets, we use their standard train/test splits: UCF101 and JHMDB21 come with multiple train/test splits and we average over the results in our reporting, unless otherwise noted; BIT has a single train/test split, with the first 34 videos in each class for training and the rest for testing.

We present action classification accuracy as a function of observation ratio, ρ , which is the ratio of observed to total frames in a video, as used elsewhere [111]. Classification is always based on the concatenation of features derived from the observed frames and those that are predicted. For mid-layer features, which are the subject of our propagation, we use the intermediate output of two convolutional layers and two max-poolings $\in \mathbb{R}^{28 \times 28 \times 192}$, unless otherwise noted. This layer is selected because empirical comparison to others generally yielded superior performance; see Section 2.4.4.

2.4.2 Implementation details

Classifiers. To examine the propagation module with minimal influence from other factors, classifiers for chosen datasets are obtained beforehand. While a pretrained TSN model is available for UCF101 [237], models for JHMDB21 and BIT are not available. To adapt the TSN model to the JHMDB21 and BIT datasets, we append a simple two layer MLP classifier consisting of two hidden layers to TSN pretrained for HMDB-RGB and UCF101-Flow. For JHMDB21, the two hidden layers have 32 and 21 activations. For BIT, the two hidden layers have 64 and 8 activations. Softmax is used for final probability generation in all cases. During the training process all pretrained weights are frozen. For training of weights added for adaptation to JHMDB21 and BIT, we randomly select three RGB samples or three optical flow samples (each sample has 5 frames) from the videos and get video labels by segment consensus. We employ a learning rate of 0.0001, batch size of 64, *Stochastic Gradient Descent* and the *Adam* optimizer. Data augmentation is the same as for the original TSN [237].

Network configurations. For the kernel generator of the RGN, $\mathcal{G}$, stacked residuals are first convolved with a 1×1 kernel that reduces the feature dimension. Then, two residual convolutional blocks [81] with kernel size 3×3 , bottleneck dimension 48 and stride of two are used to capture temporal evolution. Subsequently, with batch and channel axis fixed, flattened spatial features are individually processed with three fully-connected (FC) layers to produce 3×3 , 5×5 and 7×7 kernels. So the shape of feature map is $(28, 28, 192 \times m) \rightarrow (28, 28, 192) \rightarrow (28, 28, 192) \rightarrow (28, 28, 192) \rightarrow (9, 192), (25, 192)$ and $(49, 192)$, with $m = 3$ the empirically selected temporal window size. Convolution is performed on each channel.

For Kalman Gain, Γ_t , a set of convolutional layers with kernel size 3×3 and stride of two are used to capture the covariance. Each layer is appended with a ReLU layer [63]. The shape of feature map is $(28, 28, 128) \rightarrow (28, 28, 64) \rightarrow (28, 28, 32) \rightarrow (28, 28, 1)$. Subsequently, the flattened feature is taken as input by Γ_t -LSTM to produce Kalman gain, $\Gamma_t \in \mathbb{R}^{w \times h}$, which then is reshaped to $\Gamma_t \in \mathbb{R}^{w \times h}$, corresponding to feature map spatial dimensions. The size of the hidden state of the LSTM is 784, the same size as the input feature. The gain is then applied according to the update, (2.8).

Training. We train our model with four NVIDIA TITAN X GPUs, under PyTorch [3]. Training of the spatiotemporal feature residual generative network (RGN) employs the *Adam* optimizer and a learning rate 0.005 with $\beta_1 = 0.9$ and $\beta_2 = 0.99$ to minimize the loss, (2.14). Empirically, we set $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ with ratios of 1:1:5:5, which places more emphasis on the spatial gradient rather than raw mean square values. The batch size is set to 56. Following initial training of the RGN, we fine-tune it together with the Kalman gain transition ConvNet with LSTM, ψ , to minimize the loss (2.15). Mini-batch-Stochastic Gradient Descent is used with a learning rate of $2e^{-4}$ and exponential decay of $1e^{-5}$. α and β are set empirically with a ratio of 1:5.

For training on UCF101, we sample 30 frames from each video and use the first three to initialize our entire prediction system. For BIT and JHMDB21, we sample 25 frames from each video and use the first three to initialize our system. The TSN architecture [237] serves to provide feature extraction and classification. We apply our system to the RGB stream for JHMDB21, flow stream for BIT and both streams for UCF101. We make these choices following typical state-of-the-art practice on JHMDB21 (e.g., RGB features previously yielded top performance [194, 205]), BIT (e.g., flow features greatly outperform spatial features [110, 112]) and UCF101 (e.g., two-stream previously yielded top performance [113]).

Once features are generated, no additional modifications to TSN are needed to yield action labels. Generated features are inserted into the selected TSN mid-layer and processed up through the network tower until the MLPs produce probability scores. Video level labeling is gathered by averaging scores from each frame.

2.4.3 Comparisons to alternative approaches

UCF101. Figure 2.6 (a) shows comparative results for our algorithm **RGN-KF-2** versus various alternatives on UCF101 as well as the variant of our approach, **RGN-KF**, where rather than input the difference of the observed and predicted measurements to the gain calculation, (2.9), we more simply input the two measurements independently, as motivated in Section 2.3.4. It is seen that both our approaches, **RGN-KF-2** and **RGN-KF**, outperform all others at all observation ratios, improving accuracy by $\approx 3\text{-}4\%$ on average. The performance improvement is especially striking at lower observation ratios, e.g., $\rho = 0.1$, where we outperform the second best (AAPnet [113]) by 83.78% versus 59.85%. When being compared with the simpler approach, **RGN-KF**, our final approach, **RGN-KF-2**, uniformly surpasses it on low observation ratios, e.g., $\rho \in (0.1, 0.2, 0.3)$, with the largest improvement at $\rho = 0.2$ (i.e., 86.22% versus 85.16%). Beyond that, **RGN-KF-2** performs equally well. Since the task is *early* action recognition, improvements at the lowest observation ratios are of particular interest.

Notably, AAPnet also builds on TSN backbone features; however, it apparently does so less effectively than our approach does. There are likely two reasons for this state of affairs. First, AAPnet is not trained exclusively for inferring action labels, but also for adversarial learning on feature transformation, which might lessen its optimization for action prediction. Second, AAPnet more radically modifies the TSN architecture in aggregating across all frames at a given state of progress, which underlines the fact that our approach may be more adaptable to various architectures as it has less impact on their native operations.

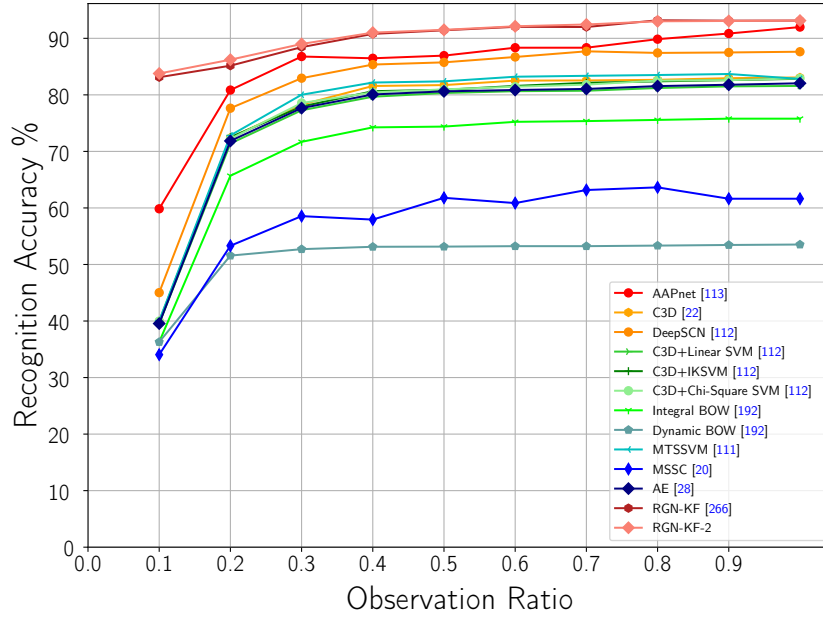
BIT. Figure 2.6 (b) shows comparative results for our algorithm versus various alternatives on BIT. It is seen that our results are equal to or better than all others, except at the lowest observation ratio, $\rho = 0.1$. For example, compared with AAPnet, our approach achieves 69.12% accuracy at $\rho = 0.3$, which is 4% higher. Notably, our **RGN-KF-2** obtains non-trivial improvements over **RGN-KF** on multiple lower ratios, e.g., $\rho \in (0.1, 0.2, 0.3)$. Particularly at

$\rho = 0.2$, our final approach boosts the accuracy from 46.09% to 49.96%, which demonstrates the solid benefit of explicitly modelling the Kalman gain from error signals, as given by, (2.9). Again, improvements at lowest observation ratios are of particular interest for early action recognition.

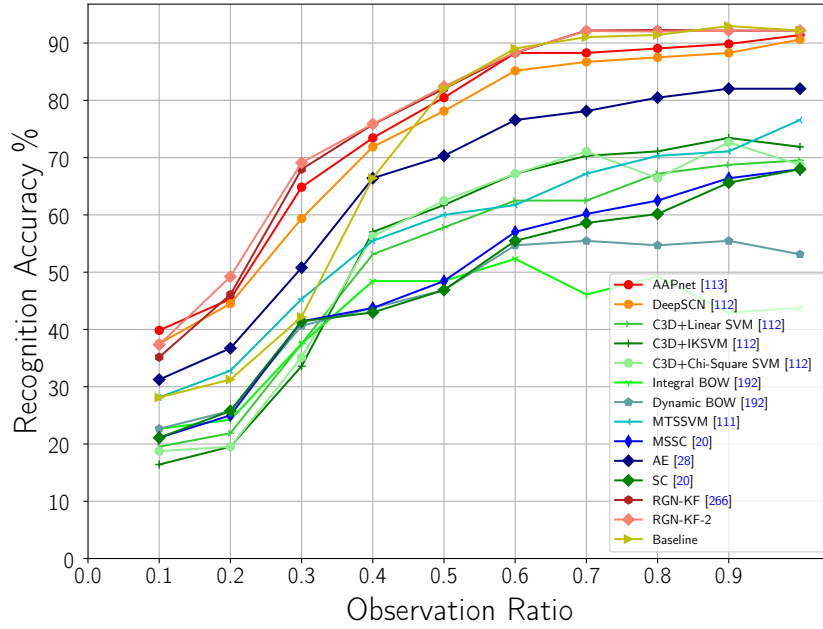
In interpreting the results on BIT it is important to recall that the beginning and ending portions of the videos tend to be very similar in appearance (e.g., two people standing facing one another), so that the most discriminatory information largely is limited to the intermediate portions. Correspondingly, there is a tendency for rapid performance rises after the initial portion, which levels out in the final portion. In our case, a peak performance of 92.28% at $\rho = 0.7$ increases that at the previous ratio by 4%, whereas AAPnet achieves no significant increase (0.78%) at the same stage.

Given that we train a modified TSN architecture in adapting TSN to BIT (Section 2.4.1), we compare how well that modified architecture works when forced to classify on just the initially provided frames without propagation into the future. These results are shown as Baseline in Figure 2.6 (b). It is seen that by propagating into the future our approach exceeds the baseline by large margins when $\rho \in [0.1, 0.4]$. For higher observation ratios, as the discriminating parts of the input videos become available to the baseline (as well as our full approach), performance becomes on par.

JHMDB21. The standard reporting protocol on JHMDB21 is to report recognition accuracy only when the initial 20% of the videos are observed, i.e., $\rho = 0.2$, which we show in Table 2.1. It is seen that our algorithm once again is the top performer, e.g., **RGN-KF-2** exceeding the RBF-RNN by 6% and **RGN-KF** by 1%. We also provide a baseline comparison, where we compare to classification made purely on the basis of adapting the TSN architecture to the JHMDB21 dataset, analogous to the baseline comparison provided on BIT. Once again, it is seen that our full propagation approach adds considerably to the performance of the baseline alone. As there exists no previous approach that evaluate on all observation ratios, we provide our version of such results in Figure 2.7. The baseline is from direct inference on partial observations, without future predicting. Other variants shown in the figure are explained in Table 2.2. It is seen that the residual processing together with the **KF-2** design yields the best performance.



(a) UCF-101 dataset



(b) BIT dataset

Figure 2.6: Action Prediction Results on the UCF101 and BIT Datasets at all Observation Ratios $\rho \in [0.1, 1]$.

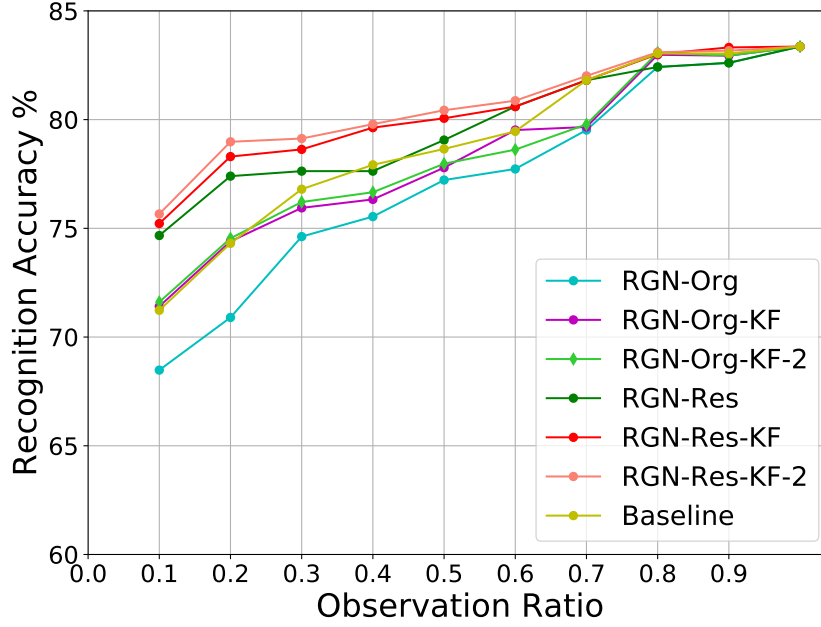


Figure 2.7: Action Prediction Results on the JHMDB-21 Datasets at all Observation Ratios $\rho \in [0.1, 1]$.

2.4.4 Ablation studies

Temporal models. In this section, we examine the influence of different temporal modeling approaches to feature propagation using JHMDB21, with ConvLSTM as an extra baseline, cf. [249, 17]; see Table 2.2. For both scenarios, we find that propagation on residuals is superior to propagation on raw features and the Kalman filter provides further benefits. Performance of ConvLSTM is on par with our RGN approach applied to the original features without the Kalman filter; however, for all other configurations our RGN approach performs better. Especially, our final version of the Kalman filter, **KF-2**, seems to benefit all used temporal models. Overall, we find that our full approach to temporal modeling (mid-layer convolutional feature residuals, RGN propagation and Kalman filtering) yields best performance.

Feature levels. We now examine the influence of different intermediate feature spaces on prediction. We consider layers that yield feature maps of dimensions [56, 56, 64], [28, 28, 192], [14, 14, 512] and [7, 7, 1024], where [w, h, c] indicate the width, height and number of channels, resp. Table 2.3 shows the results. For JHMDB21 and BIT, the [28, 28, 192] feature stage almost always achieves best results. Moreover, deeper layers, [14, 14, 512] and [7, 7, 1024], are more useful than the shallower layer [56, 56, 64]. This pattern of results may be

Method	Accuracy (%)
ELSTM [194]	55
Within-class Loss [150]	33
DP-SVM [210]	5
S-SVM [210]	5
Where/What [209]	10
Context-fusion [90]	28
RBF-RNN [205]	73
RGN-KF [266]	78
RGN-KF-2	79
Baseline	74

Table 2.1: Action Prediction Results on JHMDB21. Following the standard protocol, accuracy results are shown only for the case where initial observations are limited to the first 20% of frames, i.e., $\rho = 0.2$.

explained by the earliest layer not providing adequate abstraction from the raw input, while the highest layers have lost too much distinguishing detail. Interestingly, for UCF101 different feature stages have less impact on accuracy. This may be due to the fact that UCF101 is generally less difficult than the other datasets, as indicated by the fact that for any given observation ratio, ρ , in Table 2.3 the results on UCF101 are always better than for the others; correspondingly, the specifics of feature selection are less important. More generally, however, the results of Table 2.3 support our use of intermediate layer features, especially as the prediction task becomes more difficult.

2.5 Interpretability

In this section, we provide interpretation of the various components of our approach to early action recognition. We begin by investigating why mid-layer features are especially useful for this task, as documented in the previous section. (More general interpretation of TSN features is available elsewhere [237]). Subsequently, we provide detailed examination of the two major learned components: 1) dynamically inferred motion kernels, $\{K_n\}$, that propagate feature residuals into the future; 2) the Kalman Gain, Γ , that updates sub-optimal prior estimates.

Temporal Propagation Approach	Accuracy (%)
ConvLSTM(3x3)-128-192- Org	71.1
ConvLSTM(3x3)-128-192- Org-KF	73.4
ConvLSTM(3x3)-128-192- Org-KF-2	74.3
ConvLSTM(3x3)-128-192- Res	76.8
ConvLSTM(3x3)-128-192- Res-KF	77.1
ConvLSTM(3x3)-128-192- Res-KF-2	77.3
RGN- Org	70.9
RGN- Org-KF	74.4
RGN- Org-KF-2	74.7
RGN- Res	77.4
RGN- Res-KF	78.3
RGN- Res-KF-2	79.0

Table 2.2: Accuracy results for different temporal propagation approaches on JHMDB21 split 1. **Org** denotes applying motion kernel transformation on original features, **Res** denotes residual propagation and **KF*** denotes inclusion of the Kalman filter. For ConvLSTM, (3x3), 128 & 192 represent kernel, hidden state & feature dimension, resp. Here **KF** indicates Kalman gain calculated in a fashion of joint inputs, while **KF-2** is their difference (2.9).

2.5.1 Intermediate layer features

To understand why intermediate layer features and their residuals are especially useful for action prediction, we show comparative visualizations as well as associated statistics.

Figure 2.8 provides an example from the action baseball-swing. It is seen that the earliest layer features concentrate on low-level information (e.g., lines and edges) that may be too closely tied to a specific example, rather than the action class. In contrast, the latest layer features tend to lose too much distinguishing detail (e.g., merely a blob in the vicinity of the actor at the top-layer). Comparatively, the mid-layer features tend to concentrate on the actor, but also delineate details of the actor’s parts. In comparing the raw features to their residuals, it is seen that the residuals concentrate more on temporal changes, which are good for propagating information that is especially useful for action prediction.

The provided visualization, Figure 2.8, suggests that the residuals provide a more sparse (and hence compact) representation compared to the features per se. To quantify this observation, we define feature sparsity as the percentage of near-zero points (absolute value < 0.01) versus total points. Figure 2.9 shows comparative results for original features and their resid-

	UCF-101				BIT				JHMDB-21
<i>observation ratio</i>	0.1	0.2	0.3	0.8	0.1	0.2	0.3	0.8	0.2
56x56x64	83.16	85.03	87.92	92.10	33.41	46.65	67.51	84.38	75.83
28x28x192	83.78	86.22	89.01	93.07	37.33	49.96	69.12	88.82	79.00
14x14x576	83.29	85.46	88.19	93.06	36.62	48.70	68.19	87.50	77.63
7x7x1024	80.51	84.94	88.17	92.77	36.12	49.22	67.61	87.50	77.42

Table 2.3: Prediction Accuracy (%) at Various Intermediate Feature Stages with **RGN-KF-2**, Ordered by Decreasing Spatial Receptive Field Size. Observation ratio $\rho \in \{0.1, 0.2, 0.3, 0.8\}$ for UCF101 and BIT datasets. Set $\rho = 0.2$ for JHMDB21.

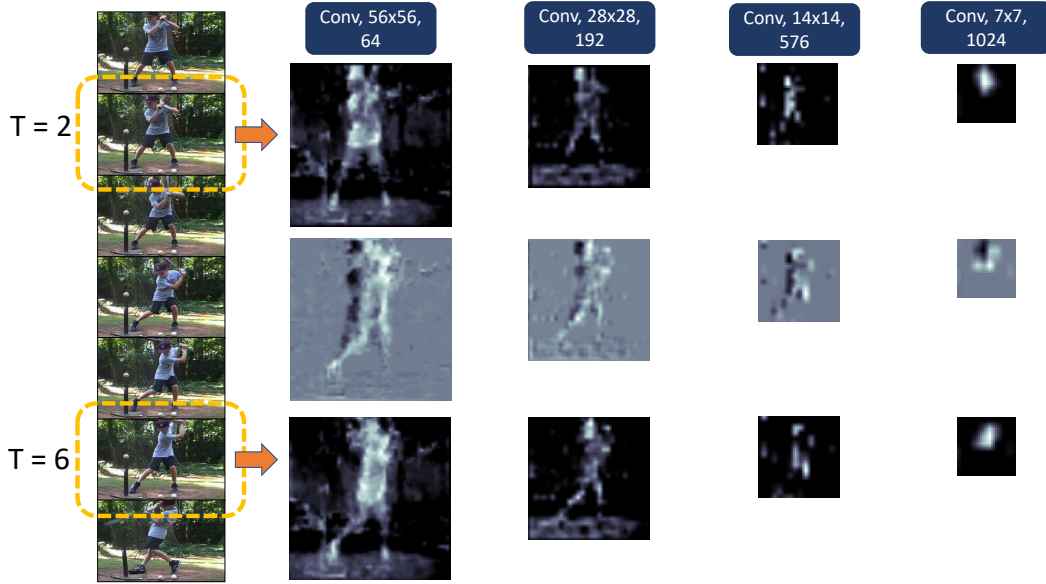


Figure 2.8: Visualization of Features and Residuals. The example shows a sequence of frames for the action *baseball-swing* along the left side. Feature maps extracted at various layers are shown along the upper/bottom row, while their residuals are shown along the middle row.

uals. It is seen that the residuals have approximately five times the sparsity of the originals, which quantitatively confirms the relative sparsity of the residuals.

Overall, both the visualizations and the quantitative analysis confirm that mid-layer feature residuals are especially information laden for action prediction without unnecessary redundancy.

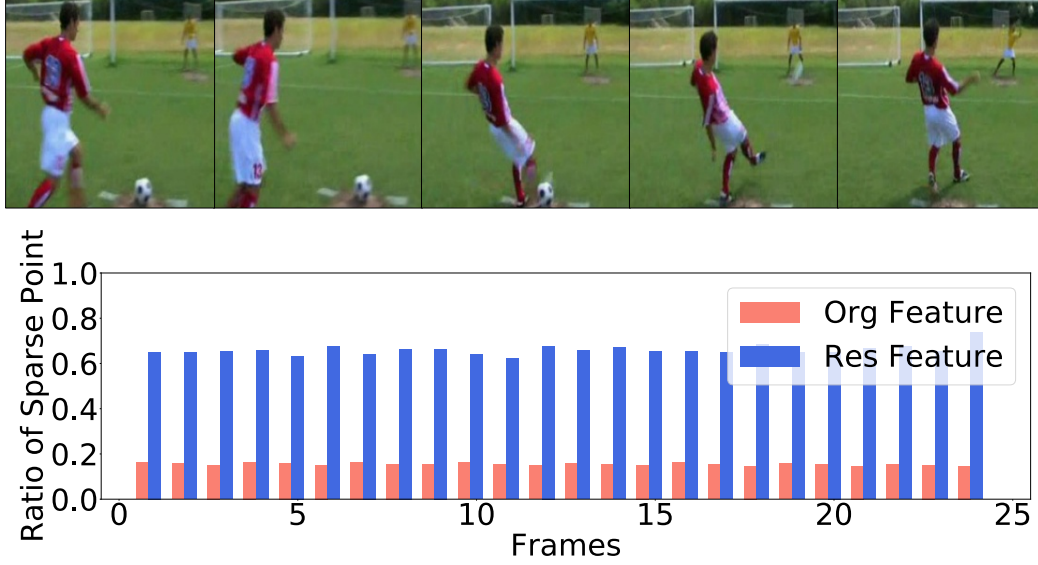


Figure 2.9: Sparsity Comparison of Features and Residuals. The top row shows frames from a video of a *kicking* action. The bottom row shows sparsity as the ratio of near-zero value points (absolute value < 0.01) to total points over time. On average, residual and original feature points are 65% and 14% sparse, resp.

2.5.2 Learned motion kernels

Convolutional spatial shifting. To understand what has been learned by the motion kernels, we begin by visualizing them, as typically seen in deep network interpretation, cf. [259, 225, 48]; representative results are shown in Figure 2.10. In all the cases, it is seen that the learned kernel has a high intensity value displaced from its center in the direction and magnitude of the depicted motion. For example, in the top row the motion is mainly from the actor stretching his arm towards the left by approximately one pixel, with additional minor motion towards the top (denoted as white arrow). Correspondingly, the motion kernels have peak value toward the left at approximately one pixel and also some smaller values in the top and top-left. The other rows also show similar patterns of a displaced bright spot capturing the motion direction. Moreover, not only is the motion direction captured, but also its magnitude: For the largest motion (second row) the displacement in the largest kernel is displaced by three pixels from the center, while the smaller kernels displace to the extent they can; for the intermediate magnitude motion the displacement never goes beyond two pixels; for the smallest displacements (top and bottom rows) the displacement is one pixel for all kernels. Interestingly, learned kernels across all actions in all datasets tend to show similar patterns.

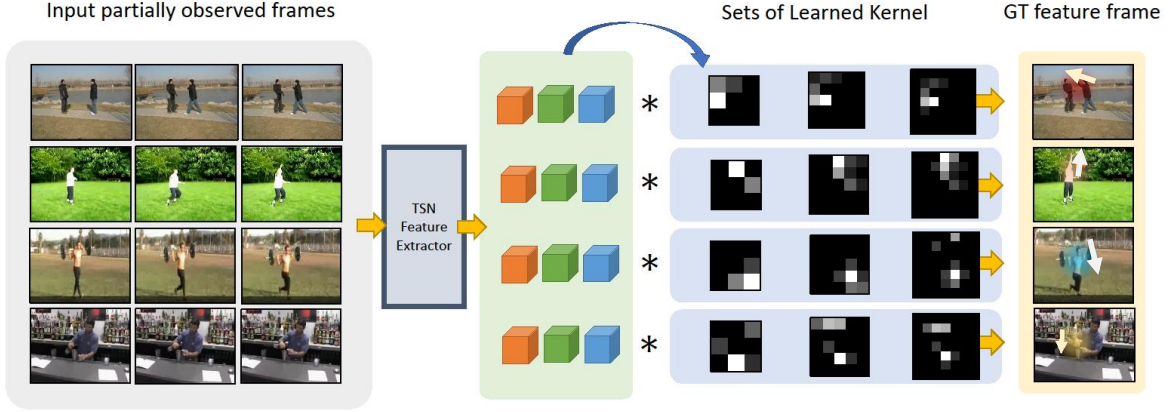


Figure 2.10: Visualization of Learned Motion Kernels. Left-to-right: Input frames and symbolized extracted features; learned kernels (3×3 , 5×5 , 7×7); ground-truth prediction target frames that follow the inputs, with an overlaid vector showing the dominant motion direction. Top-to-bottom are *boxing*, *catching*, *lunges* and *opening bottle* actions. Plots of the learned kernels resemble unit impulses that capture the depicted motion.

Here, it is important to note that the motion kernels are computed on a per feature channel basis and that different feature channels capture different components of the images: Some channels are better matched to foreground actors or even actor parts, while others are better matched to backgrounds. These different channels may therefore yield different motions and our per channel learning automatically follows these various motions.

What is the significance of such learned kernels? An explanation can be had by recalling that signal shifting via convolution is captured by operating on the signal with a unit impulse that is shifted by the desired amount, i.e.,

$$g(t) \circledast \delta(t - \Delta t) = \delta(t - \Delta t) \circledast g(t) = g(t - \Delta t), \quad (2.19)$$

where $\delta(\cdot)$ denotes the unit impulse and $g(\cdot)$ denotes an arbitrary function [16]. In this light, the learned kernels can be interpreted as (noisy) unit impulses that through convolution shift the feature maps into the future according to the motion that has been observed in previous frames.

To give more intuition, Figure 2.11 depicts the convolutional shifting operation with a toy example: Moving a triangle within an image using convolution. In the left subfigure, we

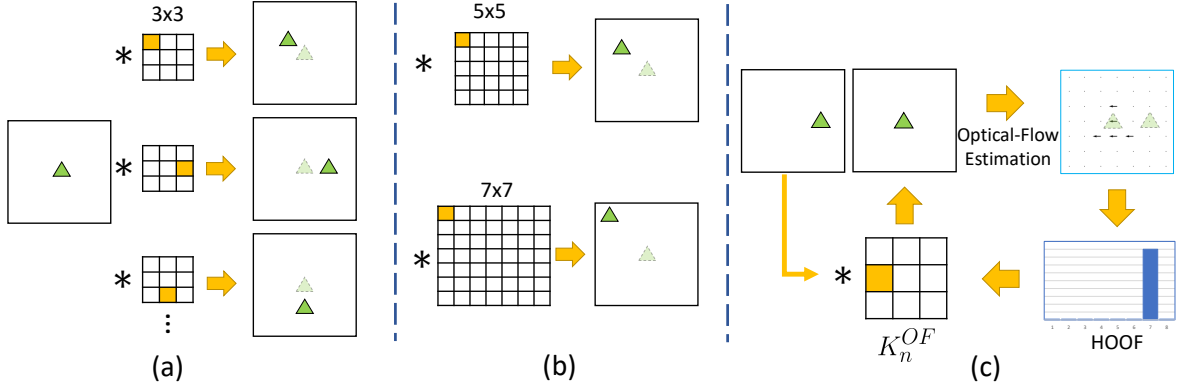


Figure 2.11: Illustration of spatial shifting via convolution using 2D unit impulse kernels. The positions of the impulse signal (yellow block) specifies the movement directions (a) and the displacement magnitude of the motion kernel specifies the movement stride (b). Optical flow kernels, K_n^{OF} , can be constructed from Histograms of Oriented Optical Flow (HOOF) to propagate frames in a convolutional manner (Section 5.1.2) and thereby can be compared to our learned motion kernels to verify that they are capturing observed motion through convolution (c).

demonstrate that there are eight degrees of freedom in terms of motion directions for a 3×3 unit impulse kernels, i.e., the eight outer entries. Once applying the convolution with any of them, the targeted triangle would be shifted accordingly by one pixel distance, or $\sqrt{2}$ for the diagonals. Arguably, the limited shifting stride can be a shortcoming for temporal feature propagation, as it enforces a small motion assumption. In the middle subfigure, however, it is shown that the shifting stride can be extended by expanding the size of motion kernels, i.e., moving by two or three pixel distance per operation with 5×5 and 7×7 kernels, respectively. Thus, kernels with multiple scales are more flexible to capture motions that cross greater distances and also support fine-grained orientations, e.g., 5×5 kernels can provide 16 more directions. Our approach has adopted this multiscale scheme.

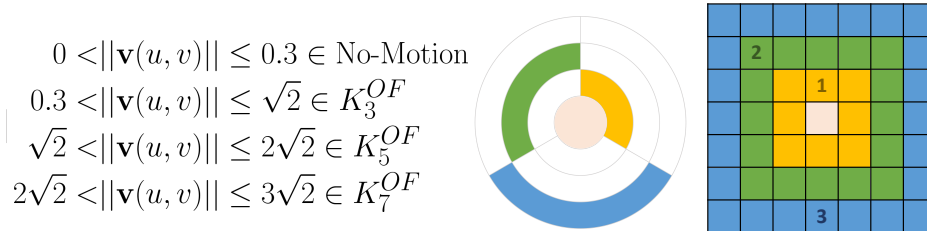


Figure 2.12: Discretization of flow magnitude for conversion to motion kernel format. Motion kernels are divided into 0, 1, 2 or 3 rings, for no motion, 3×3 , 5×5 and 7×7 kernels, resp., with rings corresponding to flow magnitude.

Motion kernels from optical flow. To further verify that the learned motion kernels are capturing the observed motion, we compare them to optical flow estimated from the same input frames. Apparently, no previous approach for comparing learned motion kernels with optical flow has been proposed. In response, we proceed by building on the classic Histogram of Oriented Optical Flow (HOOF) [26, 145], with details as follows.

Let $\mathbf{v}(x, y) = (u(x, y), v(x, y))$ be optical flow, where we use a standard estimator to recover the flow [44]; although, that particular choice is not critical, i.e., any reliable optical flow estimator could be used. Note that since we will be comparing to kernels that are learned on a feature channel basis, the corresponding feature channels are used as input to the flow estimator, rather than raw grey-level images. We define the flow magnitude, $\|\mathbf{v}(u, v)\| = \sqrt{u^2 + v^2}$, and direction, $\theta = \tan^{-1}(\frac{v}{u})$, in the usual way, where we suppress dependence on (x, y) for compactness of notation. For conversion to the motion kernel format, we quantize the flow magnitudes and directions across (x, y) into histograms with bins that correspond to the discrete kernel elements. The magnitude is quantized as shown in Figure 2.12. Note that different rings are available for quantization depending on the kernel size: 3×3 kernels only encompass the inner most ring; the 5×5 kernel encompasses both the inner most and middle rings; the 7×7 kernel encompasses all three rings. Flow direction is discretized according to an angular range for bin b , out of a total of B bins as $-\frac{\pi}{2} + \pi \frac{b-1}{B} \leq \theta < -\frac{\pi}{2} + \pi \frac{b}{B}$, where $1 \leq b \leq B$ and B is the number of orientations that can be captured by the outer ring of the motion kernel (i.e., 8, 16 and 24 for motion magnitudes mapped to the inner, middle and outer rings, respectively). Each bin of the histogram includes designated flow vectors weighted by their magnitude, as in classic HOOF. Finally, the histograms are smoothed with a Gaussian filter [145] and normalized to complete the conversion of optical flow to motion kernel format, yielding K_n^{OF} .

Match results. Figure 2.13 (a) illustrates our methodology for quantifying the comparison of learned motion kernels, K_n , with ground-truth optical flow kernels, K_n^{OF} , as defined in the previous section. As specified in the figure, we take the inner product between the vectorized versions of $\tilde{K}_n$ and $\tilde{K}_n^{OF}$ to quantify their match, with $\tilde{\cdot}$ denoting vectorization. Notice that since both K_n and K_n^{OF} are normalized by construction, their inner product is equal to their cosine similarity, which tends to focus more on the correlation of shifting directions between the learned motion kernels and the optical flow kernels. In the following, we present results for all three datasets, JHMDB21, BIT and UCF101 as well as overall discussion as a series of

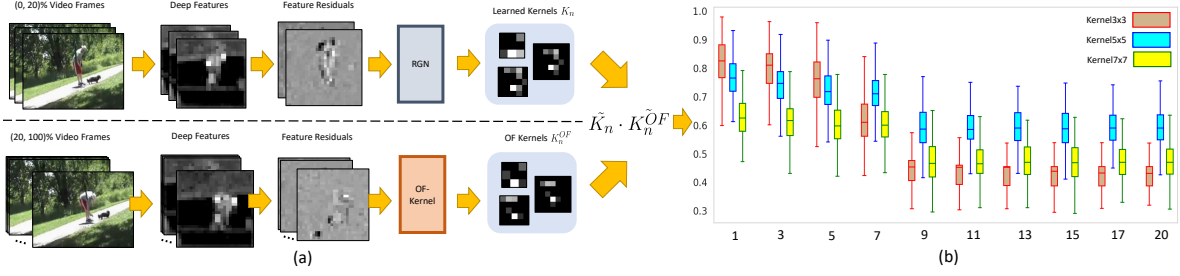


Figure 2.13: (a) Pipeline for computing the match between learned motion kernels, K_n , and optical flow kernels, K_n^{OF} , for the entire prediction horizons on JHMDB21, in the deep feature residual domain. (b) Box plots of match results across 192 feature channels. In each column, the vertical extent of the box encompasses the interquartile range, the horizontal line in the box indicates the median score and the whiskers extending above and below the box indicate the 90th and 10th percentiles, resp.

questions and answers.

Results for JHMDB21 are shown in Figure 2.13 (b) for 10 timesteps that are sampled out of all prediction horizons (i.e., 22 timesteps). It is seen that all three sets of learned kernels can demonstrate high matches for the first 7 timesteps: 3×3 kernels have median match score of ≈ 0.8 with OF-kernels; 5×5 are at ≈ 0.7 ; 7×7 are at ≈ 0.6 . Thereafter, the matches decline along prediction horizons (e.g., beyond the 9th timestep, 3×3 kernels can only achieve 45%).

It also is notable that for the 1st-7th prediction timesteps, the 3×3 kernels achieve the highest matches and the 7×7 kernels the lowest. It also is interesting to observe that the 3×3 kernels degenerate more remarkably than the other two when above the 9th prediction step. We return to both of these observations below.

Similar patterns of correlation results for BIT and UCF101 are given in Figure 2.14. As the BIT dataset includes mostly small actions (e.g., *high-five*, *patting* and *handshaking*) and actors execute their actions smoothly, the match scores for the 3×3 kernels are generally higher than the other two kernel sizes compared to that seen with JHMDB21. However, due to BIT depending on stacked optical flow input, which naturally contains noise, its best match value is still lower than JHMDB21. Notably, the variability of its 7×7 kernels are much larger than those for JHMDB21, which likely is because the depicted slow actions do not yield large displacements that would be relevant for the largest kernel. For the UCF101 dataset, the results are not as well correlated as the others, i.e., the median match scores for all three sets of kernels are lower (e.g., 0.56 at the 1st timestep). These results may arise because of the

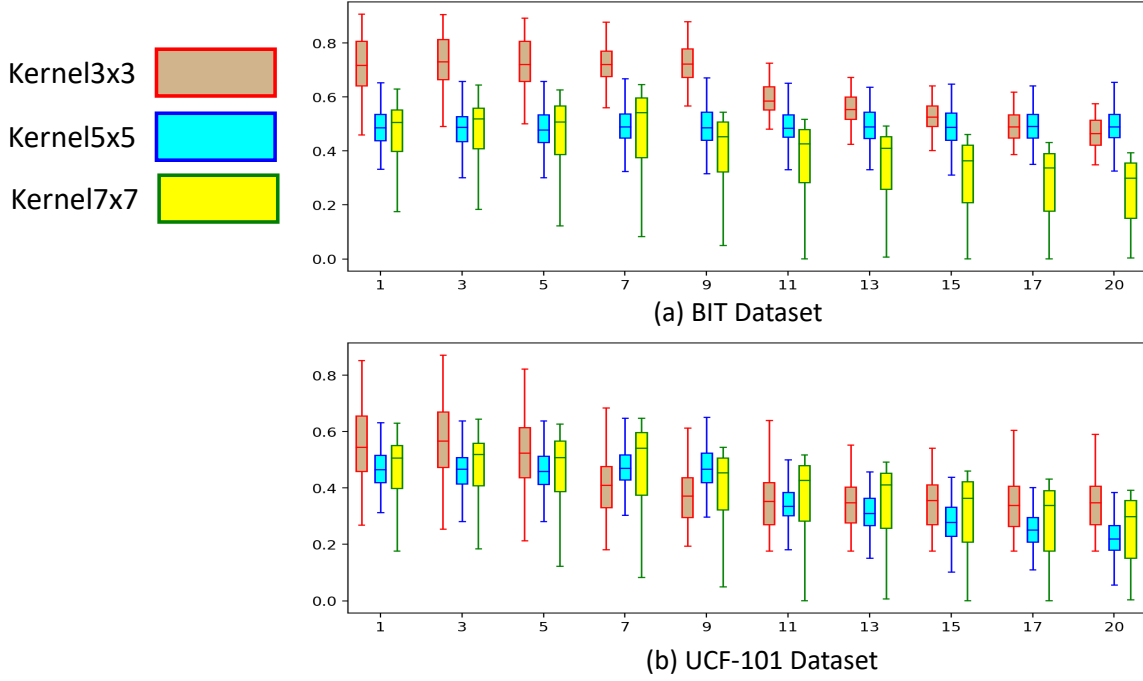


Figure 2.14: Box plots of correlation scores between learned kernels, K_n , and optical flow kernels, K_n^{OF} , for the BIT (a) and the UCF101 (b) datasets.

greater complexity of the motions captured in UCF101, e.g., arising from real-world sports video.

Are there performance differences by feature channel? Recall that the employed deep feature residual input to the motion kernels is a multi-channel tensor from the TSN middle stage, i.e., intermediate feature with dimensions [28, 28, 192]. Our approach propagates each channel with a distinct set of kernels. Here, we visualize a few representative channels and examine the match differences between them; see Figure 2.15. It is seen that certain feature channels, e.g., eight and seventy, do well in capturing the actions (i.e., high feature map responses delineating the actor), while other feature maps (e.g., 184), are mainly noise. Thus, their matches differ greatly: Clean channels show reasonable match scores between the learned and ground-truth kernels, whereas noisy channels show very low values throughout the predictions.

Is it necessary to use larger kernel sizes? From the 7×7 kernel plots in Figure 2.13(b), it is seen that these kernels have relatively low correlation scores even at early timesteps, which raises doubt regarding their usefulness. Those results aggregate across all frames and actions. Examination of particular cases, however, shows the importance of larger kernels. Figure 2.16 provides an illustrative comparison. The top row depicts a *kicking* action where the execution leads to a relatively large displacement (i.e., half of its pixels exhibit flow magnitude

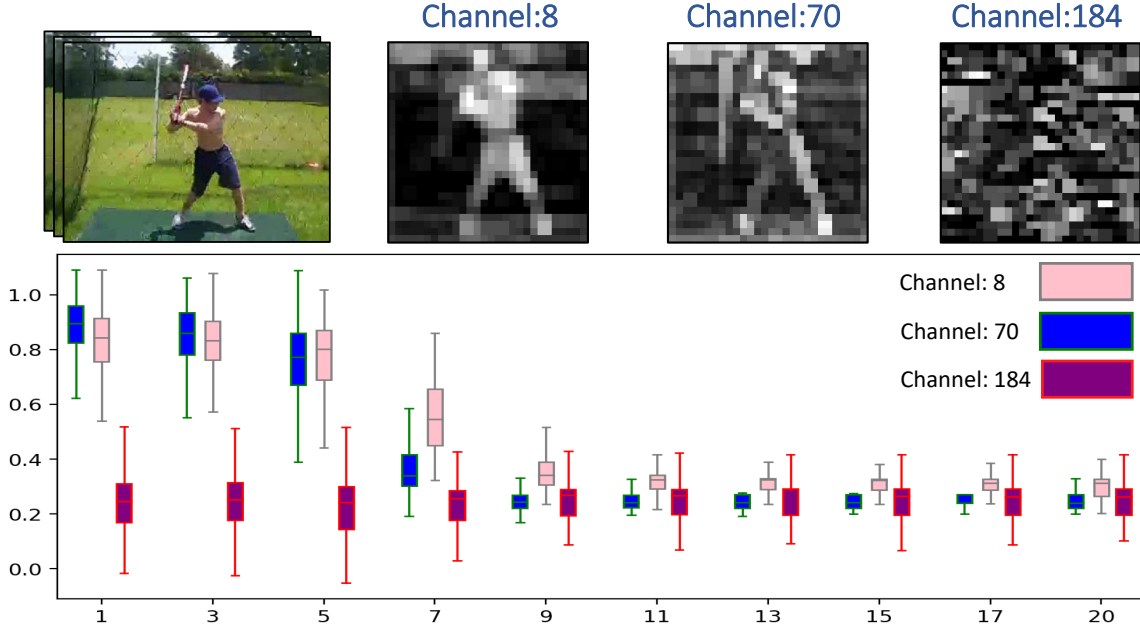


Figure 2.15: Visualization of three distinctive feature channels (top). Plots of the averaged matches of the learned 3×3 kernels for three selected channels across the test set (bottom). The visual quality of feature channels reflects the match scores.

greater than two) and our learned 7×7 kernels have high valued impulses at the three pixel displacement positions (the outermost ring). In the bottom row, however, the *standing* action executes slowly and the learned kernels reject having high values at the outermost ring. By our counting, 21% of deep feature residuals across the entire test set encompass flow magnitude larger than two and these are captured best by the 7×7 kernels.

What are the failure modes of the learned motion kernels? Another observation from Figure 2.13 (b) is that the 3×3 kernels tend to degenerate more severely at longer prediction horizons than the other two, e.g., its correlation goes down to 0.4 when prediction horizon reaches the 9th timestep and onward. To probe further, we show visualizations of a sequence of learned and ground-truth 3×3 and 5×5 motion kernels; see Figure 2.17. It is seen that the learned 3×3 kernels eventually degenerate to indicating no-motion (i.e., highest response at the center position), whereas the learned 5×5 kernels continue capturing the motion and thereby yield higher matches. This pattern is likely due to the depicted motion being relatively large and the smaller kernel has limited ability to encompass the displacement even at the beginning; this imitation is exacerbated as the shifts that it propagates lead to ever less accurate predictions at longer horizons. Thus, we see that a failure mode of the approach is capture of larger motions at longer horizons, especially as predicted by the small motion

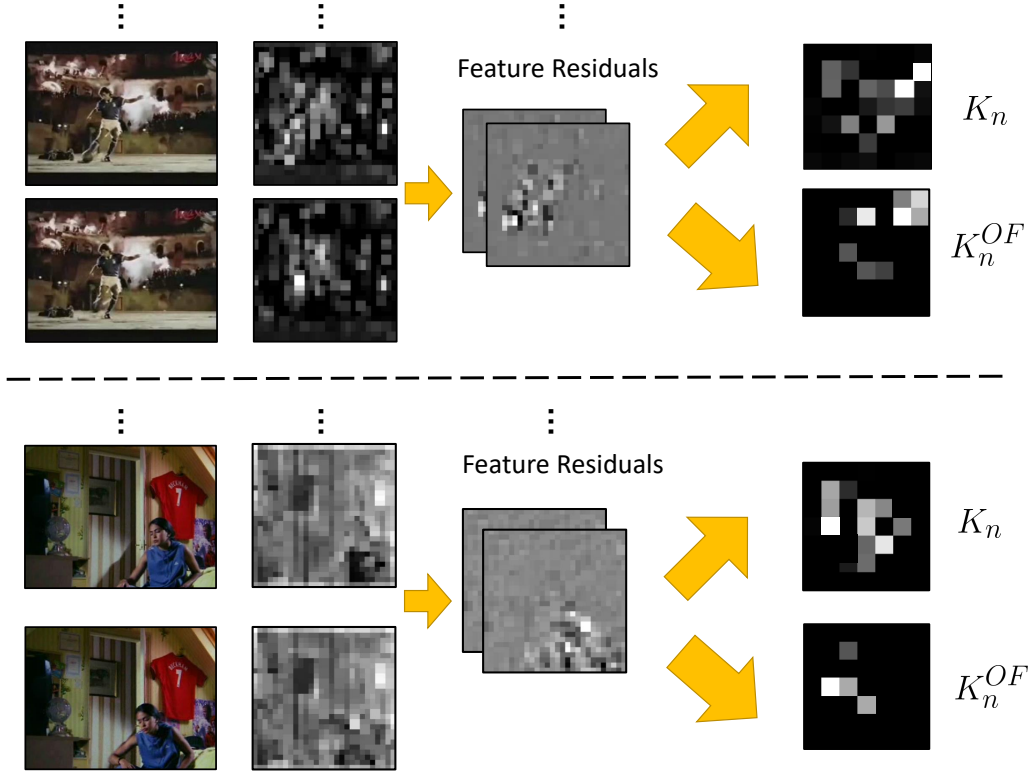


Figure 2.16: Plots of kernels for action videos whose deep feature residuals have flow magnitudes larger than 2 (top) and smaller than 2 (bottom).

kernels. This limitation is the major one that we have discovered.

Is easy-to-propagate equal to easy-to-predict? Our approach is based on the assumption that propagation into the future would bring benefits, e.g., discovering the unobserved information. To examine this assumption, we consider the relationship between accuracy improvement versus mean feature propagation error on every action category. Results on JHMDB21 are shown in Figure 2.18. The lower left subfigure shows that when being compared with the simple baseline approach (i.e., using only the first 20% of the video to predict the entire action without any further processing, as discussed in 2.4.3), our approach, RGN, sees the most performance improvement on the *sit* and *picking* actions, i.e., *sit* is improved from 33.4% to 58.3% accuracy and *picking* is improved from 66.7% to 91.7% accuracy. The lower right subfigure shows a plot of mean-square-error between propagated and ground-truth deep features. It is seen that the smallest error arises for *sit* and *pick* features, showing that these actions are relatively easy to propagate for our approach. These observations suggest that our success on action prediction is from the effectiveness of our feature propagation.

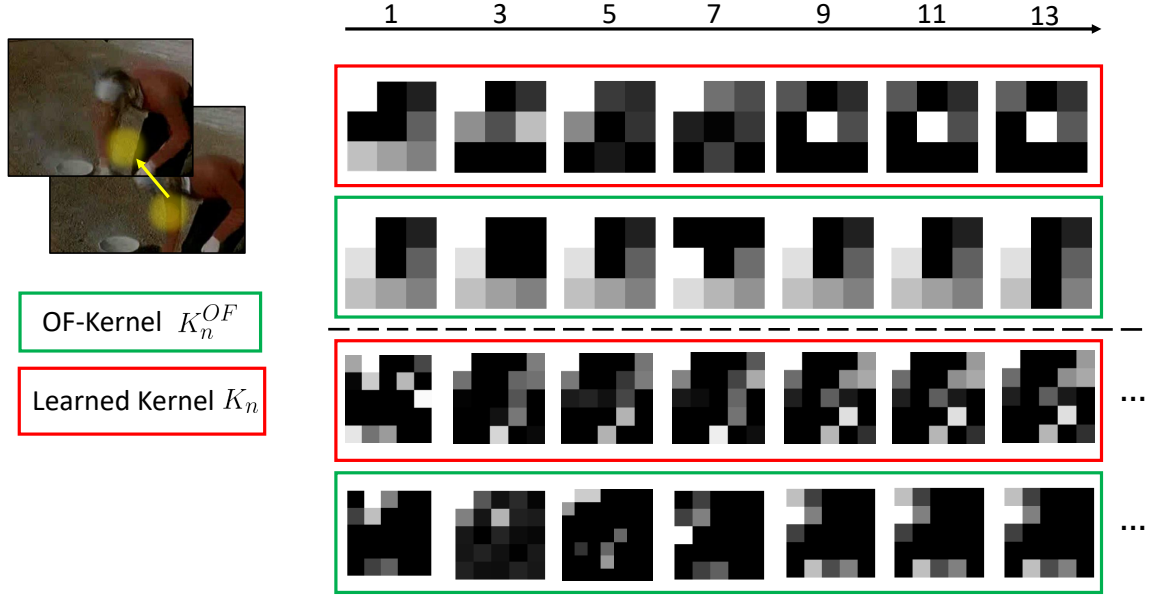


Figure 2.17: Visualization of failure modes of 3x3 and 5x5 kernels on a *picking* test action video. Smaller learned kernels (upper part of figure) are less able to capture larger motions than larger learned kernels (bottom part of figure), especially as the temporal horizon increases.

Finally, visualizations of these most improved examples and their most frequently confused categories are shown in the upper portion of Figure 2.18. It is seen that *sit* can be easily confused with *standing* when looking only at the static image, as they both share the same physical environments (e.g., chairs). Similarly, *picking* and *climbing stairs* share some common subtle movements, e.g., lifting-up legs. Nevertheless, our system yields greatest advantage on these examples, suggesting that its approach to predicting the future yields gains in discriminative power.

2.5.3 Learned Kalman gain

We now examine the learned Kalman gain by plotting its values across prediction horizons. To place our results in context, we begin with a brief discussion of related work.

Injecting ground-truth mid-stage information into a recursive generation process has been seen in various previous efforts to stabilize learning, e.g., scheduled sampling [12] and teacher forcing [239]. Long-term sequences often suffer from error accumulation due to the poor quality of intermediate results; therefore, these approaches use real midway data at scheduled

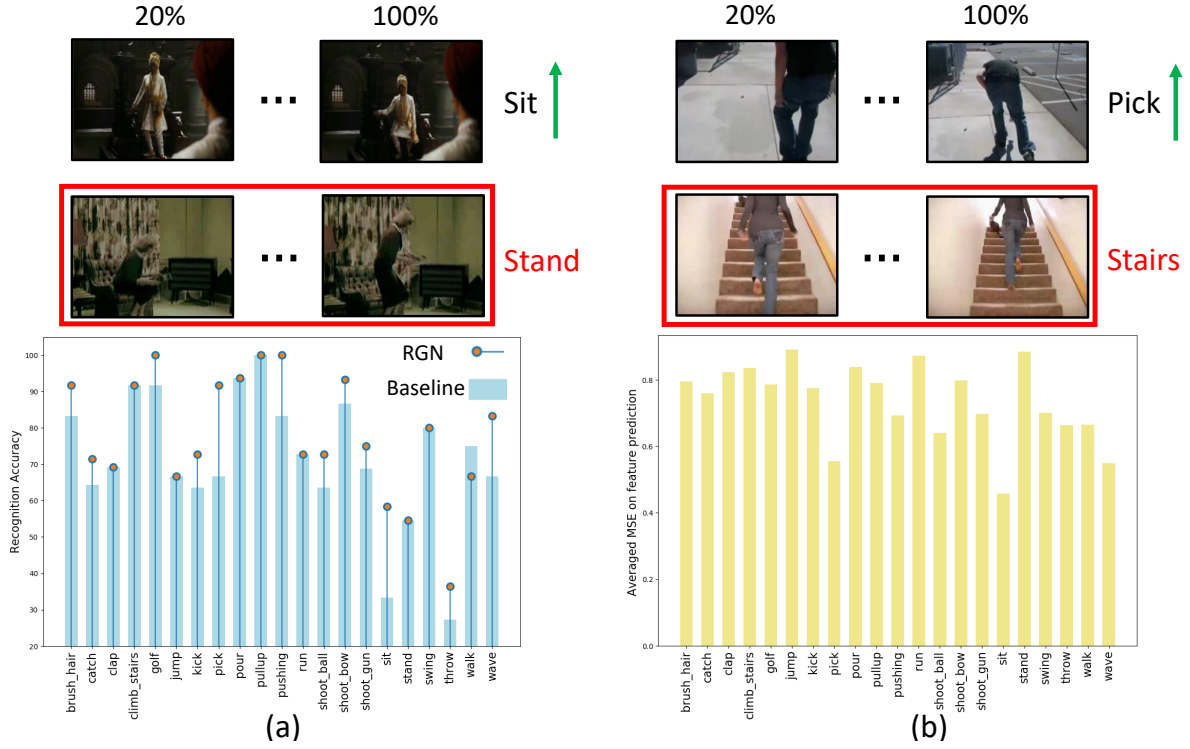


Figure 2.18: (Top) Visual examples for action *sit* and *climb stairs*, along with their most confused categories in red boxes). (Bottom) Early action recognition accuracy from our framework and baseline approach, (a). Mean feature propagation error per action category, (b).

times to replace the undesired results. The Kalman filter update step serves similar purposes, but does not simply replace some predictions with real measurements: Instead of completely discarding the predictions, it corrects them proportionally to current estimates of prediction quality according to the Kalman gain. Moreover, the update is continual, rather than simply at predetermined schedule times.

In our approach, the Kalman gain is derived (i.e., learned) from the difference between the prediction and measurement, (2.9), and maps to a value range $(0, 1)$, with larger values providing a larger update. By design, the gain update is continual and the question becomes whether it follows the prediction and measurement difference, as desired. An example is shown in Figure 2.19 showing an example of *baseball swing*. When the prediction reaches the 7th timestep, the propagated feature begins to degenerate (shown in red boxes) and meanwhile the Kalman gain increases, indicating that the prior estimate is less reliable. After the update at the 11th timestep, the corrupt feature is corrected (shown in green box) and accordingly the gain value returns to a lower value, e.g., declining from 0.83 to 0.47.

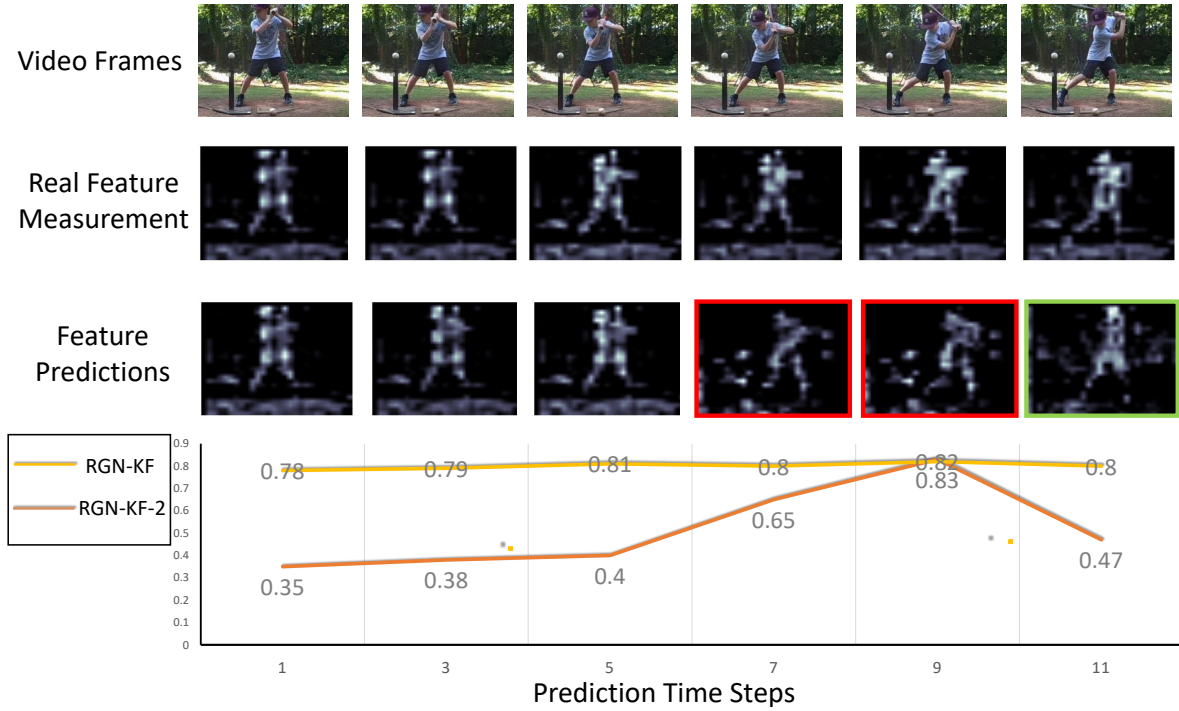


Figure 2.19: Illustration of learned Kalman gain values.

We further observe that for smooth actions, e.g., Figure 2.19, the Kalman gain starts with small values and then smoothly increases to larger values until a major update is needed and applied, after which it decreases. This behaviour follows the standard scheduled sampling strategy [12], where the probability of using on-going predictions decays as timestep increases, but in our case the update is continual and is driven directly by the current state of the estimation, rather than a predefined schedule.

In contrast to the case of smooth motion, Figure 2.20 illustrates performance in more complicated scenarios. Subfigure (a) depicts the situation where there is a sudden change in the direction of motion at the very beginning of the propagation and the gain immediately goes from high to low values. This behaviour relates to recent work that found reverse scheduled sampling useful for video frame prediction training [243], but again our approach is adaptive to the input. Subfigure (b) provides an even more complicated example (*clapping*) of multiple changes in motion directions in short time periods and the behaviour of the Kalman gain, which is seen to provide larger corrections when the direction of motion changes. These examples illustrate that scheduling strategy should be adaptive for action videos, as there exists quite diverse motion patterns, and our Kalman filter design fullfills the need in providing updates when the motion direction changes and the prediction fails.

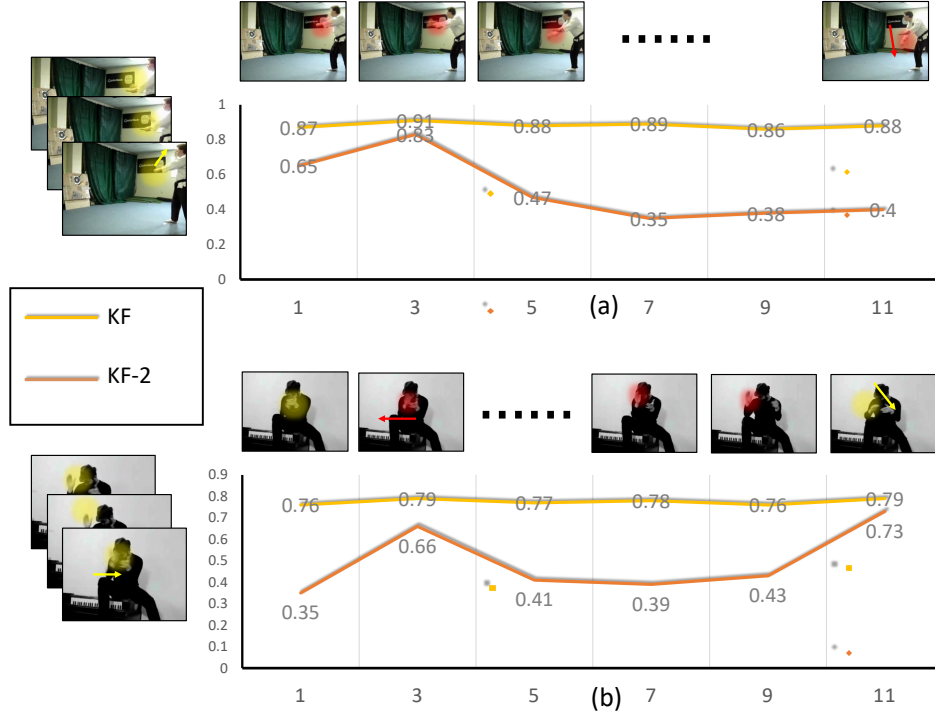


Figure 2.20: Visualization of learned Kalman gain of **RGN-KF-2** and **RGN-KF** on examples from the JHMDB21 dataset.

In addition to the plots of our final Kalman gain shown in Figures 2.19 and 2.20 we also show the values provided by the simpler approach to learning the gain, **RGN-KF**. As noted in Section 2.3.4, the simpler approach was based on inputting both the prediction and observation to the learning procedure, rather than their difference. It is seen that our final approach does much better at providing corrections when most needed. In contrast, the simpler approach provides uniformly high gain values, which will be prone to induce exposure bias when testing [184].

Finally, we consider why the UCF101 dataset enjoys the least accuracy improvements using the new versus final Kalman gain formulation among the three datasets considered. We illustrate with representative examples in Figure 2.21. It seems that certain action videos, e.g., *ShotPut*, *WallPushup* and *JumpingJack*, have solid benefits and their Kalman gain values show a similar pattern as for JHMDB21 (Figures 2.19 and 2.20), i.e., the gain increases only at particular timesteps to correct predictions and otherwise stays relatively low. In contrast, for the *HighJump* class, the gain remains uniformly high, likely due to the fact that the videos in the class show continual dramatic changes across time, e.g., actors are missing at the beginning of videos and scene backgrounds drift in a substantial way due to camera

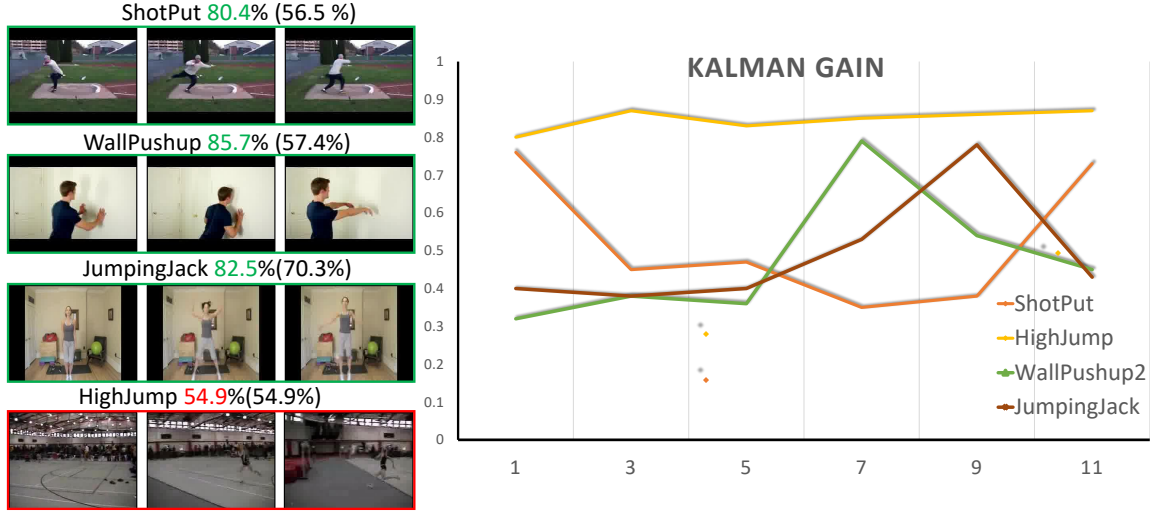


Figure 2.21: Visualization of changed early action recognition accuracies from **RGN-KF-2** versus **RGN-KF** (in parentheses) on UCF101 at observation ratio $\rho = 0.2$ (left). Plots of **RGN-KF-2** gain at observation ratio $\rho = 0.2$ across time for all four classes (right).

motion. These changes make it necessary for **RGN-KF-2** to update with high Kalman gain across time, similar to what was seen for **RGN-KF** in Figures 2.19 and 2.20; therefore, the two approaches yield similar accuracies. UCF101 tends to depict the most continual temporal changes compared to the other datasets considered, which leads the gain, and therefore final result, of the two approaches to be especially similar on this dataset. Indeed, cases where the two Kalman formulations across all datasets yield similar performance typically arise when the change within the videos is large and ongoing. Still, the final approach is important for cases where updates are required less frequently (e.g., in the top three rows of Figure 2.21 as well as Figures 2.19 and 2.20) and can thereby help avoid exposure bias, as discussed above.

2.6 Conclusion

In this chapter, we reviewed recent developments in early action recognition, introduced a novel contribution along this line, i.e., a Kalman filtering inspired deep network that propagates intermediate ConvNet feature residuals into the future, and deeply investigated *what has been learned* by the presented model. Our study shows that incorporating classic temporal modelling ideas with current trends (e.g., deep networks) is very beneficial, e.g., state-of-the-art results for early action recognition on three public benchmarks. Moreover, our investiga-

tions reveal that the learned components can be well understood with the basic knowledge of convolution, e.g., convolution with shifted unit impulse as well as adaptive schedule sampling, and thus bring an important perspective into deep network interpretability.

A number of limitations to the described approach can be noted. First, and foremost, the current implementation practice (i.e., video data pre-processing) fails to support the problem definition: **The observation ratios, ρ , can be misaligned with the phase of action**, which breaks the validity of the study. For instance, it could happen that the first 30% of the video contains the full execution of the associated action, which breaks the definition of “*early*”. Notably, this limitation holds not just for the presented work, but also more generally for all contemporary early action recognition studies. Second, even though major parts of our approach have been well interpreted, there are other components awaiting to be further analyzed, e.g., the behavior of LSTM modules in the learned Kalman filtering, in particular. Third, there is a need to extend current studies to unconstrained environments, e.g., untrimmed videos.

Chapter 3

Future activity prediction

3.1 Motivation

Other than recognizing certain *single* actions at their early stage, another emerging topic has caught researchers' attention, that is long-term future activity prediction. It differs from the previously discussed problem, early action recognition, in that its goal is to produce predictions of subsequent future actions. As illustrated in Figure 3.1, the observed frames (as solid boxes) and unobserved frames (as boundary-dotted boxes) in early recognition (bottom row) share the same label, such as *weight lifting*, whereas in activity prediction (top row) the observed and unobserved are often different yet logically related.

The ability to deduce possible future events from previous and current observations is one of the core functionalities needed by contemporary intelligent systems. In the example of automated traffic systems, i.e., artificial intelligence guided vehicles, every driver needs to be cautiously aware of surroundings and would benefit from smart prediction systems that can tell the specific next action of in-coming vehicles (e.g., turning-right), to better plan its own motion in response (e.g., slow-down or keep driving). Early action recognition does not satisfy such a demand, as it simply predicts the completion of an in progress action and thereby lacks sufficient prediction horizon to anticipate future actions. Exploring the plausible future is well studied in other fields, e.g., weather forecasting [249] and stock price prediction [151]. In the computer vision community, however, researchers have only recently become heavily active in exploring solutions.

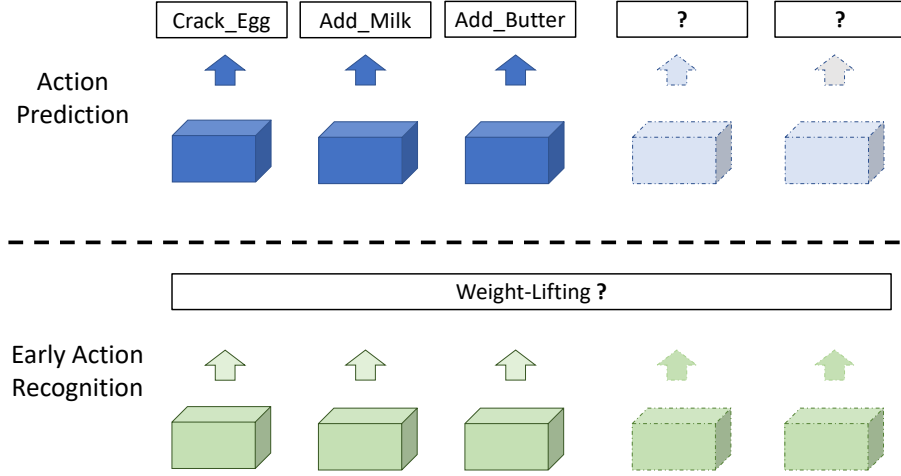


Figure 3.1: Illustration of the general dissimilarity between future activity prediction and early action recognition. The latter (bottom row) aims to infer a class label for an ongoing action from partial initial observations; the former (top row) aims to anticipate (strings of) future actions from initial observations. Solid and dotted boxes symbolize observed and unobserved portions of a video, respectively.

3.2 Related work

To examine future activity prediction in detail, we group existing research into three major divisions: (1) activity singleton or sequence prediction, (2) joint prediction of activity semantics and times and (3) egocentric video based prediction. The first division discusses the most frequent setting for future action prediction: given the raw video frame observations, $y_{1:t}$, and/or its action semantics, $x_{1:t}$, produce the semantic label for actions that would happen afterward. The prediction can be either the immediate single following action, i.e., x_{t+1} , or a sequence of future actions, i.e., $x_{t+1:t+n}$. Note that the prediction provides only action semantics (e.g., labels); time of occurrence and duration are left ambiguous. The second division considers not just prediction of sequences of future actions, but also their times. The last division tries to assist action forecastings in egocentric settings, where observed frames are taken from first-person-view and thus mimic real life carry-on camera settings.

3.2.1 Activity singleton or sequence prediction

This section describes the general version of future action prediction, which is mapping the video input, that often contains a sequence of observed actions, into its subsequent sequence of actions. The length of the predicted sequence can vary from **one** to **many**.

Hereby, we formally define the problem according to one of the initial works in future action prediction [23]. Let a sequence of observed video frames be given as $y^{obs} = (y_1, y_2, \dots, y_t)$ and their corresponding frame-wise action labels be given as $\hat{x}^{obs} = (x_1, x_2, \dots, x_t)$. Note that the symbol x^{obs} represents the ground-truth video level labels annotated through human labor. However, in most studies, researchers estimate the action labels with existing methods, rather than directly use the ground-truth video level labels. To clarify this nuance, we denote the estimated action labels as $\hat{x}^{obs}$. Since actions can take large time spans, the frame-wise action labels can be either repeating or idle (no activity performed). Our task is to produce the future action semantics $x^{unobs} = (x_{t+1}, x_{t+2}, \dots)$ that happen at unobserved time indices, i.e., $t + n, n > 0$. The set of future time indices of interest, $N = (n, n + 1, n + 2, \dots)$, are referred to as the prediction horizon. So in other words, the task is to seek a function, f , that maps $y^{obs} \rightarrow x^{unobs}$ with specified prediction horizon, N .

An initial research effort relied on a Various order Markov dependency Model (VMM) [132], i.e., a generative model implemented as graph (or tree) structures, to model the temporal relation among activity sequences, as shown in Figure 3.2 (a). The intuition is to use the **depth of trees** to represent the temporal development of activities. To represent complex activity sequence with action atoms (e.g., simple action units), these approaches advocate representing such information as a sequence of ordered discrete alphabet symbols (e.g., grammars) and group them with *n-gram* representations. Work that relies on such structures for future activity parsing include [133, 173, 132, 77]. In the example of [132], the authors choose to build a Probabilistic Suffix Tree (PST) for representing complex activity structures [132]. The PST structure is learned from training data, with the pretrained tree providing the chained transition probability for each suffix. Thus, the inference of future actions can be viewed as finding the most probable suffix leaf node given prefix node inputs, as seen in Figure 3.2 (b).

The obvious drawback of Markov based methods is that the length of prediction is predefined in the training set. For example in [132], the height of the PST is decided during training and thus the model can not be easily extended to undefined future lengths. More-

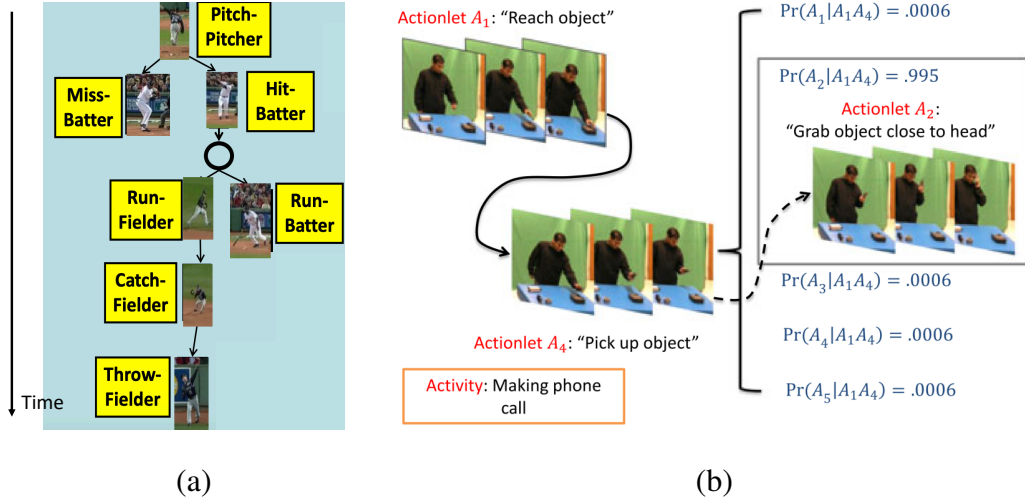


Figure 3.2: (a) Example of representing an activity sequence as a tree graph [73]. The temporal development is structured as tree nodes and leafs in a top-down fashion, where the top stands for the start and bottom for the end. (b) A path through the tree yields a set of ordered symbols, $\{A_i\}$, that capture an entire action sequence. Figure reproduced with permission from [132].

over, the *n-gram* representation of video activity is an over-simplified approach, as critical information contained in image space is ignored (e.g., appearance of agents or objects).

Instead of using generative probabilistic models as above, a discriminative instance of Markov Chains, Conditional Random Fields (CRFs) [119], has been suggested to combine with particle filters for anticipating future human-object interactions [114]. This combination enables a straightforward temporal extrapolation, as the particle filters are a type of recursive Bayesian filter that can propagate into unlimited time and space. In particular, they jointly modeled the future activity semantics with object affordances as well as movement trajectories.

The aforementioned approaches build predictive models on a structured **high-level** semantic space, mostly for **long-term** activity predictions. In real life, however, scenarios exist where the instantaneous (e.g., **short-term**) future needs to be predicted from raw video input or even visual features. For example, pedestrians walking can happen in a wide range of places and times. As another example, in surveillance camera settings, actors can enter the field of view at times that obscure their previous actions, in which cases observing one's full activity sequence is impossible. To solve such problems, many researchers resort to a direct mapping from y^{obs} to future actions.

One early effort that followed this scheme proposed to rely on cues distilled from multiple visual sources to predict the next driver maneuver [89]. Specifically, it collected information from four information sources: (1) facial motion trajectories and head angles detected from the driver-facing camera and feature tracking techniques, (2) speed logger from the vehicle system, (3) outside context from the road-facing camera and (4) the GPS map. They fused these four types of features with a novel HMM model where the maneuvers of drivers play as the hidden variables, outside sensory as control units and inside sensory as observations. However, since HMMs can not handle very high dimensional data, they converted most sensory cues into binary features, e.g., if vehicle speed is larger than 15km/hr, set the feature flag as 1 otherwise 0. To encourage more expressive features, e.g., 3D human head/facial models, the authors made an upgrade on the sequential modelling module [90]. They replaced the hidden Markov model (HMM) with a recurrent neural network (i.e., an LSTM) and modulated four sensory channels with two individual RNNs. The final fusion was done by MLPs, followed by softmax classifiers.

Similar work that focuses on single short-term (e.g., **singleton**) action prediction directly from past raw video input appears in many sub-areas. One such effort proposed to leverage environment-agent interactions to anticipate the very next risky action as well as localize future risk regions [260]. Another worked on anticipating traffic accidents from vehicle cameras through stacking local detected object features and global context [221]. Yet other work built a graph Convolutional Network (GCN) to model interactions between multiple agents and contextual objects to forecast next future activities [140, 137, 138, 136]. Pedestrian road-crossing intention prediction is another concrete example of action prediction in traffic scenarios [179, 181, 180]. Again, in these efforts, past observations contain little or no useful activity semantics, instead researchers found it helpful to mine cues from subtle details. As an example, an effort on anticipating crossing of pedestrians fused human poses, human visual representation extracted from deep networks, scene contextual information and vehicle speed, with a RNN, to make predictions [181].

3.2.2 Joint prediction of activity and occurrence time

Activity anticipation has drawn considerable recent attention and has evolved from relatively simple next frame prediction to more challenging **asynchronous** sequences of activities and

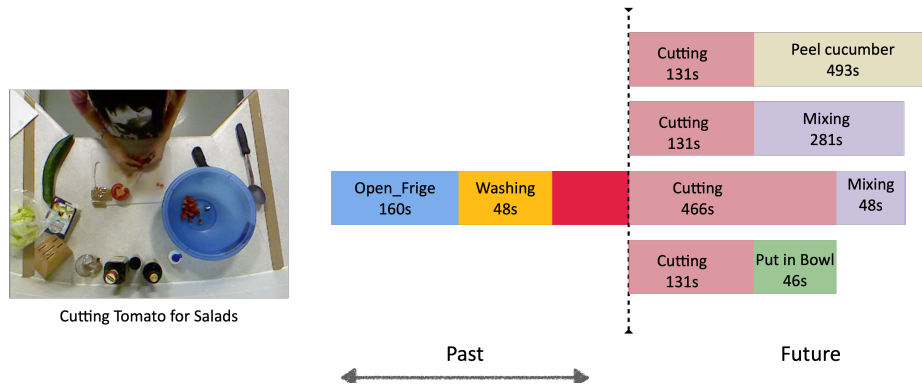


Figure 3.3: Predicting future activity labels and time durations. The left hand side depicts initial observed portions of a video from which semantic activities with their temporal duration are extracted (box length indicates duration in the illustration). The right hand side shows several possible predictions. **Uncertainty** is intrinsic to both future semantics and time duration.

times that occur further into the future [154, 156]. Indeed, the occurrence time is an important attribute of activities, e.g., *cutting meat* often takes longer than *open lid*. The ability to know *how much time would it take to finish* can be critical in applications like automated navigation, human-computer interaction as well as for the intellectual challenges it presents. All video-based action prediction work forecasts upcoming activities, here, the distinction is that both the activities and their times are predicted. An example is provided in Figure 3.3.

One particular characteristic of activities, **uncertainty**, is also gaining consideration [161, 1]. For example, given certain starting procedures of preparing a meal in a kitchen, initial actions *take_egg*, *butter_pan* can be followed by *crack_egg*, *fry_egg* for a fried egg breakfast, or alternatively *spoon_flour*, *pour_milk*, etc. for a pancake breakfast, as well as many other possibilities as in Figure 3.3 for yet another activity (making a salad). Indeed, not only the activities, but also the durations can vary (e.g., cooking time differs between *soft* and *hard* boiled eggs). This phenomenon, referred to as **diversity** in prediction, is one of the core abilities of the human prediction system, while it is much less well captured in contemporary computer vision.

An initial effort in this direction [155] adopted the Poisson process [103] as a key technique for activity inter-arrival time modelling. Notice that inter-arrival time differs from the absolute activity occurrence time in that it stands for the duration between the starting and ending of a certain action. The probability density function for the action interval random variable, τ ,

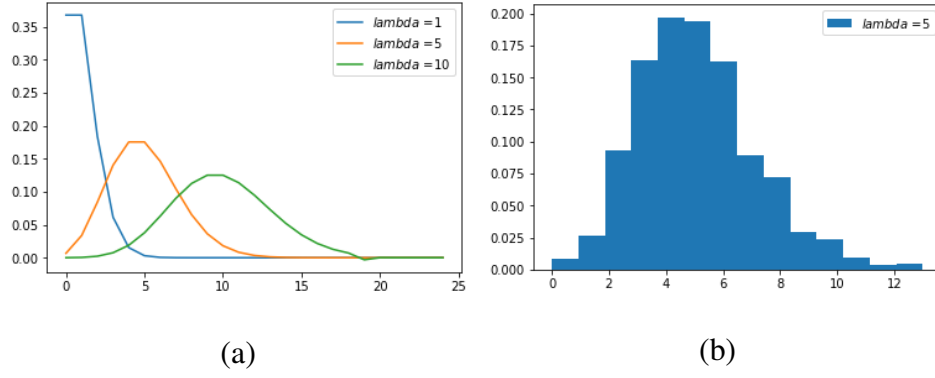


Figure 3.4: (a) Probability density function of Poisson distribution regarding various parameters λ . (b) Histogram distribution of 1000 sampled inter-arrival times from Poisson process with $\lambda = 5$. Since Poisson is a member of the exponential family, its distribution looks somewhat similar to a Gaussian distribution, but its data points are always positive integers.

under the assumed Poisson distribution is

$$p(T = \tau) = \frac{\lambda^\tau}{\tau!} e^{-\lambda} \quad (3.1)$$

where λ is the intensity function that controls the shape of the Poisson probability density function and its physical meaning is the average number of events among the whole time span. The Poisson distribution and sampling are plotted at Figure 3.4 (a) and (b), respectively.

There are three properties of the Poisson process that are worth noticing: (1) Its samples are constant values. Equation 3.1 incurs a **factorial** function on time, τ , thus its result must be an positive integer; additionally, by default researchers assume there is no negative time value, so that $\tau > 0$; (2) its shape is unimodal, e.g., an exponential family distribution with one mean centroid; (3) it is **memoryless** in that the sampling of the current step is independent from any previous samples.

To support a stochastic modelling of time, the authors pre-defined a Gaussian Process prior for $\lambda(\tau)$ and learned its parameters (i.e., the mean and variance for certain actions and times) from training data for every action. Then, they performed testing through importance sampling. Unfortunately, they only train and test on a single action input and single action output, without considering recursive temporal predictions.

An extension of the initial work [155] used the **temporal point process**, i.e., the autoregressive version of Point Process, where the intensity parameter, λ , is dynamically calcu-

lated from past observations and therefore the probability density function (PDF) is

$$p(T = \tau_k) = \lambda(x_{1:k})e^{-\lambda(x_{1:k}) \tau_k}, \quad (3.2)$$

where $x_{1:k}$ represents past information up to the k_{th} time index, such as raw frames, semantic labels or past action times, and $\lambda(\tau_k|x_{1:k})$ can be viewed as a conditional intensity whose value is regressed from past information, $x_{1:k}$. The PDF of k_{th} time stamp, i.e., τ_k , can be obtained by the integration of all previous time-intervals, i.e., $\int_0^{\tau|x_{1:k}} \lambda(\tau)d\tau$, and it is an instance of the exponential family distributions, e.g., $y = \lambda e^{-\lambda\tau}$. Their model also supports stochastic time and action predictions via sampling latent variables through a Variational Auto-Encoder (VAE) embodied as LSTMs, which, however, needs some extra effort on overcoming the **latent variable collapse issue** [161].

Although the temporal point process has been known for modelling inter-arrival events, not all researchers adopt it. Some have suggested **intensity-free** approaches for modelling such processes instead and found success [246, 203]. They argue that the intensity function is unnecessary for some sequence generation tasks and the generative model can be better trained using more advanced learning schema, e.g., generative adversarial learning. In this regard, a research direction modeled time as a real-valued output directly from regression models for joint action and time prediction, e.g., using an MLP to generate the output [156, 2, 1, 99, 58] as shown in Figure 3.5: For any next-step action, the “length of future action” and “remaining length of current action” will be regressed from fully-connected layers. In comparison to previous Poisson methods [161, 155], these approaches choose deterministic time models rather than the Poisson distribution and therefore their outputs for a given input will be fixed. An exception to this general trend instead assumes a Gaussian prior and approximates the mean and variance [1].

A third way to model temporal data is to treat time in terms of discrete categorical values. Along these lines, research on recommendation systems found that modelling time with a categorical representation and optimizing with cross-entropy consistently outperformed the use of real valued modelling [135]. It is worth noticing that the range of the discrete categorical variable normally needs to be defined beforehand. For example, in image classifications, the ImageNet dataset defines 1000 categories and thus its models (i.e., AlexNet) would output probability score vectors $\in \mathbb{R}^{1000}$. For discrete times, the maximum time index is often chosen as the time range value [168].

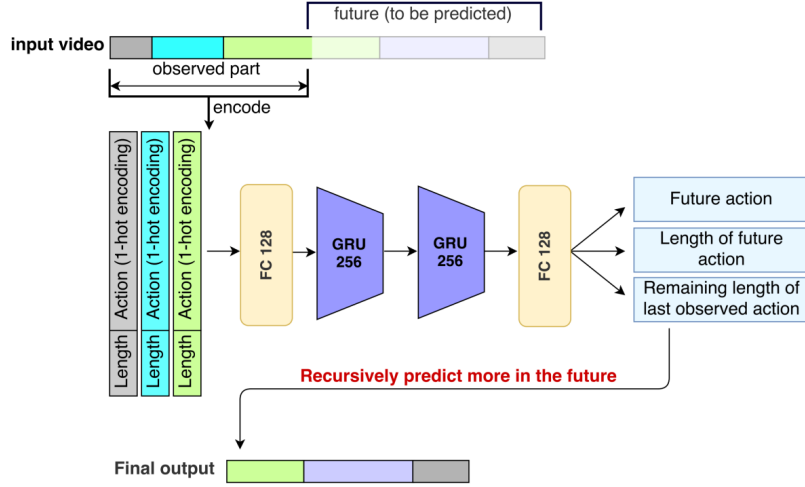


Figure 3.5: A regression approach to modeling action and time prediction [2]. Notice the observed input action is pre-processed into a one-hot encoding and the input time is framed as a real value scalar. Subsequent processing alternates use of fully connected (FC) layers and Gated Recurrent Units (GRUs). The whole scope of the future requires recursive inference. Figure reproduced with permission from [2].

3.2.3 Egocentric action prediction

First person video analysis has attracted increasing attention due to the rich information contained in egocentric visual data and easier access to wearable recording devices. As seen in Figure 3.6, egocentric filming exhibits much closer viewing angles of objects, environments and interactions of daily activity compared to third-person views (e.g., UCF101 [211]). However, another critical difference in the third person view is the lack of notable motion of actors, i.e., only the hand motion is typically visible. Moreover, since the camera moves along with its carrier, there usually exists large background motion as well as jittering in video frames.



Figure 3.6: Examples of egocentric activity footage. Top and bottom figures in a column are two different frames of the same action. Figure reproduced with permission from [34].

Missing actors in activity videos makes classic human action analysis methodologies not naturally applicable and larger view-point changes as well as camera jittering makes motion feature extraction (e.g., optical flow) very noisy. These obstacles incentivise researchers to tackle egocentric data in a slightly different way.

Related research can be grouped into two main divisions: (1) work that actively explores non-traditional visual cues (i.e., hand motion, gaze salience, object trajectory, etc.) that are especially salient in ego-view data and (2) work that follows regular techniques and treats egocentric data in a similar way as third-person view data. In this approach, even though videos are filmed under egocentric view, performed actions can be inferred without direct observation of large portions of the actor. In the following, we briefly introduce some major visual cues used by the first division.

Gaze information. One beneficial information source hidden in first person video is gaze information. Gaze moving on/off of a certain object can closely correspond to the occurrence/ending of a particular action. Much work has enjoyed the incorporation of such information (e.g., [216, 128, 204]). Yet, gaze localization does not come freely. One way to obtain gaze information is the 3D reconstruction of actors and then inferring the gaze direction from head direction and body poses [216]. Another way to gain gaze information is through manual annotation and/or use of additional instrumentation. Though expensive, it is easily doable through eye-tracking devices [204].

Some other work that considers gaze as an important visual cue focused efforts on predicting future gaze fixations from observed frames [263]. Their framework proceeded in a similar way to video frame prediction but output binary gaze heat-maps. However, the authors did not show any application of the predicted gaze on other related problems.

Hand information. In first person videos, hand movement typically is the major human visible action. Human-object interactions often consist of a movement part and one or more object parts, with “take bowl” as an example. First person videos capture complex hand movements during a rich set of interactions, thus providing a powerful source for studying the connection between hand actions and future representation.

In imitation of the acquisition of gaze information, hand motion can be either annotated or estimated online. For example, an extant hand shape detector [272] can be applied to obtain a hand mask across frames [204]. The use of the estimated hand mask is straightforward:

Overlaying the hand mask on RGB input frames. In other recent work, it is proposed to use the hand movement trajectory as an output regularizer [142]. The authors enforced the learned network model to jointly predict future actions, future hand trajectories and interactive hotspots (regions where interaction would happen with high likelihood). In an ablation study, it was shown that the three related tasks were complementary to each other.

Object information. Objects that are interacted with are another potentially important source of information for predicting future actions. While such information can be generally useful for action prediction, it can be of particular importance in first-person video where the movements of the actor may be obscured.

An early work along these lines investigated the relevance of egocentric object trajectories in the task of next-active-object prediction [54]. Provided that an object detector/tracker is available, they proposed to analyze object trajectories observed in a small temporal window to detect next-active-objects before the object-interaction is actually started.

A more recent work incorporated information on the spatial location of objects into egocentric action prediction [56]. The authors adopted a pretrained deep object detector to localize objects and merged their deep features with another two feature modalities (i.e., RGB and optical flow) captured globally from the entire frame. However, their major inference engine is a novel Rolling-Unrolling (RU) LSTM system and object features are simply used as input. In an ablation study, they found that inclusion of the object information helped achieve much better results compared to neglecting that information.

As mentioned above, there are other efforts that treat ego-data no differently from regular third person data. One such approach processed egocentric video with a standard deep neural network action recognition model pretrained on third person video datasets [34, 236], but now fine-tuned for ego-action future prediction assisted by top-k loss [55]. Other research made use of standard third person analysis tools to anticipate events that might occur in different places than those currently in view [185, 15]. For example, the current view is of the living room, while the future activity will occur in the kitchen. In response, they proposed to merge 3D point cloud location information obtained from a simultaneous localization and mapping (SLAM) system with interactive objects and high-level scene labels to do the predicting

3.3 Overall contribution: Diversified discrete GAN for joint activity and time prediction

We now detail our second major contribution, investigation of the joint anticipation of **long-term** activity labels and their corresponding times, with the aim of improving both the **naturalness** and **diversity** of predictions. We address these matters using conditional Generative Adversarial Networks for Discrete Sequences (cGAN-DS) [68]. Central to our approach is a re-examination of the unavoidable sample quality vs diversity trade-off of the recently emerged Gumbel-Softmax [70] relaxation based GANs on discrete data [91]. In response, we ameliorate this trade-off with a simple but effective sample distance regularizer. Moreover, we provide a unified approach to inference of activity labels and their times so that a single integrated optimization succeeds for both. With this novel approach in hand, we demonstrate the effectiveness of the resulting discrete sequential GAN on multimodal activity anticipation. We evaluate the approach on three standard datasets and show that it outperforms previous approaches in terms of both accuracy and diversity, thereby yielding a new state of the art in activity anticipation. Figure 3.7 provides an overview.

3.3.1 General pipeline

We seek to anticipate what will happen next from an observed initial part of a video, cf. [2, 161, 99]. We are interested in not just anticipating a single action that immediately follows the observed video, but potentially **an entire sequence of subsequent actions**. Specifically, the input contains a sequence of n observed action tokens $A_{1:n} = (\mathbf{a}_1, \dots, \mathbf{a}_n)$, with $\mathbf{a}_i$ the i^{th} token in the sequence. Actions are taken to belong to an action vocabulary of size V , i.e., $\mathbf{a}_i \in \mathbb{R}^V$, and each $\mathbf{a}_i$ is a one-hot vector action encoding. Corresponding to each action sequence, $A_{1:n}$, is a time sequence, $T_{1:n} = (\mathbf{t}_1, \dots, \mathbf{t}_n)$ with $\mathbf{t}_i$ the temporal length (e.g., in units of seconds) of the i^{th} action belonging to a duration vocabulary of size H , $\mathbf{t}_i \in \mathbb{R}^H$. Each $\mathbf{t}_i$ is a one-hot vector encoding of the duration of action at position i in the sequence, taken from the vocabulary of temporal durations.

Our temporal representation is motivated by a desire to treat encoding, decoding and adversarial learning in a consistent fashion across input, output and ground truth and will be shown to yield state-of-the-art results. In all three of the considered standard datasets, the

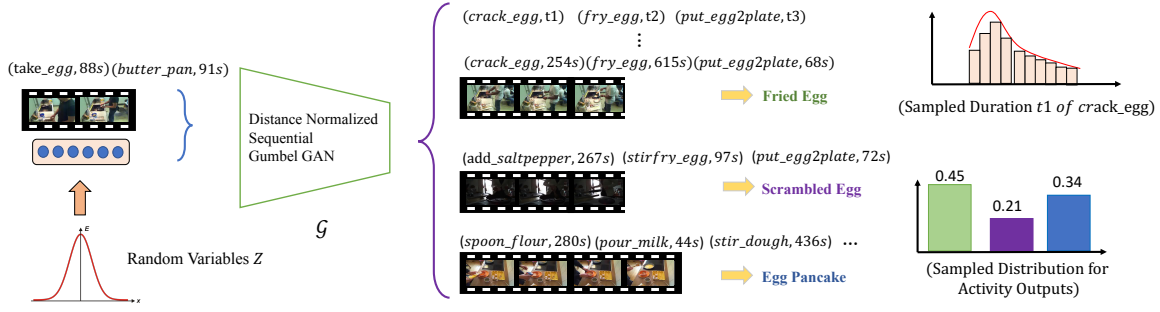


Figure 3.7: Proposed Activity Anticipation Process Overview. An initial observation of an activity, as a discrete sequence of actions and times (e.g., take_egg, 88s, etc.) and a random variable, z , input to a sequence generator, $\mathcal{G}$. The generator combines adversarial learning with Gumbel sampling and normalized distance regularization so that sampled outputs are both diverse and realistic.

time data format (e.g., seconds) is given as discrete. Previous work [1, 99, 161], using MLPs for time decoding of future activities causes difficulty in learning, given discrete ground truth. Also, recursive predictions that rely on preceding results suffer in such settings, as only discrete values are in the training data. We match our representation to the data by enforcing one-hots for prediction. It has been shown beneficial to treat such formatting as categorical under adversarial learning in allied domains [32, 120, 169].

We formulate the task as: Given such a partial sequence as an ordered pair, $\langle A_{1:n}, T_{1:n} \rangle$, we produce a distribution over subsequent action-temporal duration pairings, $\langle A_{n+1:\tau}, T_{n+1:\tau} \rangle$, up to and including time τ . Our activity anticipation pipeline has three parts: (1) an input sequential token embedding module, (2) an output sequential generator, $\mathcal{G}$, and (3) an ensemble of ConvNet based discriminators, $\mathcal{D}$, for adversarial learning, which are elaborated separately in the following sections. Prior to diving into details, we first illustrate the mode-collapse issue in discrete **Gumbel-Softmax** based GANs.

3.3.2 Gumbel-Softmax for differentiable discrete variable sampling

To enable discrete random variables sampling with differentiable **backward-gradients**, two groups (e.g., [91, 152]) simultaneously proposed using softmax relaxation on top of the Gumbel-max trick. The Gumbel-max trick [70] was first introduced to approximate the sampling operation on un-normalized categorical probability logits (e.g., multi-nomial), $\mathbf{o}$, and it is

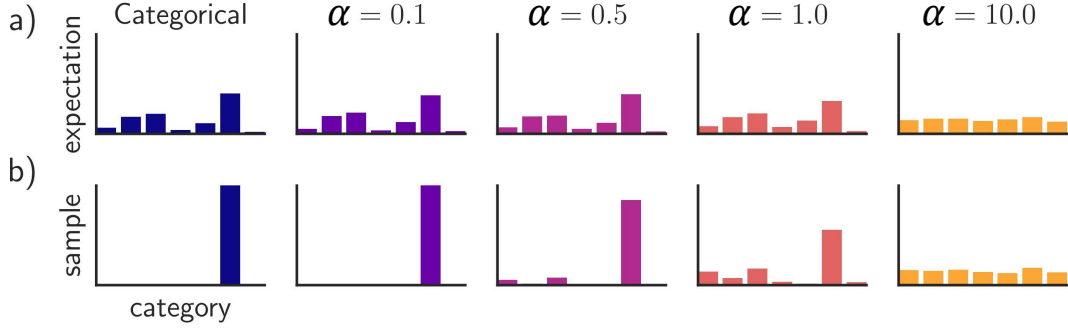


Figure 3.8: Illustration of the impact of the temperature hyper-parameter on Gumbel-Softmax function (3.4). Figure reproduced with permission from [91]. Top row shows the expectation from a collection of Gumbel samplings. The bottom row shows one single sample with respect to the temperature, α . As $\alpha \rightarrow 0$, the sample is closer to one-hot; when $\alpha \rightarrow \infty$, the sample is more akin to uniform.

formulated as

$$\mathbf{o} \sim \operatorname{argmax}(\mathbf{o} + \mathbf{g}), \quad \mathbf{g} \sim \text{Gumbel i.i.d.}, \quad (3.3)$$

where the Gumbel noise, $\mathbf{g}$, is drawn from the standard Gumbel distribution [70], which can be easily obtained via two consecutive logarithmic transformations from uniform distributions (e.g., $-\log(-\log U)$, with U drawn from $\mathcal{U}(0, 1)$).

To better work with the one-hot vector embedding design in current discrete stochastic machine learning models (e.g., VQ-VAE [182] and SeqGAN [257]), the softmax function is used as an approximation to one-hot vectors, according to

$$\text{Gumbel}(\mathbf{o}_k[i]) \equiv \frac{\exp((\log(\mathbf{o}_k[i]) + \mathbf{g}_k[i]) / \alpha)}{\sum_{j=1}^m \exp((\log(\mathbf{o}_k[j]) + \mathbf{g}_k[j]) / \alpha)}, \quad (3.4)$$

with m the dimension of unnormalized probability logits, $\mathbf{o}_k$, and $\mathbf{g}_k$ the i.i.d. Gumbel noise for logit dimension k and i indexing logit elements.

The above equation can support two essential operations: (1) It imitates the sampling procedure of categorical distributions, $\mathbf{o}_k$, through adding the logits with Gumbel noise. The distribution represented by $\mathbf{o}_k$ equals to that of samples draw from the argmax operator over perturbed logits, i.e., $\operatorname{argmax}(\mathbf{o}_k + \mathbf{g}_k)$; see [70] for mathematical proof. (2) It approximates one-hot vectors with the softmax function as well as the temperature hyper-parameter, i.e., α .

In the definition of the Gumbel operator (3.4), the α is a tunable parameter, called the

temperature, which controls the similarity of the approximated one-hot vector (3.4) to the actual one-hot vector, as shown in Figure 3.8. Intuitively, smaller α provides more accurate outputs for the discriminator, thereby leading to better sample quality. However, the associated drawback is mode-collapse [169] caused by high gradient variance regarding the temperature as

$$\frac{\partial \text{Gumbel}(\mathbf{o})}{\partial \mathbf{o}} \propto \frac{1}{\alpha^2}. \quad (3.5)$$

It is seen that when $\alpha \rightarrow 0$, the backward gradient will explode. To combat this situation, one can adopt a temperature annealing strategy to encourage diversity [152, 91], which is heuristic in terms of how much and when the temperature should be annealed. Some others attempt applying an entropy loss on logits to encourage more exploration [68]. Instead, we make use of the recently proposed Normalized Distance Regularizer (NDR) [254] to augment (3.4). In the next subsection we illustrate the utility of the NDR, but delay its formal definition to subsection 3.3.6 where we incorporate it into our activity prediction model.

3.3.3 Mode collapse in discrete GANs

Most likely due to the high-variance of updating gradients, GAN training is prone to failing on capturing multi-modal distributions of the training data, which is known as the **mode-collapse** issue. In the continuous domain (e.g., GAN for image pixel intensity generation), researchers typically reveal and examine this notorious issue using some simple synthetic datasets to provide a sanity check. Popular test datasets for such experiments include a “Gaussian Grid” or “Gaussian Circle”. Consideration of such datasets allows the effectiveness of the approach to be verified in relative isolation of other issues and thereby focus on the particular abilities to combat mode collapse. With the effectiveness demonstrated in these simplified scenarios, the approach is then applied to more complicated real-life datasets and tasks. Yet, no analogous experiment has appeared for the *discrete domain*, especially not for discrete GANs.

Here, we provide a novel experiment setting for verifying mode-collapse mitigation in the discrete domain (e.g., one-hot vector generation) and use it to evaluate our approach. Similar to above continuous Gaussian examples, we synthesize a dataset with 8 modes located on a (5×5) 2D grid at (x, y) positions, namely (1, 1), (1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2), (3, 3). Each mode is given as a unit impulse shifted to one of the specified locations. The

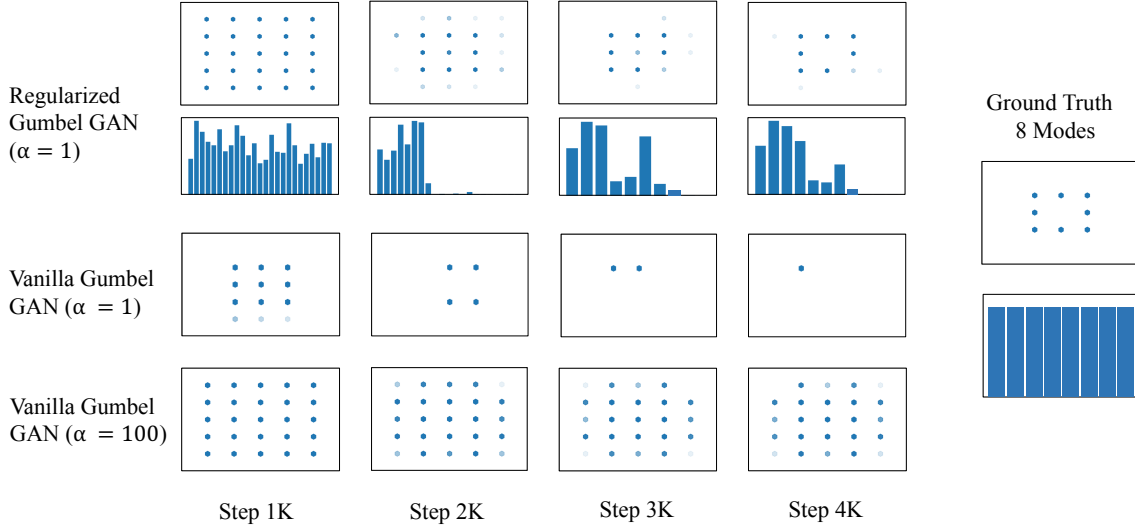


Figure 3.9: Simplified demonstration of the quality *vs.* diversity trade-off with Gumbel softmax relaxation (3rd and 4th rows) and the improved result provided by the proposed regularizer (1st and 2nd rows). The first four columns show the evolution of results across training. The far right column shows ground truth. Each bar chart in the 2nd row displays the numeric distribution of modes after every 1K training steps, which ideally should resemble the ground-truth bar chart (uniformly distributed among eight modes), last row of the far right column.

rightmost column of Figure 3.9 (e.g., “Ground Truth eight Modes”) provides an illustration.

To simulate a stochastic conditional generation environment, we construct a simple discrete GAN with a two-layer multilayer perceptron (MLP) as generator, that maps the input, $(0.5, 0.5)$, to two logit vectors, that represent the generated (x, y) coordinates. In complement, a separate two-layer MLP is used as a discriminator to provide an adversarial loss for optimization. Standard Gaussian noise is injected into the model during the generative process to provide stochastic capacity. During training, Gumbel-Softmax relaxation, is used to sample from learned multi-nomial distributions and obtain the approximated one-hot vector. We compare results for optimization that solely works under a pure adversarial loss vs results for optimization that augments the adversarial loss with the normalized distance regularizer. Figure 3.9 shows three sets of experimental results to help understand the quality/diversity trade-off issue in discrete GANs and especially to demonstrate the effectiveness of the proposed NDR regularizer in mitigating mode collapse.

Vanilla Gumbel discrete GAN (Gumbel GAN without the normalized distance regularizer) suffers from low diversity with low temperature (e.g., for $\alpha = 1$, only a single mode is

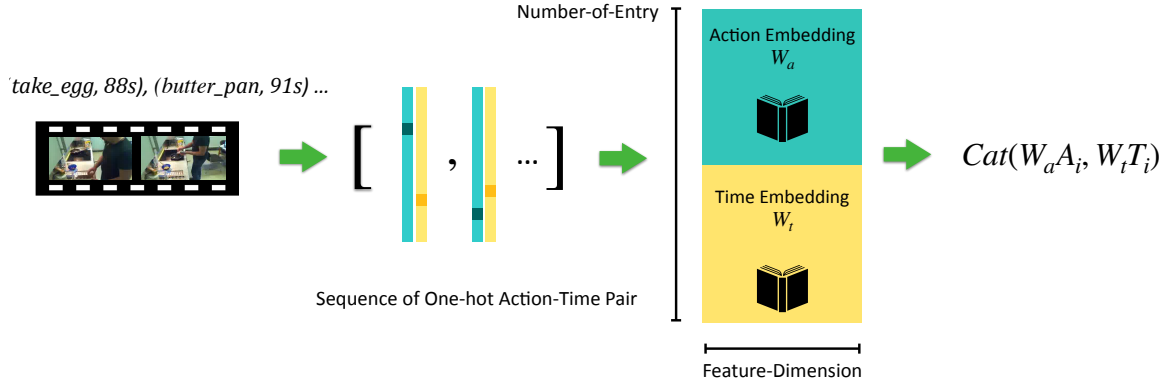


Figure 3.10: Illustration of the dual token embedding process. Both the action labels and times of partial videos, in one-hot vector format, are embedded through a dictionary. Then, embedded feature vectors are concatenated, denoted by $Cat(\cdot, \cdot)$, together as input to the sequence generation module.

preserved) and low quality with high temperature (e.g., for $\alpha = 100$, we find the presence of false modes along with the correct ones). Overall, either choice leads to undesirable outcomes. In response, the majority of recent alternative approaches seek to find a reasonable balance between quality and diversity, which can be expensive and heuristic. Instead, we augment the vanilla approach with the normalized distance regularizer and find that it supports both high quality and diversity, while mitigating mode collapse. This ability is documented in the 4th column of the 1st row of Fig 3.9, where all eight modes are captured much better than when the regularizer is lacking (even though not all modes are captured equally).

3.3.4 Dual token embedding

As widely discussed in the NLP and image/video captioning literatures, the expressive power of raw discrete tokens can be greatly enhanced through dictionary embeddings in higher dimensional continuous feature spaces, cf. [2, 99, 161]. To this end, each entry of the initial pairing of action and temporal tokens, $\mathbf{a}_i$ and $\mathbf{t}_i$, are projected to higher dimension continuous spaces, $\mathbb{R}^a$ and $\mathbb{R}^t$, resp., via dictionary mapping matrices W_a and W_t . Here, W_a , of dimension $\mathbb{R}^a \times V$, is the mapping for actions, while W_t , of dimension $\mathbb{R}^t \times H$, is the mapping for times. The embedding matrices are constructed such that their j^{th} columns are the mappings for actions and times indicated by the one-hot vector encodings of actions and times. (Note

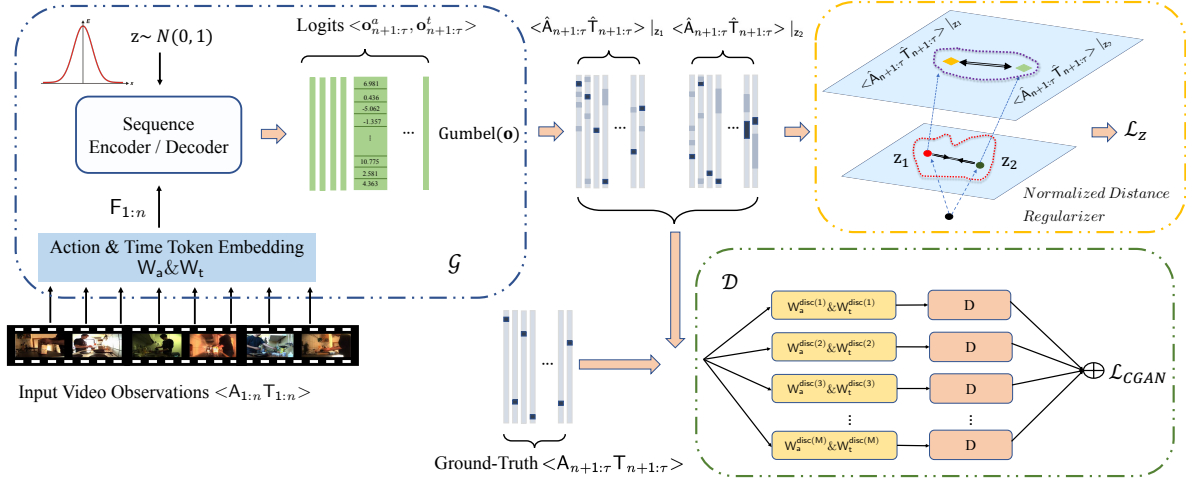


Figure 3.11: Depiction of our diversity improved discrete sequential GAN architecture. For both training and inference, input action and time duration token sequences $\langle A_{1:n} T_{1:n} \rangle$ are embedded via learnable matrices W_a and W_t , (3.6), to generate a compact feature representation, $F_{1:n}$. $F_{1:n}$ is processed iteratively along with latent noise signals, z , by an encoder (3.7) and decoder (3.8) (RNN or LSTM) along with a linear mapping, (3.9), to produce logits $\langle o_{n+1:\tau}^a, o_{n+1:\tau}^t \rangle$. At inference time (not shown), the logits are simply sampled with $\arg \max$, (3.10), to produce predicted activity sequences. At training time (shown), Gumbel sampling, (3.4), is performed to facilitate gradient-based learning. Pairs of samples, $\langle \hat{A}_{n+1:\tau} \hat{T}_{n+1:\tau} \rangle|_{z_1}$ and $\langle \hat{A}_{n+1:\tau} \hat{T}_{n+1:\tau} \rangle|_{z_2}$, produced with different latents, z_1 and z_2 , are compared under a normalized distance regularizer, (3.16), to define a loss, $\mathcal{L}_z$, that encourages a generator capable of diverse outputs. In complement, a sampled output is adversarially compared, (3.17), to ground truth, $\langle A_{n+1:\tau} T_{n+1:\tau} \rangle$, by a discriminator, $\mathcal{D}$, under multiple embeddings, $W_a^{\text{Disc}(i)}, W_t^{\text{Disc}(i)}$, to define an additional loss, $\mathcal{L}_{CGAN}$, that encourages a generator capable of realistic outputs.

that for a given observation pair, $\langle a_i, t_i \rangle$, j generally is different for the action and time.) Our overall embeddings are then given as

$$\mathbf{f}_i = \text{Cat}[W_a \mathbf{a}_i, W_t \mathbf{t}_i], \quad (3.6)$$

with $\mathbf{f}_i$ associating action and temporal features at the i^{th} index via concatenation of their respective embedding vectors; see Figure 3.10. Analogous to the input sequences, $A_{1:n}$ and $T_{1:n}$, we define embedded sequences, $F_{1:n} = (\mathbf{f}_1, \dots, \mathbf{f}_n)$.

3.3.5 Sequence generation

For computing outputs, $\langle A_{n+1:\tau}, T_{n+1:\tau} \rangle$, given a sequence of input feature embeddings, $F_{1:n}$, we use the *seq2seq* generator [220]. To lend insight into the generality of our approach, we consider two different backbones, RNN and LSTM, as instantiating mechanisms for the generator and results for both are reported in Section 3.4.3. While this design choice might be replaced by additional alternatives (*e.g.*, a temporal convolutional network (TCN) [170, 35] or relational memory [199, 169]), RNN and LSTM are general sequence learning structures also adopted by our main comparison approaches [2, 1]. In the following, we make use of a general notation such that *Sequence* stands for either RNN or LSTM.

Sequence generation proceeds by sequentially encoding and decoding the embedded features, followed by mapping to logits and sampling. More specifically, the components of the input feature sequence, $(f_1, \dots, f_n)$, are iteratively injected into an encoder, $Sequence^{enc}$, to obtain a fixed dimensional representation $\mathbf{v}$, given by the last hidden state of $Sequence^{enc}$, according to

$$\mathbf{h}_k^{enc} = Sequence^{enc}(\mathbf{f}_k, \mathbf{h}_{k-1}^{enc}), 1 < k < n \text{ and } \mathbf{v} = \mathbf{h}_n^{enc}, \quad (3.7)$$

$\mathbf{h}_k^{enc}$ the k^{th} hidden encoder state and the initial hidden state, $\mathbf{h}_0$, randomly set.

Next, a decoding sequence model, $Sequence^{dec}$, whose initial hidden state is set to $\mathbf{v}$ and first input variable to the last observation $\mathbf{f}_n$, is used for producing output hidden state sequence $(\mathbf{h}_{n+1}^{dec}, \dots, \mathbf{h}_\tau^{dec})$ according to

$$\mathbf{h}_{k+1}^{dec} = Sequence^{dec}(\mathbf{f}_k, \mathbf{h}_k^{dec}), n < k < \tau - 1. \quad (3.8)$$

Then, a pair of linear transformations defined by matrices W_o^ϕ and vectors $\mathbf{b}_o^\phi$, $\phi \in \{a, t\}$, maps hidden states, $\mathbf{h}_k^{dec}$, to logits according to

$$\left. \begin{aligned} \mathbf{o}_k^a &= W_o^a \mathbf{h}_k^{dec} + \mathbf{b}_o^a \\ \mathbf{o}_k^t &= W_o^t \mathbf{h}_k^{dec} + \mathbf{b}_o^t \end{aligned} \right\} \quad n + 1 \leq k \leq \tau, \quad (3.9)$$

where $\mathbf{o}_k^a$ and $\mathbf{o}_k^t$ are vectors representing the logits for each entry of action or time vocabulary, *i.e.*, they are of dimensions $\mathbb{R}^a \times V$ and $\mathbb{R}^t \times H$, resp.

Finally, the logits, (3.9), are sampled in one of two ways, depending on whether the system is operating in inference or training mode. At inference time, traditional discrete data sampling is used to select the most likely probability index, which is then reformatted as one-hot vectors according to

$$\left. \begin{aligned} \mathbf{a}_k &= \text{one_hot} \left(\arg \max_i (\mathbf{o}_k^a[i]) \right) \\ \mathbf{t}_k &= \text{one_hot} \left(\arg \max_i (\mathbf{o}_k^t[i]) \right) \end{aligned} \right\} \quad n+1 \leq k \leq \tau, \quad (3.10)$$

with $\mathbf{o}_k^a[i]$ and $\mathbf{o}_k^t[i]$ selecting the i^{th} element of $\mathbf{o}_k^a$ and $\mathbf{o}_k^t$, resp.

Unfortunately, the $\arg \max$ operator incurs zero derivative with respect to parameters of operations coming before it, i.e., in the present context those of the embedding matrices and generator, which interferes with gradient-based training. Therefore, to obtain differentiable one-hot vectors from logits $\mathbf{o}_k$ during training, we adopt the Gumbel-Softmax relaxation technique [91, 152] that mimics one-hot vectors from categorical distributions. In particular, we replace (3.10) with Gumbel sampling, (3.4), which yields the output discrete tokens according to

$$\left. \begin{aligned} \mathbf{a}_k[i] &= \text{Gumbel}(\mathbf{o}_k^a[i]) \\ \mathbf{t}_k[i] &= \text{Gumbel}(\mathbf{o}_k^t[i]) \end{aligned} \right\} \quad n+1 \leq k \leq \tau. \quad (3.11)$$

For both sampling techniques (3.10) and (3.11), for each iteration on k , we use

$$\mathbf{f}_k = \text{Cat}[\mathbf{W}_a \mathbf{a}_k, \mathbf{W}_t \mathbf{t}_k], \quad n+1 \leq k \leq \tau \quad (3.12)$$

in the encoding, (3.7), to reinitialize subsequent decoding (3.8), logits mapping (3.9), and sampling (3.10) or (3.11), to give the next generation of action/time tokens. Having produced $\mathbf{a}_k$ and $\mathbf{t}_k$ for $n+1 \leq k \leq \tau$ via (3.11), we define $\mathbf{A}_{n+1:\tau} = (\mathbf{a}_{n+1}, \dots, \mathbf{a}_\tau)$, $\mathbf{T}_{n+1:\tau} = (\mathbf{t}_{n+1}, \dots, \mathbf{t}_\tau)$ and ultimately $\langle \mathbf{A}_{n+1:\tau}, \mathbf{T}_{n+1:\tau} \rangle$ as output.

Overall, the generating process, $\mathcal{G}$, of our approach entails sequential application of the embedding (3.6), encoding (3.7), decoding (3.8), logits mapping (3.9) and sampling, (3.10) or (3.11), to each element of the input, $\langle \mathbf{A}_{1:n}, \mathbf{T}_{1:n} \rangle$, formalized as

$$\langle \mathbf{A}_{n+1:\tau}, \mathbf{T}_{n+1:\tau} \rangle = \mathcal{G}(\langle \mathbf{A}_{1:n}, \mathbf{T}_{1:n} \rangle, z; \theta_g), \quad (3.13)$$

where θ_g includes learnable parameters of Sequence^{enc} , Sequence^{dec} , $\mathbf{W}_\phi$, $\mathbf{W}_o^\phi$, $\mathbf{b}_o^\phi$, $\phi \in \{a, t\}$,

and z is a randomized input that encourages diversity during sampling; see Sec 3.3.6. From a probabilistic view, the overall sequence generation procedure can be seen to operate as a Markov process,

$$p(A_{n+1:\tau}, T_{n+1:\tau} | A_{1:n}, T_{1:n}) = \prod_{k=n+1}^{\tau} p(a_k, t_k | v, f_n, h_{n+1}, \dots, h_{k-1}). \quad (3.14)$$

3.3.6 Diversified discrete adversarial learning

To get diversity in generated sequences, we use a Normalized Distance Regularizer [254, 143], as motivated in Subsection 3.3.3. This regularizer operates via pairwise distances between various outputs from the same input according to

$$\max_{\mathcal{G}} \mathcal{L}_z(\mathcal{G}) = \mathbb{E}_{z_1, z_2} \left[\min \left(\frac{\|\mathcal{G}(Z, z_1) - \mathcal{G}(Z, z_2)\|}{\|z_1 - z_2\|}, \tau \right) \right], \quad (3.15)$$

where Z is input (*e.g.*, in our case, an observed action-time sequence pair, $\langle A_{1:n}, T_{1:n} \rangle$), $\mathbb{E}[\cdot]$ is the expectation operator, $\|\cdot\|$ is a norm, $z_i \sim N(0, 1)$ is a random latent variable that converts the deterministic model into a stochastic one and τ is a margin for thresholding the minimum distance. The normalized distance regularizer, (3.15), operates such that the denominator, $\|z_1 - z_2\|$, encourages the two random codes to be close in latent space, while the numerator, $\|\mathcal{G}(Z, z_1) - \mathcal{G}(Z, z_2)\|$, encourages the outputs to be distant in the output space. The intuition is to encourage $\mathcal{G}$ to visit more important modes given an input variable, but with minimum traveling effort in the sampling space. In this way, distinctive outcomes are circumscribed within a tighter latent space, so that the model is less likely to ignore some modes.

In application to our case, the overall generated sequence $\langle A_{n+1:\tau}, T_{n+1:\tau} \rangle = \mathcal{G}(\langle A_{1:n}, T_{1:n} \rangle, z)$, is used to calculate the l_1 distance between samples,

$$\mathcal{L}_z(\mathcal{G}) = \mathbb{E}_{z_1, z_2} \left[\frac{\|\mathcal{G}(\langle A_{1:n}, T_{1:n} \rangle, z_1) - \mathcal{G}(\langle A_{1:n}, T_{1:n} \rangle, z_2)\|_{l_1}}{\|z_1 - z_2\|} \right], \quad (3.16)$$

where $\|\cdot\|_{l_1}$ denotes use of the l_1 norm; explicit dependence of $\mathcal{G}$ on θ is suppressed for conciseness.

Let $\langle \hat{A}_{n+1:\tau}, \hat{T}_{n+1:\tau} \rangle_{|z_1}$ and $\langle \hat{A}_{n+1:\tau}, \hat{T}_{n+1:\tau} \rangle_{|z_2}$ be two samples generated by our genera-

tor, $\mathcal{G}$, from input $\langle \mathbf{A}_{1:n}, \mathbf{T}_{1:n} \rangle$ with two different latent noise signals, z_1 and z_2 , respectively. Then, Equation (3.16) is realized by calculating normalized l_1 distances separately between the actions and temporal durations and then combining through averaging according to

$$\begin{aligned}
& \|\mathcal{G}(\langle \mathbf{A}_{1:n}, \mathbf{T}_{1:n} \rangle, z_1) - \mathcal{G}(\langle \mathbf{A}_{1:n}, \mathbf{T}_{1:n} \rangle, z_2)\|_{l_1} \\
&= \|\langle \hat{\mathbf{A}}_{n+1:\tau}, \hat{\mathbf{T}}_{n+1:\tau} \rangle|_{z_1} - \langle \hat{\mathbf{A}}_{n+1:\tau}, \hat{\mathbf{T}}_{n+1:\tau} \rangle|_{z_2}\|_{l_1} \\
&= \frac{1}{2} \left(\|\hat{\mathbf{A}}_{n+1:\tau}|_{z_1} - \hat{\mathbf{A}}_{n+1:\tau}|_{z_2}\|_{l_1} + \|\hat{\mathbf{T}}_{n+1:\tau}|_{z_1} - \hat{\mathbf{T}}_{n+1:\tau}|_{z_2}\|_{l_1} \right) \\
&= \frac{1}{2} \left(\frac{1}{\tau - n} \sum_{i=1}^{\tau-n} \|\hat{\mathbf{a}}_i|_{z_1} - \hat{\mathbf{a}}_i|_{z_2}\|_{l_1} \right. \\
&\quad \left. + \frac{1}{\tau - n} \sum_{i=1}^{\tau-n} \|\hat{\mathbf{t}}_i|_{z_1} - \hat{\mathbf{t}}_i|_{z_2}\|_{l_1} \right).
\end{aligned}$$

It appears that our solution to preserving both quality and diversity in discrete sequence generation is novel.

3.3.7 Ensembled discriminator

To train our generator, $\mathcal{G}$, adversarially, we refer its output to a discriminator, $\mathcal{D}$. Successful examples of deep discriminators for sequence generation include DNNs, ConvNets and RNNs [228, 100, 120], amongst which incorporation of a ConvNet classifier with an ensemble of multiple embeddings has achieved top performance [169]. Notably that approach has shown an ability to dispose of discriminator pretraining. We adopt that approach.

Given a generated output sequence, $\langle \hat{\mathbf{A}}_{n+1:\tau}, \hat{\mathbf{T}}_{n+1:\tau} \rangle$, and real (i.e., ground truth) sequence, $\langle \mathbf{A}_{n+1:\tau}, \mathbf{T}_{n+1:\tau} \rangle$, we embed the elements of the sequences analogous to the embeddings of Section 3.3.4, and execute our discriminator, $\mathcal{D}$. Let W_a^{Disc} and W_t^{Disc} , be discriminator embedding matrices, distinct from, but analogous to the embedding matrices used earlier, (3.6). Then, the process unfolds in three steps. First, for each pair $\langle \mathbf{a}_i, \mathbf{t}_i \rangle, i \in \{n+1, \tau\}$, and $\langle \hat{\mathbf{a}}_i, \hat{\mathbf{t}}_i \rangle, i \in \{n+1, \tau\}$ we produce embedded vectors $\mathbf{f}_i^{\text{Disc}}$ and $\hat{\mathbf{f}}_i^{\text{Disc}}$ by replacing W_a and W_t with W_a^{Disc} and W_t^{Disc} , resp. in (3.6). Second, the complete sets of produced embeddings $\mathbf{f}_i^{\text{Disc}}$ and $\hat{\mathbf{f}}_i^{\text{Disc}}$ are converted into matrices, $F_{n+1:\tau}^{\text{Disc}}$ and $\hat{F}_{n+1:\tau}^{\text{Disc}}$, whose i^{th} columns are $\mathbf{f}_i^{\text{Disc}}$ and $\hat{\mathbf{f}}_i^{\text{Disc}}$, resp. Third, the resulting $F_{n+1,\tau}$ and $\hat{F}_{n+1:\tau}$ are processed akin to 2D images with a discriminator ConvNet, $\mathcal{D}$, consisting of multiple convolutional, nonlinear activation

and max-pooling layers with a fully connected layer at the top to produce a label, i.e., 1 versus 0 for real versus generated.

Finally, to account for the desired multiple embeddings, the entire three step process is repeated M times with distinct embeddings, $W_a^{\text{Disc}(i)}$ and $W_t^{\text{Disc}(i)}$, $i \in \{1, \dots, M\}$, and the final adversarial loss from $\mathcal{D}$ comes by averaging across the results of those embeddings according to

$$l_{\mathcal{D}} = \frac{1}{M} \sum_{i=1}^M \log(1 - \mathcal{D}(\hat{F}_{n+1,\tau}^{\text{Disc}(i)}, 0; \theta_d)) + \log(\mathcal{D}(F_{n+1,\tau}^{\text{Disc}(i)}, 1; \theta_d)), \quad (3.17)$$

with θ_d the learnable parameters of the M matrix embeddings and $\mathcal{D}$.

Combined objective Our learning goal is to generate samples that are both realistic (high quality) and diverse. To this end, we merge the standard adversarial learning formula (for quality)

$$\mathcal{L}_{CGAN}(\mathcal{G}, \mathcal{D}; \theta_g, \theta_d) = \mathbb{E}_y[\log \mathcal{D}(y, 1; \theta_d)] + \mathbb{E}_{x,z}[\log(1 - \mathcal{D}(\mathcal{G}(x, z; \theta_g), 0; \theta_d))], \quad (3.18)$$

which in our case specializes to $l_{\mathcal{D}}$, (3.17), with the normalized distance regularizer, (3.16), for diversity, to consider

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \lambda_1 \mathcal{L}_{CGAN}(\mathcal{G}, \mathcal{D}; \theta_g, \theta_d) - \lambda_2 \mathcal{L}_z(\mathcal{G}, \theta_g), \quad (3.19)$$

which we optimize over θ_g and θ_d according to methods documented under implementation details. The overall learning process is summarized in Figure 3.11.

3.4 Empirical evaluation

3.4.1 Evaluation protocol

We evaluate on three standard datasets: Breakfast [117], 50Salads [214] and Epic-Kitchens [34].

Breakfast Dataset. The Breakfast dataset contains 1,712 videos of 52 actors making breakfast using 48 fine-grained actions. We follow the data processing strategy used elsewhere for ConvNet modeling [2], which generates four sets of training examples by using the first 10%,

20%, 30% and 50% of the video, resp., as observation and the rest of the video as ground truth for prediction evaluation. This procedure yields 7305 training samples and 253 testing samples. We report average accuracy across four splits, as elsewhere [117].

50Salads Dataset. The 50Salads dataset has 50 videos of 25 subjects, each making two salads using 17 fine grained actions. The same processing used for the Breakfast dataset is adopted here, which leads to 160 training samples and 40 testing samples. We perform five-fold cross-validation for evaluation, using previous splits [124].

Epic-Kitchens Dataset. The Epic-Kitchens dataset contains 1st-person videos of 32 actors in 32 kitchens performing unscripted daily activities. There are 272 training videos recorded from 28 subjects, with the rest used for testing. In total there are 128 actions. We follow the preprocessing strategy used previously that randomly split the training video into 7 subsets, each containing four subjects [99]. The final result is averaged across all subsets. For each train and test input, an arbitrary clip is sampled by applying temporal windows of 30 seconds and the following 60 seconds is taken as ground-truth prediction.

Ground truth time format. We conform to previous temporal formats [2, 161]. For Breakfast and 50Salads, we use temporal intervals in units of seconds as observed in the original videos. For example, in the minutes long *egg_cake* video from Breakfast, the 3rd action in the vocabulary is *cracking_egg* and occupies 15s, which we correspondingly represent as (3, 15). Accordingly, the column entry size of the time embedding matrix, W_t , is set to the maximum action length across the entire dataset, which is 5791s for Breakfast and 4149s for 50Salads, by our counting. For Epic-Kitchens, time units are also in seconds. Random temporal crops of 90s are extracted from the videos, with the first 30s taken as observation and the remaining 60 for prediction. So, the column size of W_t is 90, [99]. We set the output length to 25 and 20 for Breakfast and 50Salads, resp., based on the maximum sample length in each dataset and pad zeros for shorter samples. The generated length is 60 for Epic-Kitchen by default [99].

3.4.2 Implementation details

Network configurations. We set the embedding dimension for input actions and times to be 32 while the hidden states of $Sequence^{enc}$ and $Sequence^{dec}$ to have dimension 128. The number of embeddings for the discriminator is $M=64$, each of dimension 32. For noise, z ,

we sample a 32-dimensional random latent variable from the $N(0, 1)$ distribution, [91] and concatenate it to the input at each time step of the generator, $\mathcal{G}$, [254]. For the ConvNet based discriminator, $\mathcal{D}$, four 2D convolutional layers with 2×2 kernels are followed by ReLU [63] and max-pooling with factor of two subsampling. A fully connected layer is appended on top for 1 versus 0 classification for ground truth/real versus generated, resp. These configurations are applied identically for both RNN and LSTM backbones.

Training. Sequential GANs suffer from training difficulties that usually necessitates pre-training with maximum likelihood estimation (MLE) for both generator and discriminator [257, 45, 71]. Recently, RelGAN [169] successfully avoided discriminator pretraining, while still relying on it for the generator under the Gumbel GAN structure. We also found pretraining unnecessary for our discriminator, but still begin with generator pretraining using MLE. Both pretraining and adversarial training employ the Adam optimizer [101]. Mini-batch-Stochastic Gradient Descent is used with a learning rate of $1e^{-2}$ for pre-training and $1e^{-4}$ for adversarial training, both with exponential decay of $1e^{-5}$. Empirically, we set λ_1, λ_2 to a ratio of 1:1. For both training and testing we input ground-truth tokens (action and time labels), with subsequent ground truth beyond the input (i.e., prediction) used for training only, as with the compared results [2, 99, 161]. A NVIDIA TITAN X GPU is used for model training.

3.4.3 Comparisons to alternative approaches

Breakfast dataset protocol 1. When evaluating on this dataset, we first conduct experiments under the setting where partial observation video is obtained by arbitrarily cutting from the entire sequence [2, 99, 1]. Results are reported for combinations of observation (20% and 30%) versus prediction (10%, 20%, 30% and 50%). Table 3.1 shows comparisons. As our model captures uncertainty via use of randomized latents, we report two evaluation metrics, average and maximum (i.e., mean or max accuracy across multiple samples, 16 for results shown).

It is seen that our approach using RNN or LSTM outperforms the alternatives in almost all settings by an average of 5–8% accuracy under the average metric, with the exception being the most extreme case, i.e., 50% prediction from 20% observation, where TOS-Dense [99] performs slightly better (by merely 0.16%) than our RNN backbone; however, our LSTM backbone still outperforms by 0.96%. Given more information to start with, as in 30% obser-

Observation	20%				30%			
Prediction	10%	20%	30%	50%	10%	20%	30%	50%
RNN [2]	0.6035	0.5044	0.4528	0.4042	0.6145	0.5025	0.4490	0.4175
CNN [2]	0.5797	0.4912	0.4403	0.3926	0.6032	0.5014	0.4518	0.4051
TOS-Dense [99]	0.6446	0.5627	0.5015	0.4399	0.6595	0.5594	0.4914	0.4423
RNN-HMM (Avg) [1]	0.5039	0.4171	0.3779	0.3278	0.5125	0.4294	0.3833	0.3307
RNN-HMM (Max) [1]	0.7884	0.7284	0.6629	0.6345	0.8200	0.7283	0.6913	0.6239
Ours-RNN (Avg)	0.7101	0.6200	0.5421	0.4383	0.7304	0.7053	0.6257	0.5119
Ours-RNN (Max)	0.8076	0.7010	0.6649	0.6283	0.8210	0.7539	0.7134	0.6251
Ours-LSTM (Avg)	0.7222	0.6240	0.5622	0.4595	0.7414	0.7132	0.6530	0.5238
Ours-LSTM (Max)	0.8208	0.7059	0.6851	0.6406	0.8336	0.7685	0.7213	0.6406

Table 3.1: Breakfast Dataset Protocol 1 results with dense anticipation mean over classes (MoC) accuracy. Avg stands for averaged results across 16 samplings, while Max stands for taking the best result among 16 samples. Best Avg is highlighted with bold font.

vation, both of our models uniformly surpass the others. The biggest difference between our instantiations reside in remote predictions, i.e., 30% and 50%, affirming that LSTM generally is better than RNN in long-term performance [99]

Compared with another stochastic approach, RNN-HMM [1], our RNN model achieves similar performance on the **Max** metric, while notably excelling on the **Avg** metric. Under both metrics our LSTM model is the top performer. Arguably, our model uses a similar base architecture as RNN-HMM [2] and is less complicated than TOS-Dense [99], which introduced skip-connections and attention, yet achieves better results. We attribute this pattern to two reasons: 1) Our approach avoids error propagation coming with iterative long-term generation [2, 1], where anticipation errors in early stages are propagated to the end. 2) Our adversarial learning focuses on the quality of overall generated sequences, rather than any single prediction. The discriminator pushes the generator to produce outputs that are more realistic as a whole.

Breakfast dataset protocol 2. To further demonstrate our strength in capturing uncertainty, we present an experiment using the protocol from Action Point Process (APP-VAE) [161], which aims to capture stochastics of single next action anticipation, but with a variational inference structure. In this scenario, the input sequence is obtained by clipping the whole video at the exact end point of random actions. Log-Likelihood (LL) is used as a metric for measuring the approximated posterior via standard importance sampling [161]. Our method is accommodated to this evaluation by selecting the immediate next action from the generated

Model	Loss	stoch. var.	accuracy	MAE	LL
TD-LSTM [161]	CE + MSE	-	53.64	173.76	-
APP-LSTM [161]	CE + NLL	-	61.39	152.17	-6.668
RNN-HMM [1]	CE + MSE	✓	57.80	-	-
APP-VAE− [161]	NLL+KL	✓	27.09	270.75	≥ -9.427
APP-VAE+ [161]	NLL+KL	✓	62.20	142.65	≥ -5.944
Ours-RNN	Adv+NormDist	✓	68.46	87.58	$\geq$ -4.851
Ours-LSTM	Adv+NormDist	✓	68.96	87.07	$\geq$ -4.836

Table 3.2: Breakfast Dataset Protocol 2 results with various learning objectives and evaluation metrics. For training objectives listed in the second column, “CE ” stands for cross-entropy, “MSE” for mean square error and “NLL” for negative log-likelihood. Note that larger accuracy is better, while smaller MAE is better. For LL, smaller absolute value is better. APP-VAE+ and APP-VAE− refer to APP-VAE with and without a learned prior, resp. For the evaluation metrics in the last two columns,

sequence. To report a lower bound LL, we chose the worst score (largest absolute value) from multiple runs. Results are shown in Table 3.2.

Both our models achieve better LL compared to the previous best of -5.944, with our LSTM at -4.836 slightly outperforming our RNN, after averaging inference results from test sets. Since this protocol focuses on a single next action, results from our two backbones are close. We also calculate the Mean Classification Accuracy over the entire anticipated sequence for action category performance and find our results outperform APP-VAE by $\approx 7\%$. For temporal duration estimation, we use Mean Absolute Error (MAE), i.e., the averaged l_1 distance between ground truth and predictions, as the evaluation metric and find that our approach reduces the time length estimation error by $\approx 40\%$.

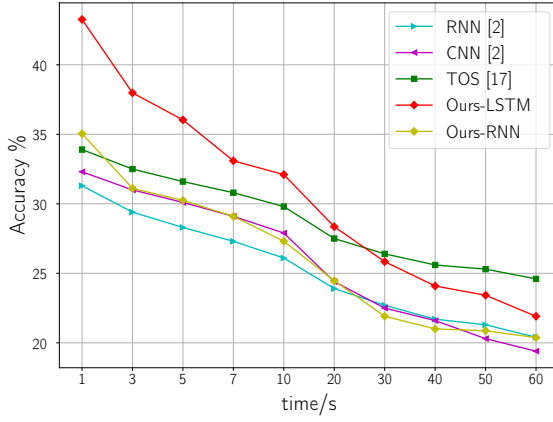
50 Salads dataset. We adopt the same experimental design as for Breakfast Protocol 1; results are in Table 3.3. As in the previous experiment, both our models achieve better average performance than the alternatives for most cases, but slightly worse at the 30% observation versus 10% prediction ratio (LSTM worse by merely 0.27%). When it comes to 30% and 50% prediction scales, our models hold a larger lead of $> 5\%$. Comparably, the CNN approach [2] also anticipates the whole sequence in one-shot but does so less effectively. A critical difference from ours is that they optimize category and temporal length with the MSE loss, which is known to produce blurred outcomes [67]. Instead, we directly work in the discrete domain with adversarial learning for sharper outcomes, which provides further support for our discrete representation of time.

Observation	20%				30%			
Prediction	10%	20%	30%	50%	10%	20%	30%	50%
RNN [2]	0.4230	0.3119	0.2522	0.1682	0.4419	0.2951	0.1996	0.1038
CNN [2]	0.3608	0.2762	0.2143	0.1548	0.3736	0.2478	0.2078	0.1405
TOS-Dense [99]	0.4512	0.3323	0.2759	0.1727	0.4640	0.3480	0.2524	0.1384
RNN-HMM (Avg) [1]	0.3495	0.2805	0.2408	0.1541	0.3315	0.2465	0.1884	0.1434
RNN-HMM (Max) [1]	0.7489	0.5875	0.4607	0.3571	0.6739	0.5237	0.4673	0.3664
Ours-RNN (Avg)	0.4571	0.3517	0.3182	0.2095	0.4597	0.3591	0.3073	0.1746
Ours-RNN (Max)	0.4903	0.3998	0.3596	0.2537	0.4926	0.4582	0.3572	0.2646
Ours-LSTM (Avg)	0.4663	0.3562	0.3191	0.2137	0.4613	0.3637	0.3310	0.1945
Ours-LSTM (Max)	0.5150	0.3845	0.3606	0.2762	0.5079	0.4754	0.3783	0.2908

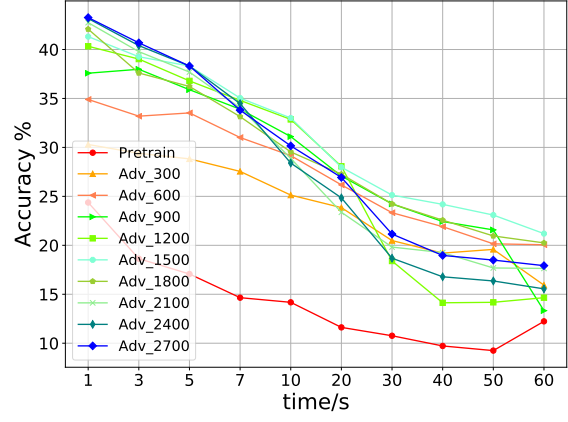
Table 3.3: Results for the 50Salads dataset with dense anticipation mean over classes (MoC) accuracy.

If we choose the best results across multiple samples, our accuracy is further boosted by $\approx 6.5\%$, which while an improvement over mean accuracy, is not as large as for Breakfast and not as good as the best max results [1]. The difference likely owes to 50Salads being much less multimodal compared to Breakfast and the 50Salads dataset being too small (merely 160 training samples) for learning GANs well. Although our RNN model has similar structure to [1], the discriminator incurs extra parameters and therefore requires more training data.

Epic-Kitchens dataset. Figure 3.12a shows results for Epic-Kitchens as prediction accuracy at specific times in seconds, i.e., (1, 3, 5, 7, 10, 20, 30, 40, 50, 60) with both backbones. Our LSTM results are the top performer when prediction time is up to 20 seconds. Afterward, we are on par with TOS [99] at 30s, but then trail by an average of $\approx 1.7\%$ out to 60s. The largest difference happens at 60s, where the gap is 3%. Resembling observations by comparable work in related areas [257, 169, 71, 45], we find the quality of generated samples decreases as the required prediction length increases. Still, our approach outperforms classic RNN and CNN approaches at all prediction times. It is notable that TOS uses a joint learning of long-term prediction with recognition of current observations [99], whereas RNN/CNN [2] and ours solely focus on sequential modelling. These results might indicate that the auxiliary term helps stabilize long-term performance. Our RNN results outperform alternatives (except our LSTM) only at one temporal position, i.e., 1s, and otherwise are analogous to the alternatives [2].



(a) Epic-Kitchen dataset



(b) Ablation on Epic-Kitchen dataset

Figure 3.12: Comparison table for the Epic-Kitchen dataset (a). Ablation study on Ours-LSTM for the Epic-Kitchen dataset. Adv_* indicates * adversarial training epochs (b).

3.4.4 Ablation studies

Analysis of adversarial learning. We now show the benefits of adversarial learning by examining the accuracy curve at various training stages using Epic-Kitchens, reporting averages across 16 samples. We use Epic-Kitchens as our above experiment with this dataset showed the largest performance change with prediction time; Figure 3.12b has results. Just after MLE pretraining, accuracy is lower than our comparison approaches, Figure 3.12a. As we start adversarial learning, accuracy immediately rises and does so essentially monotonically with increased training epochs for lower prediction times, and begins to converge at 1200 epochs. For epochs beyond 1500, there is evidence of overfitting at longer prediction times, as performance decreases.

Analysis of number of samples. We now examine the influence of the number of samples on accuracy using the Breakfast dataset as it is larger than the others; see Table 3.4. We see the largest accuracy increases occur as samples go from two to eight and (less so) eight to 16. Afterwards, improvements grow much slower and performance is stable. In accord with Section 3.4.4 results, adjacent predictions (e.g., 10%) benefit more than remote ones (e.g., 20-50%). Empirically, we find 16 samples work for both accuracy and efficiency.

Analysis of normalized distance regularization and diversity. We now provide experimental analysis of the influence our model has on the quality-diversity trade-off. Here, we use the Breakfast dataset because of its large number of videos and relatively high diversity. To

Observation	20%				30%			
Prediction	10%	20%	30%	50%	10%	20%	30%	50%
2 Samples	0.6157	0.4525	0.3541	0.2382	0.6408	0.4839	0.3775	0.2570
8 Samples	0.7060	0.5950	0.4395	0.3927	0.7032	0.6564	0.5716	0.4320
16 Samples	0.7222	0.6240	0.5622	0.4595	0.7414	0.7132	0.6530	0.5238
32 Samples	0.7464	0.6319	0.5475	0.4643	0.7611	0.7155	0.6566	0.5219
64 Samples	0.7572	0.6325	0.5547	0.4674	0.7758	0.7150	0.6650	0.5291

Table 3.4: Influence of various number of samples for Breakfast Dataset, Protocol 1, with dense anticipation mean over classes (MoC) accuracy under Avg metric and LSTM backbone.

Temperature	Loss	LL	Diversity	Accuracy
$\alpha = 0.1$	Adv	-10.83	0.033	23.01
$\alpha = 1$	Adv	-6.77	0.124	60.98
$\alpha = 10$	Adv	-8.42	0.345	35.19
$\alpha = 100$	Adv	-8.52	0.423	33.78
$\alpha = 1000$	Adv	-11.46	0.582	19.97
$\alpha = 0.1$	Adv + NormDist	-9.04	0.062 (+0.029)	27.61 (+4.60)
$\alpha = 1$	Adv + NormDist	-4.83	0.262 (+0.138)	68.96 (+7.89)
$\alpha = 10$	Adv + NormDist	-8.19	0.358 (+0.013)	36.64 (+1.45)

Table 3.5: Comparison across various temperatures as well as impact of the normalized distance regularizer. + indicates improvement of adding the normalized distance regularizer (NormDist) to pure adversarial learning (Adv). LSTM backbone is adopted.

evaluate quality, we adopt the accuracy metric, as in Table 3.2. To evaluate diversity, we calculate the averaged pairwise cosine distance amongst 10 samples, cf. [254] as well as LL, as in Table 3.2. Both metrics are presented for pure adversarial learning (Adv) and with the aide of normalized distance based adversarial learning (Adv + NormDist); Table 3.5 has results. With pure adversarial learning (i.e., no normalized distance regularizer), too small or too large of a temperature, *e.g.*, $\alpha \in \{0.1, 10, 100, 1000\}$ leads to unsatisfactory results. Best results are obtained when $\alpha=1$; however, the diversity score is not as good as with higher α and the quality score merely equals that of deterministic APP-LSTM in Table 3.2, row 2.

After joint training with the distance regularizer, quality and diversity scores grow together. Especially, when $\alpha=1$ the model receives the most benefit in terms of quality and diversity (+2% for LL, +0.138 for cosine distance and +7.89% for accuracy), while with $\alpha=10$ the model yields smaller improvements. The reason for this pattern is that large α values already yield adequate diversity; so, there is little for the distance regularizer to offer. In the case of $\alpha \geq 10$, however, LL suffers noticeably. Theoretically, small temperature yields accurate

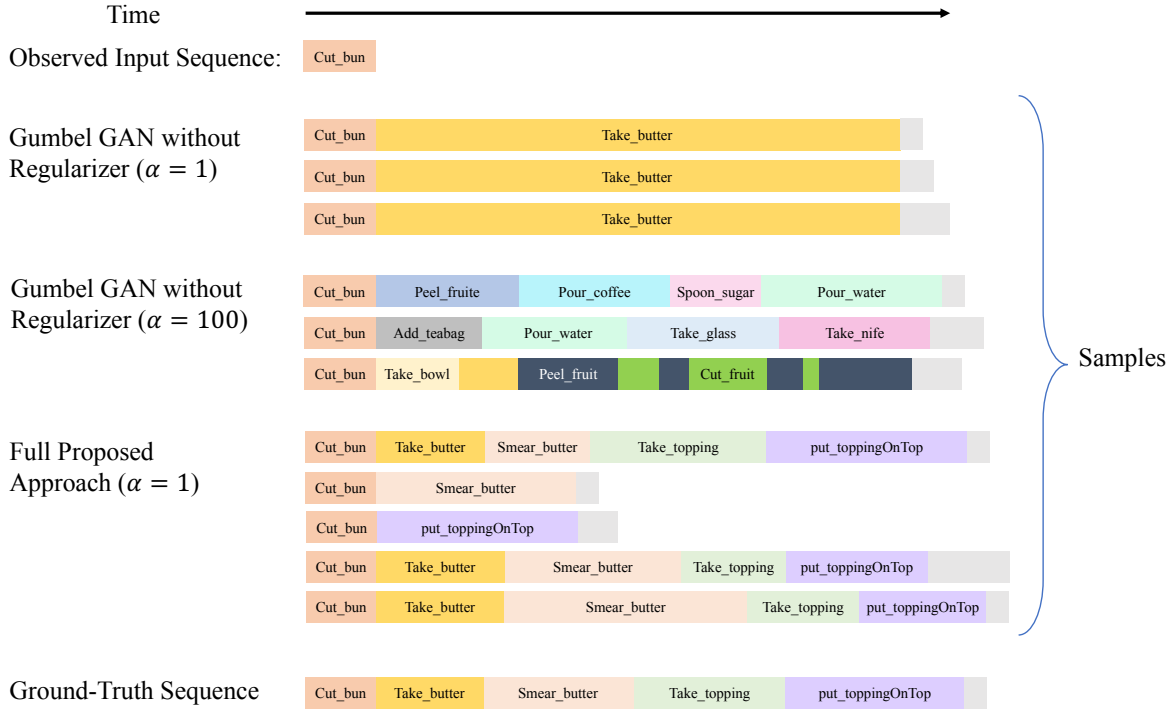


Figure 3.13: Visualization of multimodal samples from our full model vs our model lacking normalized distance regularization.

one-hot approximation; however, in our experiments $\alpha=0.1$ does not perform better, possibly because even with our regularizer, gradient variance is too high.

3.4.5 Visualization of diversity and quality

Figure 3.13 shows sampled outcomes for our full approach versus our approach lacking normalized distance regularization. Without regularization the small temperature ($\alpha = 1$) samples lack diversity (i.e., *Take_butter* is always generated after *Cut_bun*), whereas large temperature ($\alpha = 100$) samples are diverse to the point of being unrealistic (e.g., *Add_tea* generated after *Cut_bun*), i.e., they lack quality. In contrast, our full approach gives both plausible and diverse outputs. Figures 3.14 and 3.15 provide two further illustrations of the efficacy of our approach.

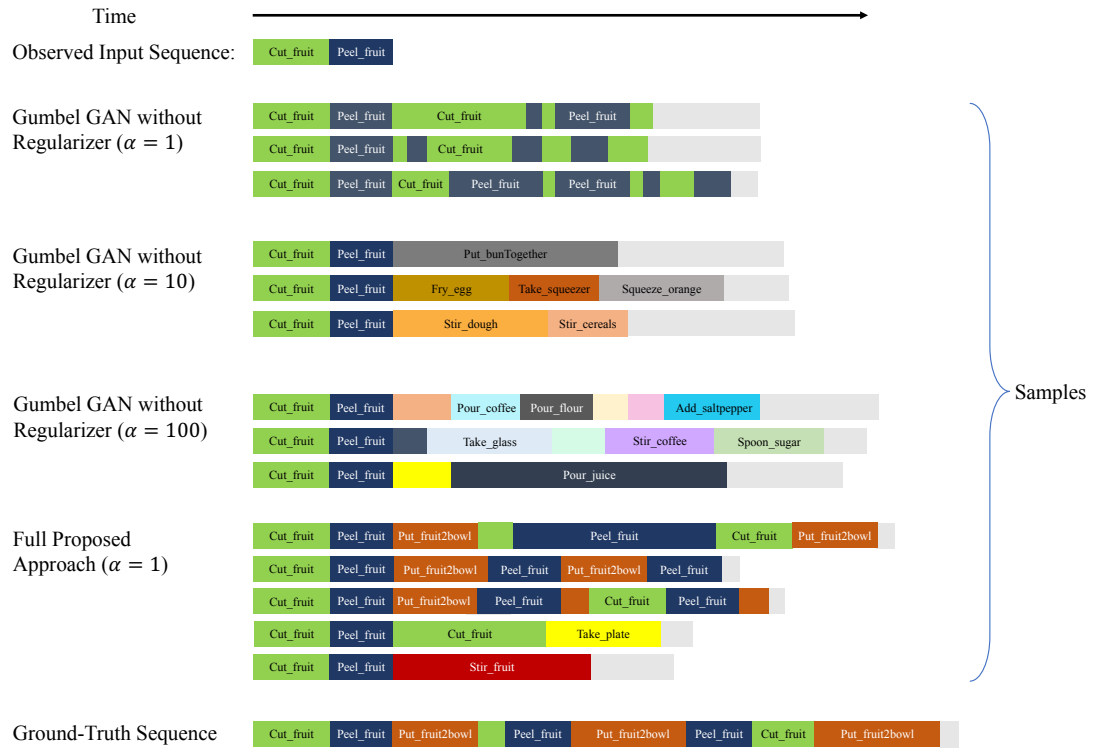


Figure 3.14: Visualization of multimodal samples from our full model vs our model lacking normalized distance regularization.

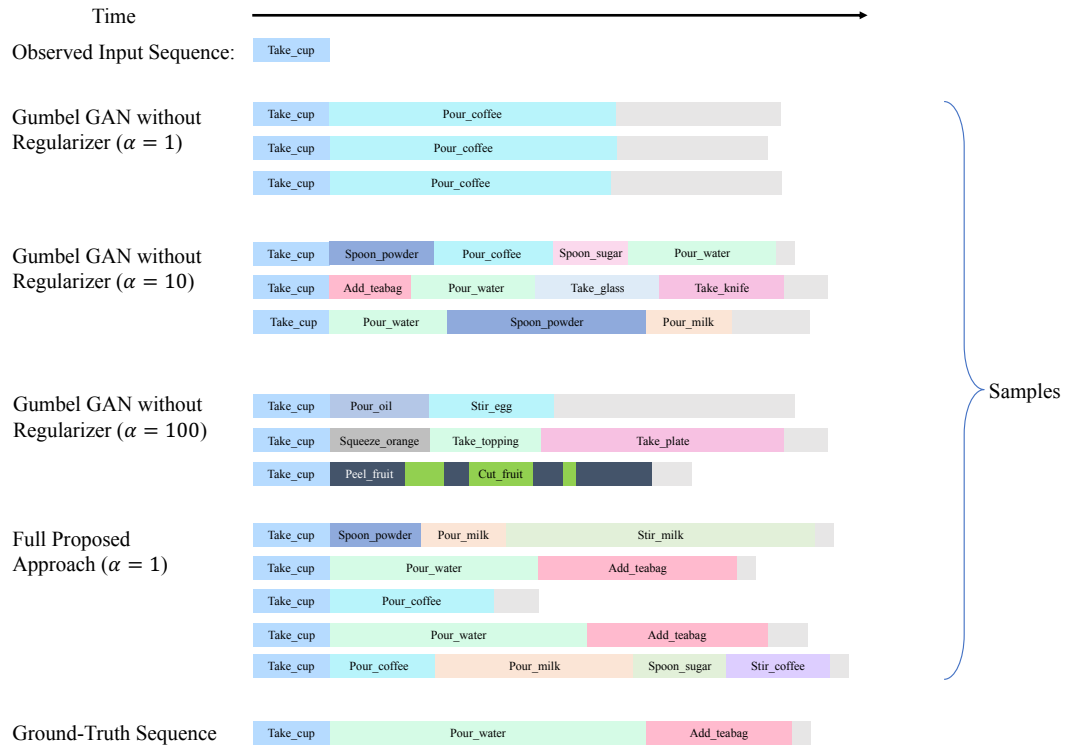


Figure 3.15: Visualization of multimodal samples from our full model vs our model lacking normalized distance regularization.

3.5 Conclusion

In activity anticipation, it is desirable to produce results that realistically capture the potentially multimodal distribution of future actions given initial observations. We have responded with a novel model combining a sequential GAN (for realistic predictions) with a sample distance regularizer (for diverse predictions). By unifying action and time representations, the proposed model can produce outputs that are both realistic and diverse, thereby yielding new state-of-the-art performance on standard activity anticipation datasets. We demonstrated our approach with two different sequence generation backbones (RNN and LSTM).

In terms of limitations, the interpretability of certain parts of our approach needs to be enhanced. Although the used distance regularizer loss is self-explanatory, other parts, e.g., the seq2seq model and embedding layers, need further analysis regarding what has been learned. Moreover, daily activity videos are often minutes long and it is unrealistic to use frame-wise features, which motivates researchers (including our work) to use high-level abstraction (e.g., activity semantic labels) as input. The drawback of this approach is a loss of detailed feature information; therefore, to maintain a manageable feature history that balances fine-grained details and high-level abstraction is the key to effectively associating past events with future predictions and deserves further investigation.

Chapter 4

Video trajectory prediction

4.1 Motivation

Video predictive understanding on physical motion patterns of human or robotic agents (e.g., 2D coordinate trajectories) is as essential as those discussed in previous chapters (i.e., early action recognition and future activity prediction). Forecasting the future trajectories of pedestrians in crowded scenes is an example of such research and recently has received considerable attention [4, 72, 256, 115, 131]. It studies the ability of artificial vision systems to anticipate the future motion of individuals from current observations and therefore is of importance to a variety of allied areas, including self-driving vehicles, service robots and surveillance systems [190].

Research on modeling pedestrian walking trajectories has evolved from relatively simple physical motion models (e.g., social force [82] or constant velocity [202]) to more sophisticated efforts that take into account social compliance [165, 264, 198], environmental awareness [140, 195, 215, 139] as well as end-goal policies [158, 37]. Recent efforts have found notable performance improvements by encoding goal positions (also dubbed destinations or endpoints) together with historically observed trajectories, with special effectiveness noticed on long-term prediction horizons. These efforts operate essentially in two steps: (i) Inference of goal positions from estimators typically trained in parallel with trajectory estimators; (ii) subsequent trajectory prediction that forecasts unseen movement, conditioned on both the past motion history and the inferred goal information. In nature, this scheme implicitly converts

trajectory extrapolation to interpolation (i.e., bridging the pathway between initial trajectories and goal positions).

Goal-based research has been seen in a variety of places, e.g., motion planning [95] and reinforcement learning [96, 167, 53, 61]. These efforts either pre-define the desired goal space with human supervision [40, 172] or leverage a learnable module to obtain that information directly from input, e.g., preliminary states or raw images [167]. The latter is favored by the general trajectory prediction field [158, 37], because typically pedestrians walk through scenes that do not have a priori specified goal positions. However, this choice raises an additional need: Training side models to infer goal positions during testing, which demands extra learnable parameters and goal annotations, if not given by default. Moreover, the learned goals might not be of ideal quality, e.g., violating road boundaries or traffic rules.

4.2 Related work

4.2.1 Collective dynamic modelling on pedestrians trajectories

Human trajectory forecasting has seen great recent progress. Exploring the collective dynamics behind a group of walking pedestrians in complex scenes is one of the main focuses in the past few years [4, 72, 14, 264]. Trending techniques for modelling this social compliance include attention [227] and graph neural network [104] frameworks. The core idea is to pass the information of each individual pedestrian (e.g., motion history) to the neighboring agents, therefore the decisions that are made considering his/her surroundings can contribute to a more sensible motion forecasting, e.g., notable improvements on collision-avoidance [165] compared to solely modelling the information of a single trajectory.

4.2.2 Modelling physical constraints

Meanwhile, modelling the constraints from physical environments (e.g., obstacles, walls, walking/driving lanes, etc.) is another direction that has shown solid benefits [195, 130, 140, 198, 215]. Some research has implicitly conditioned the trajectory forecasting on the raw landscape image (e.g., RGB image of the viewed scenes) [195] or corresponding semantic

extractions (e.g., semantic segmentation maps) [130, 140, 198]. Yet, some others explicitly build potential fields to guide pedestrians using the prior knowledge from scene images [215].

4.2.3 Future distribution modelling

Producing multi-modal predictions also has received considerable attention, as it captures the stochastic possibilities of human daily walking patterns, cf. [72, 198, 130]. Generative probabilistic models are used as the major source for this purpose, e.g., Variational Auto-Encoder (VAE) [102] and Generative Adversarial Networks (GAN) [67]. *Social-GAN* [72] is such a work that used random latent variables sampled from a standard Gaussian distribution to capture variability. Meanwhile, *Trajectron++* [198] provides a typical use of VAEs for future distributions modelling. Also, there are parallel research efforts that enforce a Gaussian Mixture Model (GMM) distribution prior on regressed output trajectories, e.g., [88, 165]. More recently, some efforts have demonstrated that the mixture of non-parametric generative models can enhance the distribution modelling by considering the out-of-distribution (OOD) samples [36].

4.2.4 Goal conditioned prediction

Recently, goal conditioned approaches have shown superior performance over the aforementioned approaches [222, 179, 158, 37]. One such effort models the causal relationship between semantic goals (e.g., right-turn or go-straight) and future trajectories [179, 222, 186], while others rely on positional goals (e.g., destination coordinates) [158, 37]. Common across these approaches is establishment of a supervised goal estimator to assist later trajectory forecasting. In contrast, while our work exploits goal information, it does so with a novel, nonparametric search-based approach.

4.2.5 Expert learning

Learning from an expert is an established principle. This research direction assumes that a group of representative examples can act as an intelligent system to model versatile real world data. For instance, an earlier effort grouped a set of human walking examples to model crowd

trajectories in simulation environments [129]. Some recent work also found it useful to assist multi-modal video frame prediction [251] as well as adaptive robot locomotion generation [253]. Other work has used example extrapolation to remedy data under-representation for robust learning [126]. The intuition of using expert examples also has been used in recent efforts aimed at data efficient learning (e.g., one-shot [230], prototype [208] and few-shot [273] learning). Here, research finds that training of intelligent models can depend on only a small amount of annotated examples, as other unobserved data can be self-annotated by matching with adjacent expert examples [10, 64, 230, 208, 247, 126].

4.3 Overall contribution: Dynamic trajectory prediction with expert goal examples

Our proposed solution is inspired by techniques seen in expert learning and data efficient machine learning; see Figure 4.1 for an overview. We apply their insight on the use of expert examples to the task of goal conditioned trajectory prediction, with a particular focus on helping the goal inference step. We make use of available trajectory training data to serve as an expert repository that we can index into based on observed test trajectories. Then, the goals of the indexed trajectories are used as input to our full trajectory estimator. We found that running similarity search with a customized dynamic time warping (DTW) [197] metric yields high-quality goal estimations for unseen test trajectories, which further produces superior evaluation results for the overall forecasting. Notably, the searching step can be sped by existing tools [166] to satisfy real-time inference. In the following sections, we provide detailed descriptions on each introduced component.

4.3.1 Problem formulation

We seek to predict the correct future trajectory of the i^{th} pedestrian in 2D coordinates: $\hat{\mathbf{Y}}_i = \{(\hat{x}_i^t, \hat{y}_i^t) \in \mathbb{R}^2, t = \{t_{obs+1}, \dots, t_{end}\}\}$, given M co-existing pedestrians and their observed trajectories $\mathbf{X}_i = \{(x_i^t, y_i^t) \in \mathbb{R}^2, t = \{1, \dots, t_{obs}\}\}$ as inputs, where $i \in [1, M]$. More specifically, we assume the predicted coordinates $(\hat{x}^t, \hat{y}^t)$ are random variables that follow a bivariate Gaussian distribution, i.e., $(\hat{x}^t, \hat{y}^t) \sim \mathcal{N}(\mu_x, \mu_y, \sigma_x, \sigma_y, \text{corr}_{xy})$, so that diverse

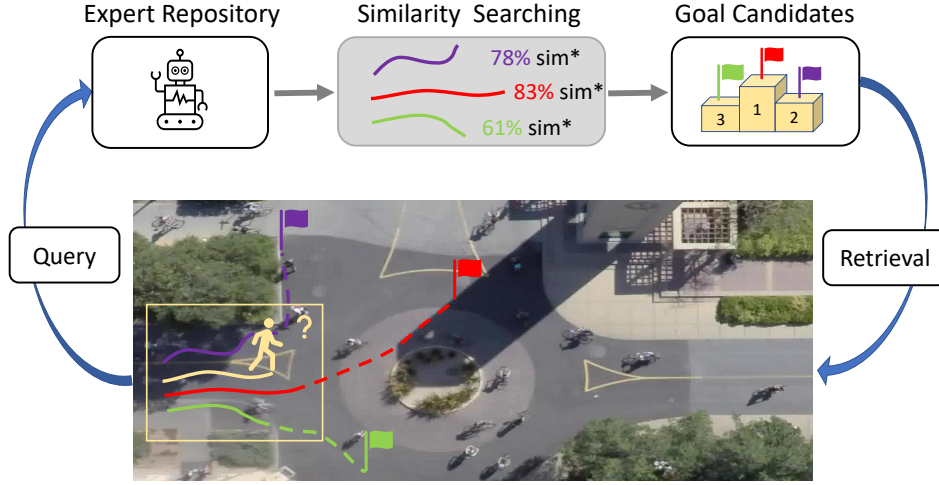


Figure 4.1: Overview of the Query-Retrieval based framework. A test trajectory with unknown future goal position is matched with expert examples that previously have been seen and stored in an expert repository. Comparisons are based on same observation length test (query) and stored expert example (shown as solid lines) trajectories. This step results in multi-modal nearest trajectories with high similarity being retrieved, i.e., those shown in **purple**, **red** and **green**, as well as their goal positions (denoted as colored **flags**) as potential goals for subsequent full trajectory forecasting.

outcomes can be sampled to support multi-modality.

Our approach proceeds in the following two steps: First, we query pseudo goal positions $(\hat{x}_i^{t_{end}}, \hat{y}_i^{t_{end}})$ of the testing input, $\mathbf{X}_i$, through a search in an expert repository of example trajectories, $\mathbf{X}$. Each entry in this repository is comprised of a trajectory sequence, $\mathbf{X}_e$, in the same format as $\mathbf{X}_i$ and its corresponding end positions $(x_e^{t_{end}}, y_e^{t_{end}})$. The end positions of the K_e nearest neighbors of the test trajectory, $\mathbf{X}_i \in \mathbf{X}$, are returned, with K_e the number returned. The repository is built from training data, as detailed in Section 4.4.2. Second, we predict the future trajectory, $\hat{\mathbf{Y}}_i = f(\mathbf{X}_i, \hat{x}_i^{t_{end}}, \hat{y}_i^{t_{end}})$, with $f(\cdot)$ denoting the subsequent trajectory predictor. In the following sections, we elaborate how these two steps work in detail. Figure 4.2 provides a summary of our overall approach.

4.3.2 Goal retrieval via dynamic time warping

The first component to our approach is a search engine that runs a similarity comparison on testing data and expert examples, i.e., those contained in $\mathbf{X}$. We retrieve useful goal estimates

according to

$$\{(\hat{x}_i^{t_{end}}, \hat{y}_i^{t_{end}})\}_{K_e} = \mathcal{S} \left(\arg \min_{\mathbf{X}_e \in \mathbf{X}} (\mathcal{D}(\mathbf{X}_i, \mathbf{X}_e)) \right), \quad (4.1)$$

where $\mathcal{D}$ is a distance function between two trajectories, K_e above the $\arg \min$ operator symbolizes that the K_e entries in $\mathbf{X}$ that yield the smallest distance are returned and $\mathcal{S}$ selects the end positions of those matches. We select the K_e smallest distance $\mathbf{X}_e$ by calculating the distance between the test trajectory, $\mathbf{X}_i$, and each entry in $\mathbf{X}$, sorting them by distance and taking the K_e with the smallest distance. $\mathcal{S}$ simply selects the end positions associated with each of these trajectories in the repository. In other words, we take the goal position out of the closest K_e expert examples as the pseudo goal for testing data.

For the matching function, $\mathcal{D}(\cdot)$, we find Dynamic Time Warping (DTW) works effectively for our needs. DTW is a well-established approach for measuring the distance between temporal sequences [197]. Originally, it was solved via dynamic programming, which is prohibitive to real-time systems due to the quadratic time and space complexity. Recently, however, it has been relaxed in computational expense (i.e., parallel computing with CUDA acceleration), made differentiable and gained in popularity, e.g., [31, 269, 24, 153]. What is particularly interesting to us is its computational efficiency. Specifically, we follow some existing examples [31, 24] to define the matching function γ -Soft-DTW as

$$\begin{aligned} \mathcal{D}(\mathbf{X}_i, \mathbf{X}_e) &= \text{DTW}_\gamma(\mathbf{X}_i, \mathbf{X}_e) \\ &= \min_\gamma \{ \langle A, \Delta(\mathbf{X}_i, \mathbf{X}_e) \rangle, A \in \mathbb{R}^{n \times m} \}, \end{aligned} \quad (4.2)$$

where $\Delta(\cdot)$ is the distance matrix (e.g., Euclidean) measuring element-wise adjacency, A is the alignment matrix that denotes the matching choices and the inner product operator, $\langle \cdot \rangle$, yields the similarity score. Here, the soft min, $\min_\gamma$, with $\gamma \geq 0$, is defined as [31]

$$\min_\gamma(a_1, \dots, a_n) = \begin{cases} \min_{i \leq n} a_i, & \gamma = 0 \\ -\gamma \log \sum_{i=1}^n e^{-a_i/\gamma} & \gamma > 0, \end{cases} \quad (4.3)$$

where the a_i represent entries in the distance matrix and γ is a smoothing factor with value set empirically; see Section 4.4.2.

Finally, for better informed matching, we enrich the trajectory descriptors by concatenating their motion information as velocities ($\mathbf{V}_i, \mathbf{V}_e$), i.e., the argument to $\mathcal{D}$ in (4.1) becomes

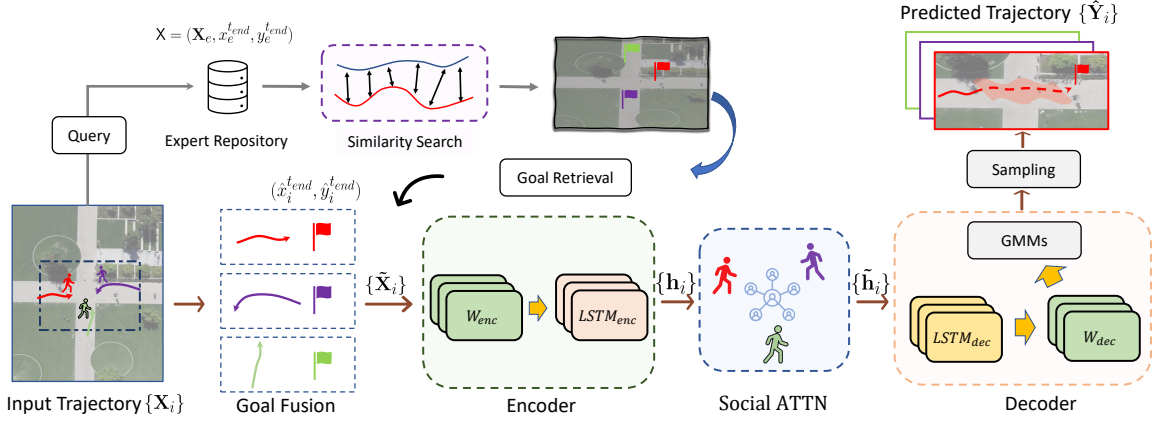


Figure 4.2: Pipeline of our goal-retrieval based trajectory prediction algorithm. Groups of partially observed trajectories, $\{X_i\}$, are processed initially by a Query-Search engine operating over an expert repository, X , to produce pseudo goal candidates, $\{\hat{x}_i^{t_{end}}, \hat{y}_i^{t_{end}}\}$ (purple dashed box). Subsequently, the fused information of observed trajectories and estimated goals, $\{\tilde{X}_i\}$, is forwarded through a sequential encoder (green dashed box) where inputs are embedded in a high dimensional feature, $\{h_i\}$. Following social attention processing (Social ATTN), a sequential decoder (pink dashed box) recursively predicts a bivariate Gaussian at every time step. Final predicted trajectories, $\{\hat{Y}_i\}$, are obtained via multiple sampling from the bivariate Gaussian.

$(cat(X_i, V_i), cat(X_e, V_e))$. Thus, similarity considers not only geo-location, but also speed and direction.

Figure 4.3 plots goal search results on the evaluated datasets using our approach. It is seen that a large portion of goal retrievals are of high quality, e.g., 83% of test data from the Stanford Drone Dataset [189] yield retrieval error smaller than 10 pixels, amongst which more than half are close to perfection, i.e., ≤ 1 pixel error, (a). Goal searching on another five datasets [129, 174] demonstrates consistently good results, (b)-(f). Notably, we are able to achieve this level of performance without the need to learn a model, as the training data serves as its own model in terms of the repository, X . Moreover, our similarity search through the repository can be implemented with modest computational cost; see Section 4.4.5.

Following the protocol for goal conditioned trajectory prediction proposed elsewhere [158], we assess all K_e goal candidates with respect to groundtruth and select the one that provides smallest error. Thus, only a single goal candidate, $(\hat{x}_i^{t_{end}}, \hat{y}_i^{t_{end}})$, is used along with the test trajectory, X_i , as input to the trajectory predictor, as described next.

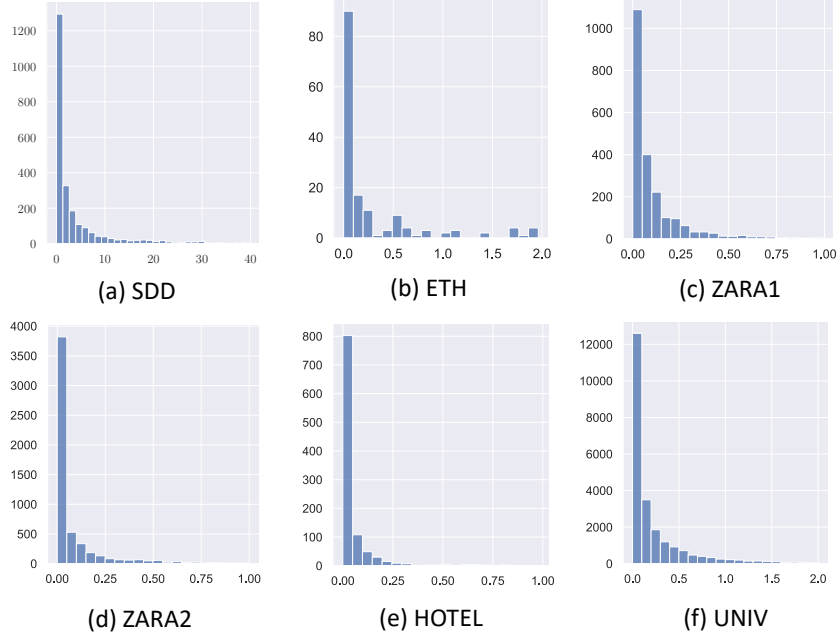


Figure 4.3: Illustration of goal retrieval quality by plotting the bin histogram by grouping retrieval errors (l_2 norm) into "bins" of equal width over test data of all datasets in our experiments. X-axis stands for the error, with values closer to 0 the better. Y-axis stands for how many data points are in that bin. These results are achieved via similarity searching across an expert repository using dynamic time warping.

4.3.3 Goal conditioned trajectory predictor

We now detail our subsequent trajectory predictor incorporating past observations, $\mathbf{X}_i$, and queried goal positions, $(\hat{x}_i^{t_{end}}, \hat{y}_i^{t_{end}})$, to infer diverse and accurate predictions. Figure 4.2 provides an overview.

Goal encoding as shifting by goal. Our model handles goal information differently from existing work, where goal positions were concatenated with motion history in high-dimensional feature space [158], or explicitly used to compute the remaining distance as an additional input, e.g., [37]. Both methods lead to extra embedding efforts.

Instead, we are motivated by the intuition of shifting data according to the mean, adopted by work in machine learning (e.g., batch normalization) and sequence modelling (e.g., temporal subtraction for trajectory stationarization in *Trajectron++* [198]), and find it equally sufficient to subtract goal position values from all past motion trajectories before encoding them with Multi-Layer Perceptrons (MLPs). By performing this transformation, we incorpo-

rate goal information into feature embedding with zero extra effort; in particular, we define

$$\begin{aligned}\tilde{\mathbf{X}}_i &= \mathbf{X}_i - (\hat{x}_i^{t_{end}}, \hat{y}_i^{t_{end}}) \\ &= \{(x_i^t, y_i^t) - (\hat{x}_i^{t_{end}}, \hat{y}_i^{t_{end}}), t = \{1, \dots, t_{obs}\}\},\end{aligned}\tag{4.4}$$

as our shifted input trajectory and

$$\mathbf{F}_i = W_{enc}(\tilde{\mathbf{X}}_i)\tag{4.5}$$

as the shifted encoding. $\mathbf{F}_i$ associates the projected high-dimensional feature of 2D coordinates for every time stamp, i.e., $\mathbf{F}_i \in \mathbb{R}^{D \times t_{obs}}$ and $W_{enc} \in \mathbb{R}^{2 \times D}$. W_{enc} is realized as a MLP. An ablation study on our choice over concatenating goals with input motion history is provided in Section 4.4.5.

Note that during training, we use the ground-truth goal positions, $(x_i^{t_{end}}, y_i^{t_{end}})$, as input for (4.4) to prevent the learning process from being disturbed by noisy data, whereas the queried goal positions are used for testing.

Trajectory Prediction. For computing outputs, $\hat{\mathbf{Y}}_i$, given a sequence of input embeddings, $\mathbf{F}_i$, a seq2seq generator implemented as two Long Short-Term Memory (LSTM) units [86] is adopted. Sequence generation proceeds by sequentially encoding and decoding the embedded features, followed by mapping to intermediate results that are used recursively for subsequent prediction according to

$$\mathbf{h}_{enc}^k = \text{LSTM}_{enc}(\mathbf{F}_i^k, \mathbf{h}_{enc}^{k-1}), \quad k \in (1, t_{obs}),\tag{4.6}$$

where $\mathbf{h}_{enc}^k$ is the k^{th} hidden encoder state and the initial hidden state, $\mathbf{h}^0$, is sampled from a normal distribution.

For decoding, another LSTM whose first input is set to the concatenation of the encoded history, $\mathbf{h}_{enc}$, and the last observed coordinates, $\mathbf{X}_i^{t_{obs}}$, is used to produce an output hidden states sequence in a recursive fashion according to

$$\mathbf{h}_{dec}^{k+1} = \text{LSTM}_{dec}(\text{cat}(\mathbf{h}_{enc}, \hat{\mathbf{Y}}_i^k), \mathbf{h}_{dec}^k), \quad k \in (t_{obs}, t_{end}),\tag{4.7}$$

where $\hat{\mathbf{Y}}_i^k$ is the next coordinate produced online.

To allow multi-modal forecasting, we set the output to be the parameters of a bivariate Gaussian cf. [165, 198]:

$$\mu_x, \mu_y, \sigma_x, \sigma_y, \text{corr}_{xy} = W_{dec}(\mathbf{h}_{dec}^k); \quad (4.8)$$

$$\hat{\mathbf{v}}_i^k \sim \mathcal{N}(\mu_x, \mu_y, \sigma_x, \sigma_y, \text{corr}_{xy}); \quad (4.9)$$

$$\hat{\mathbf{Y}}_i^k = \hat{\mathbf{Y}}_i^{k-1} + \hat{\mathbf{v}}_i^k, \quad (4.10)$$

where W_{dec} is a MLP decoder that projects the decoded LSTM hidden state, $\mathbf{h}_{dec}^k$, to a five-dimensional vector representing the bivariate Gaussian, $\mathcal{N}(\mu_x, \mu_y, \sigma_x, \sigma_y, \text{corr}_{xy})$. Finally, the full prediction, $\hat{\mathbf{Y}}_i^k$, can be recovered by adding the previous prediction, $\hat{\mathbf{Y}}_i^{k-1}$ and the sampled motion vector, $\hat{\mathbf{v}}_i^k$, according to (4.9) and (4.10).

Social Compliance. To consider the collective effect from co-existing pedestrians, we follow recent findings and use an attention mechanism on pedestrians that are near to each other according to a threshold. Within the threshold, neighboring pedestrians, e.g., $(\mathbf{X}_i, \mathbf{X}_j)$, are given a connectivity value, $C_{i,j}$ of 1, otherwise 0, i.e., if $d(\mathbf{X}_i, \mathbf{X}_j) < \text{threshold}$: $C_{i,j} = 1$; else $C_{i,j} = 0$. We use the l_2 norm as the distance function $d(\cdot)$ and choose thresholds using precedent procedures [158, 198], as detailed in Section 4.4.2. The attention mechanism operates on the last output of LSTM_{enc} , here simplified as $\mathbf{h}_i$. In particular, letting

$$e(i, j) = \text{softmax}(W_\theta \cdot (\mathbf{h}_i) W_\phi(\mathbf{h}_j)), \quad (4.11)$$

the attention weighted output is given as

$$\tilde{\mathbf{h}}_i = \sum_{j \in M} C_{i,j} e(i, j) W_g(\mathbf{h}_j), \quad (4.12)$$

where W_θ and W_ϕ are learned linear transformation matrices on arbitrary pairs of pedestrians prior to normalized weights conversion, $e(i, j)$. Subsequently, a weighted sum operation, (4.12), is applied on the results of another learned linear transform matrix, W_g , to produce the outputs. This socially attentioned embedding is more informative since it accounts for neighboring agents' motion history as well as their destination plans. We use this output as the input to the trajectory decoder, (4.7), i.e., $\mathbf{h}_{enc} = \tilde{\mathbf{h}}_i$.

4.3.4 Learning scheme

We found it sufficient to train the model end-to-end solely by minimizing the negative log-likelihood of the bivariate Gaussian on all pedestrians and future times,

$$L(\theta) = - \sum_{k=t_{obs}+1}^{t_{pred}} \sum_{i=1}^M \log(\mathbf{Y}_i | \mu_x, \mu_y, \sigma_x, \sigma_y, \text{corr}_{xy}), \quad (4.13)$$

where θ refers to parameters associated with all learnable modules, i.e., W_{enc} , W_{dec} , LSTM_{enc} , LSTM_{dec} and attention module weights $\{W_\phi, W_\theta, W_g\}$.

4.4 Empirical evaluation

4.4.1 Evaluation protocol

Datasets. To evaluate our approach, we choose three widely examined datasets, the Stanford Drone (SDD) [189], ETH [174] and UCY [129] datasets. SDD is a human trajectory prediction dataset that consists of 20 scenes in top down view. We follow the train-test split in the *TrajNet++* challenge [116] and focus on pedestrians. The ETH dataset contains two scenes (ETH and Hotel) and the UCY dataset contains three scenes (ZARA1, ZARA2 and UCY). They together consist of 1536 pedestrians. For both datasets, our model takes as input an observation of an eight timestep long trajectory and predicts the trajectory for the next twelve timesteps.

Metrics. We present prediction accuracy in terms of two well-known metrics, **Average Displacement Error** (ADE) and **Final Displacement Error** (FDE), given as

$$\text{ADE} = \frac{\sum_{i=1}^M \sum_{k=t_{obs}+1}^{t_{end}} \|\mathbf{Y}_i^k - \hat{\mathbf{Y}}_i^k\|_2}{M \times T} \quad (4.14)$$

and

$$\text{FDE} = \frac{\sum_{i=1}^M \|\mathbf{Y}_i^{t_{end}} - \hat{\mathbf{Y}}_i^{t_{end}}\|_2}{M}, \quad (4.15)$$

where M is the number of targets, T is the number of predicted timesteps, $\mathbf{Y}_i^k$ and $\hat{\mathbf{Y}}_i^k$ are

the predicted and groundtruth (resp.) positions of target i at time step k and t_{end} is the final predicted timestep.

Goal-based evaluation. Extant protocol for goal-based trajectory prediction assesses an initial set of goal samplings, selects the one closest to the ground-truth final trajectory position and then proceeds to produce midway predictions [158]. We follow the same procedure to evaluate our model, but substitute the goal sampling with our approach to goal retrieval by searching through an expert repository, as detailed in Section 4.3.2. Prior to selection of the single goal candidate passed to the trajectory predictor, the initial set of candidates searched for in the repository is $K_e = 20$, which we found to be effective and efficient, which is validated in the ablation studies; see Section 4.4.5.

Best-of-N Sampling. We report the best ADE and FDE accuracy out of multiple sampled results from our trajectory predictor, using the single selected goal retrieval. In the following evaluations, N is set to 20 for fair comparisons with existing work [4, 72]. We denote this minimizing value as Min_x , e.g., Min_{20} for $N = 20$. Various values, $N \in [5, 10]$, are considered in an ablation study in Section 4.4.5.

4.4.2 Implementation details

To build our model, we specify that LSTM_{enc} and LSTM_{dec} have hidden states of dimension 128. For the motion history encoder, W_{enc} , we adopt a MLP that consists of sequential activations with shape of $[2 \rightarrow 512 \rightarrow 256 \rightarrow 128]$. A similar MLP that has activations with shape of $[128 \rightarrow 64 \rightarrow 32 \rightarrow 5]$ is used for the bivariate-GMM decoder, W_{dec} . For the attention module, we specify the linear transformation matrices, W_θ and W_ϕ , as two MLPs with the same shape, $[128 \rightarrow 256 \rightarrow 64]$, and the W_g as the same but with shape $[256 \rightarrow 256 \rightarrow 128]$. Throughout, the ReLU activation function is used to increase nonlinearity.

For the SDD dataset, training employs the Adam optimizer and a learning rate 0.0003 with $\beta_1 = 0.9$ and $\beta_2 = 0.99$ to minimize the loss (4.13). The batch size is 512 and the training proceeds through 350 epochs. For the ETH and UCY datasets, the same optimizer is adopted to train the model for 250 epochs, with a batch size of 128. The learning rate is initialized as 0.01 for the first 150 epochs, which decays to 0.002 for the rest, cf. [165]. The values for β_1 and β_2 are the same as for SDD.

Models	Evaluation Metrics (ADE / FDE) on Min ₂₀					
	ETH	HOTEL	ZARA1	ZARA2	UNIV	AVG
Linear [4]	1.33 / 2.94	0.39 / 0.72	0.62 / 1.21	0.77 / 1.48	0.82 / 1.59	0.79 / 1.59
Social-GAN [72]	0.81 / 1.52	0.72 / 1.61	0.34 / 0.69	0.42 / 0.84	0.60 / 1.26	0.58 / 1.18
SoPhie [195]	0.70 / 1.43	0.76 / 1.67	0.30 / 0.63	0.38 / 0.78	0.54 / 1.24	0.54 / 1.15
Social-STGCNN [165]	0.64 / 1.11	0.49 / 0.85	0.34 / 0.53	0.30 / 0.48	0.44 / 0.79	0.44 / 0.75
Goal-GAN [37]	0.59 / 1.18	0.19 / 0.35	0.43 / 0.87	0.32 / 0.65	0.60 / 1.19	0.43 / 0.85
PECNet [158]	0.54 / 0.87	0.18 / 0.24	0.22 / 0.39	0.17 / 0.30	0.35 / 0.60	0.29 / 0.48
Trajectron++ [198]	0.43 / 0.86	0.12 / 0.19	0.17 / 0.32	0.12 / 0.25	0.22 / 0.43	0.20 / 0.39
Ours	0.30 / 0.62	0.09 / 0.15	0.15 / 0.31	0.12 / 0.24	0.19 / 0.44	0.17 / 0.35

Table 4.1: Evaluation results on the ETH and UCY datasets for next 12 timestep prediction. Numbers are taken from the minimum ADE/FDE of 20 randomly evaluated samples, denoted as Min₂₀. Though **Linear** is deterministic, we list it here as a sanity check. Bolded numbers indicate best performance.

We build the expert repository, $\mathbf{X} = \{\mathbf{X}_e, x_e^{t_{end}}, y_e^{t_{end}}\}$, with the same training data introduced in Section 4.4.1 for all datasets. We also enrich the repository of the ETH and UCY datasets by rotating all trajectories in a scene over a range of angles from 0° to 360° with an interval of 15° . Random rotation is often used as a data augmentation method in recent work, to combat overfitting [202, 198]. We find this augmentation unnecessary for the SDD dataset, which indicates SDD is more balanced. Empirically, we set the γ -Soft-DTW smoothing parameter to $\gamma = 2$. The social attention threshold in Section 4.3.3 is set to 100 pixel distance for SDD and three world distance for ETH/UCY, cf. [158, 198]. The entire framework is coded in PyTorch and trained on a NVIDIA TITAN X GPU.

4.4.3 Comparison to alternative approaches

ETH and UCY datasets. Table 4.1 shows comparative results for our algorithm versus various alternatives. Ours perform on-par with the previous best method Trajectron++ on the average ADE (i.e., 0.17 versus 0.20), while further reducing the FDE by 10% on average, with the biggest improvement happening in the ETH subset (e.g., around 30%). We find the lowest absolute displacement error in both ADE and FDE when evaluated on the HOTEL subset, i.e., 0.09/0.15. The overall relative success of our approach can be explained by the discrepancy in data use. Trajectron++ uses the full future trajectory (i.e., more than just goal positions) to learn a latent structure in training. This structure is supposed to implicitly provide future

information for testing. In contrast, we go further to use goal information more explicitly in both training and testing. (We further explore the peculiarities of the HOTEL dataset in the ablation studies.)

Especially when compared with two other goal-based methods, i.e., Goal-GAN [37] and PECNet [158], ours has shown to be more effective, likely for two reasons: First, both methods use deep generative models with a fixed prior distribution (standard Gaussian) to approximate the goal distribution. This paradigm has been found to suffer from diversity collapse as well as limited sample quality [251]; second, their methods are constrained to modelling the diversity of goal positions, not that of other trajectory points, which naturally lose the ability to cover a diverse set of midway trajectories. Instead, ours uses a nonparameteric approach to goal retrieval, which decouples the goal inference from subsequent trajectory sampling, and therefore reprioritizes the sampling on the overall trajectories.

SDD dataset. The evaluation results on this dataset can be viewed in Table 4.2 (i.e., Ours). Looking especially at the goal based methods (Goal-GAN [37], PECNet [158] and Ours), it is seen that more desirable performance is observed when compared to all others (e.g., graph neural network based EGraph [131], scene image conditioned CGNS [130] and the rest [72, 195]). These results show solid improvement from incorporating goal information into trajectory forecasting. Notably, our approach again achieves best results overall. Similar to the earlier discussion, we can explain these improvements in terms of goal search with respect to an expert repository being more effective than alternatives, which we further document in Section 4.4.5.

To explore further the possible performance of our model, we also show results from full twelve step trajectory sampling given retrieved goals (denoted as ours- $\mathcal{F}$), rather than the standard protocol we report elsewhere, i.e., using the goal prediction (or retrieval) results for FDE and then merging them with the first eleven timestep trajectory sampling for ADE. If allowed, our model produces exceptional results on FDE (e.g., 9.03 versus 14.38) through refinement of initial goal estimates. This result suggests that current goal-based evaluation does not adequately consider the power of goal-based estimators to influence final destinations.

Metrics	Evaluation Metrics (ADE / FDE) on Min_{20}							Ours- $\mathcal{F}$
	S-GAN-[72]	Sophie [195]	CGNS [130]	EGraph [131]	Goal-GAN [37]	PECNet [158]	Ours	
ADE	27.23	16.27	15.6	13.9	12.20	9.96	7.69	7.51
FDE	41.44	29.38	28.2	22.9	22.10	15.88	14.38	9.03

Table 4.2: Evaluation results on the SDD dataset for the next 12 timesteps trajectory prediction. Numbers are taken from the minimum of 20 random evaluated samples, denoted as Min_{20} . $\mathcal{F}$ denotes the result of sampling all next 12 steps given the retrieved goals, to reveal the full power of the presented approach.

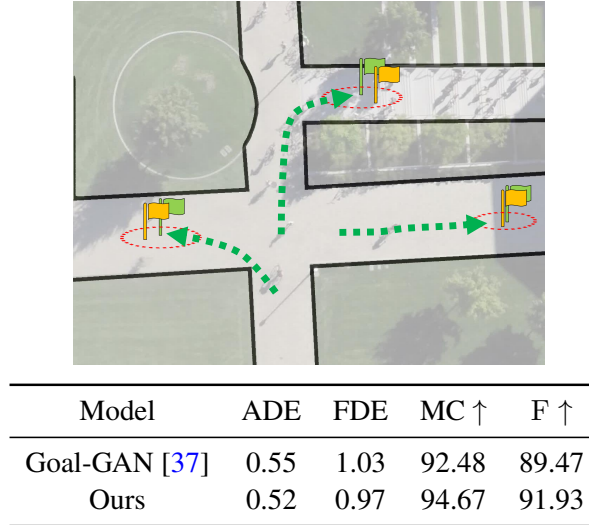


Figure 4.4: Illustration of the feasibility quality of our results on the SDD Hyang4 scene. Most of our goal retrievals (green flags) are reasonably close to GT goals (yellow flags) and our trajectory predictions (green dotted lines) respect road boundaries. See text for definition of metrics.

4.4.4 Feasibility of expert examples

To further validate our approach, we provide additional comparisons using the feasibility evaluation protocol [37] on the SDD Hyang-4 scene; see Figure 4.4 for results. Notably, two extra metrics are designed for this purpose: mode-coverage (MC) that measures the portion of goal predictions (or goal retrievals in our approach) that are distant to ground-truth goals up to $2m$ (red dotted circle) and, F, denoting the ratio of trajectories lying inside the feasible area (manually segmented road boundary).

Mode-Coverage. This metric measures the portion of goal predictions that have distance to ground-truth goals up to 2 meters in world coordinates. Formally, the MC evaluation metric

is defined according to

$$\mathbf{C}(\cdot) = \begin{cases} 1, & ||\mathbf{W}_{\text{pixel-to-world}}(\hat{\mathbf{Y}}_i^{t_{end}} - \mathbf{Y}_i^{t_{end}})||_2 \leq 2 \\ 0, & \text{else} \end{cases} \quad (4.16)$$

and

$$\text{MC} = \frac{\sum_{i=1}^N \mathbf{C}(\hat{\mathbf{Y}}_i^{t_{end}}, \mathbf{Y}_i^{t_{end}})}{N}, \quad (4.17)$$

where $\hat{\mathbf{Y}}_i^{t_{end}}$ and $\mathbf{Y}_i^{t_{end}}$ are estimated and groundtruth trajectory end points, resp., $\mathbf{W}_{\text{pixel-to-world}}$ denotes the image-to-world coordinate transformation matrix, $\mathbf{C}(\cdot)$ is the mode-coverage calculating function and MC is the final score considering all N testing data.

Feasibility. The F metric denotes the ratio of trajectories lying inside the feasible area. The transformation matrix between pixel and world coordinates for this scene is provided by [195]. The feasibility score, F, is computed via

$$\mathbf{H}(\cdot) = \begin{cases} 1, & \hat{\mathbf{Y}}_i \notin \mathbf{S}_{\text{boundary}} \\ 0, & \text{else} \end{cases} \quad (4.18)$$

and

$$\text{F} = \frac{\sum_{i=1}^N \mathbf{H}(\hat{\mathbf{Y}}_i, \mathbf{S}_{\text{boundary}})}{N}, \quad (4.19)$$

where $\mathbf{S}_{\text{boundary}}$ is the road contour and $\mathbf{H}(\cdot)$ is the function that decides if predicted trajectories traverse boundaries.

The results of Figure 4.4 show that without using any goal learning, our results outperform Goal-GAN [37]. We attribute this superiority to expert goal examples respecting environmental constraints, e.g., staying on walkways.

4.4.5 Ablation studies

Goal search efficiency. Our search engine runs in real-time, thanks to three main factors: First, relaxed soft dynamic time warping that can be computed with CUDA acceleration [31]; second, fast search for the K_e nearest neighbors to a test trajectory in the expert repository [166]; third, the searched data entity is of low dimensionality, i.e., each entry is a concatena-

tion of positions and velocities of an eight timestep trajectory. Therefore, each testing entry would cost about 10ms to grab the nearest 20 goal examples. A thorough study of other matching options and their efficiency is provided in Table 4.3a. Geo., Vel. and Euc. denote geo-locations, velocity and Euclidean, resp. Our proposed approach is denoted DTW-Dual.

Use of goal information. Given that existing work has turned to different strategies for employing goal information, we conduct experiments to systematically validate them. In particular, we study four goal use strategies: Our proposed Goal-Shift (Eq. 4.4) that subtracts goal positions from input trajectories; Goal-Cat that concatenates goals with raw inputs before encoding; Goal-Cat2 that concatenates encoded goals and inputs in feature space, cf. [158]; finally, Goal-Res that concatenates the ongoing prediction and its residual distance to the goal, cf. [37]. Results are listed in Table 4.3b. Check mark indicates that the approach incurs extra parameters. We find that the simplest strategy, i.e., Goal-shift, produces the best ADE / FDE and that is what we use for all results reported elsewhere in this work.

Table 4.4 shows results of using our predictor without conditioning on goal information. Comparison to results from the full approach (Tables 4.1 & 4.2) shows considerable benefit of goal conditioning. Table 4.4 also shows results when rather than invoking our predictor, trajectory prediction is based simply on the next twelve timesteps of the best matched eight timestep trajectory in the expert repository. Again, it is seen that the full approach provides much better results.

Number of samples. Table 4.5 has results of different combinations of K_e and N , which always sum to 20, as previous approaches typically rely on a total of 20 samples. We find that a good balance between goal candidates and trajectory prediction samples, (e.g., $K_e = 12$,

Match $\mathcal{D}(\cdot)$	ADE	Time
DTW-Dual.	7.69	10.9ms
DTW-Vel.	7.95	7.1ms
DTW-Geo.	8.68	7.1ms
Euc.-Vel.	8.43	6.2ms
Euc.-Geo.	9.01	6.2ms

(a) Goal search comparison.

Methods	ADE	Param
Goal-Shift	7.69	✗
Goal-Cat	10.74	✓
Goal-Cat2	9.06	✓
Goal-Res	11.43	✓

(b) Goal use comparison.

Table 4.3: Ablation studies for accuracy and search speed versus matching function as well as accuracy versus goal-encoding on SDD. See text for details.

Dataset	ETH-UCY		SDD	
	ADE	FDE	ADE	FDE
Predictor w/o goal	0.35	0.65	16.35	20.65
Retrieval only	1.34	1.81	32.35	46.46

Table 4.4: Ablation results based on our trajectory predictor without goal conditioning and on a retrieval-only approach.

K_e	5	10	12	15	20	20
N	15	10	8	5	1	1 *
ADE	10.39	9.65	9.43	9.89	13.24	9.11
FDE	23.40	18.43	17.32	15.82	14.38	15.20

Table 4.5: Ablation study on two hyperparameters, i.e., K_e and N , which correspond to the top two rows. Each cell shows results of different configuration on the SDD dataset in the ADE / FDE metrics. * denotes results from our goal and pretrained trajectory predictor of PECNet [158].

$N = 8$), excels on the ADE; yet, the larger the K_e , the lower the FDE.

Table 4.6 further shows more generally that while more goal samples yields better results, in most cases there are diminishing returns beyond 20 samples. We also see that the retrieved goal results most favor the HOTEL subset amongst the five, i.e., smallest displacement error with groundtruth goals against goal candidate at all levels. This may be explained by a greater portion of its trajectories being linear, cf. results of **Linear** in Table 4.1.

Repository size. Table 4.7 shows accuracy results as the size of the repository is reduced systematically, i.e., few-shot goal retrieval. It is seen that there is only a gradual fall-off in accuracy as fewer entries are made available.

Compatibility. As another comparison, for the single-shot trajectory prediction setup (i.e., $N=1$) in the right-most column of Table 4.5, we insert our goal retrieval results into the pretrained trajectory predictor of PECNet [158]; we choose that predictor module as it is trained intentionally for deterministic prediction. We see that our goals bring instant improvement, without any modification on either side. This result reaffirms our goal retrieval quality, e.g., as shown in Figure 4.3, as well as that our goal retrieval module is readily compatible with other approaches.

	ETH	HOTEL	ZARA1	ZARA2	UNIV
$K_e = 5$	1.01	0.27	0.45	0.45	0.60
$K_e = 10$	0.71	0.17	0.34	0.31	0.49
$K_e = 20$	0.62	0.15	0.31	0.24	0.44
$K_e = 50$	0.48	0.09	0.25	0.19	0.41

Table 4.6: Goal retrieval quality versus various number of repository search candidates, K_e , on the ETH and UCY datasets. Lower is better.

$R(\%)$	10	20	30	40	60	80	90
ADE	9.12	8.68	8.10	8.02	7.84	7.73	7.70
FDE	18.83	17.06	16.20	15.80	15.92	14.46	14.40

Table 4.7: Ablation study on few-shot goal retrieval by having only a percentage, R , of the original repository available for the SDD dataset.

4.4.6 Goal and prediction visualization

To understand further why our model exhibits its strong results, we show visualizations of goal retrievals and trajectory predictions in Figure 4.5. Results from another goal-based work, i.e., PECNet [158], are also given. For both datasets, we provide three types of visual examples to shed light on the reasons behind our results: equally good for both approaches, ours performs better and both fail.

For the ETH/UCY datasets, i.e., the top row in Figure 4.5, we plot testset trajectories of the ETH subset, from which we have seen the most improvement. We observe that our method performs on-par with PECNet on linear-like trajectories (a), while ours can achieve better predictions on trajectories with relatively high curvature (b and c). The reason might be that DTW is efficient at curvy shape matching, cf. [269]. Yet, both methods fail at the U-shape trajectory (d).

For the SDD dataset (bottom row), the same good performance on linear trajectories is observed by both methods, (a). However, our approach performs much better when it comes to special road conditions, e.g., 4-way intersections and pedestrian stairs, (b). We believe the reason is that a few expert examples have been found behaving similarly at nearby geo-locations. We also see improved results on cases where the alternative fails to match the speed of future trajectories, e.g., either too fast or too slow, (c). This result can be attributed to use

of velocity in our goal searching module that explicitly considers matching quality in motion. Again, for failures, neither approach captures complex motion dynamics, e.g., 180° turn or unanticipated right turn, (d).

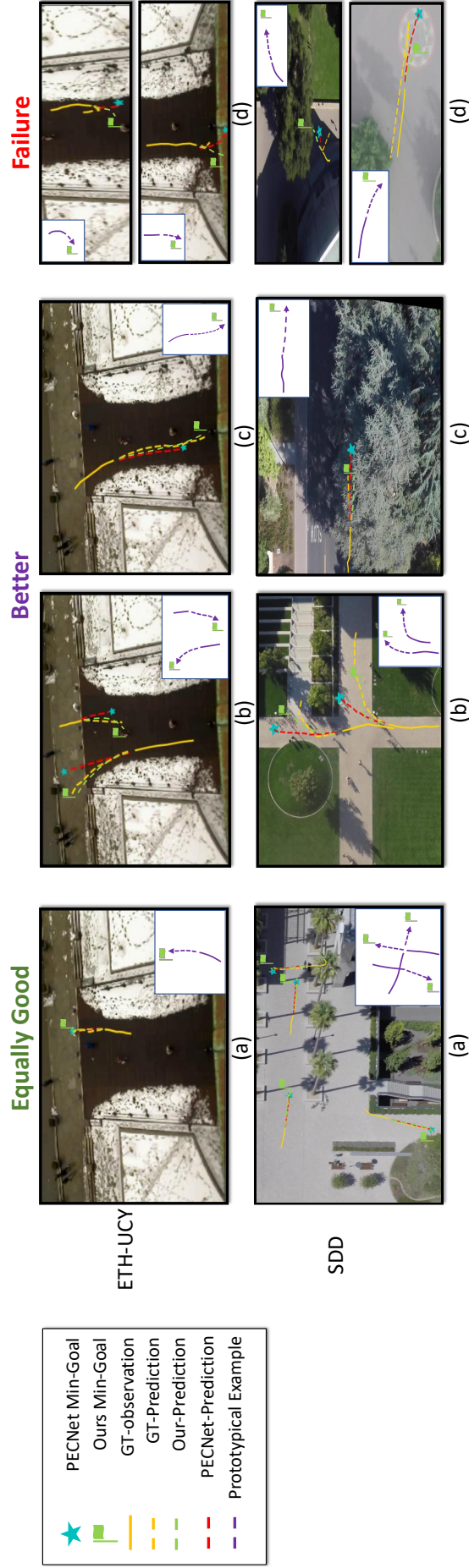


Figure 4.5: Plotting of evaluation results from the presented approach, ground-truth and a comparison work (PECNet). Top row shows studied cases for the ETH test set and bottom row shows that for the SDD test set. Each column is noted with its comparative category. From left to right: both equally good; ours better; and both failure. We found no cases where the comparison approach was noticeably better than ours. The retrieved best prototypical examples are shown as inset boxes to illustrate why goal examples are helpful (or misleading). Our goal retrievals always follow physical constraints and are interpretable.

4.5 Conclusion

We have introduced a novel approach to pedestrian trajectory prediction, where the key innovation is the use of goal search through an expert repository to provide endpoints for goal conditioned prediction. Our approach does not require learning of model parameters for goal generation, yet produces high accuracy goals at modest computational expense. We also propose a novel way to use goal data (shifting by goal) that is simpler and incurs less overhead than current alternatives, yet sets a new state-of-the-art on the SDD, ETH and UCY datasets with the goal conditioned predictor we implemented. Moreover, when using our goals as input to an alternative goal conditioned trajectory predictor (PECNet) its performance also improves, which suggests broad applicability of our approach.

One drawback of our approach is that we assume the existence of expert repositories for the viewed scenes, which might not hold when it comes to unseen environments. How to adapt existing repositories for previously unreleased data is a challenge. Domain adaptation techniques could be a potential solution. Another limitation of the current approach is lack of interpretability of the learned components in the trajectory predictor. While the goal indexing via search through an expert repository is transparent as is the shift by goal encoding, the subsequent components of the predictor require further interpretation.

Chapter 5

Weakly supervised procedure planning in instructional videos

5.1 Motivation

Procedure planning is a natural task for humans – one must plan out a sequence of actions that takes one from the current state to the desired goal. While effortless for humans, procedure planning is notoriously hard for artificial agents. Nevertheless, solving procedure planning is of great importance for building next-level AI systems capable of analyzing and mimicking human behaviour, and eventually assisting humans in goal-directed problem solving, e.g., cooking, assembling furniture and other tasks that can be represented as a clear set of instructions. Traditionally, procedure planning has been addressed in structured environments, such as object manipulation on a table surface [52, 212]. While restricting the environment helps improve planning it also limits the range of possible applications. Here, we follow more recent work [25] and tackle procedure planning in the realm of instructional videos [224, 274]. Given visual observations of the start and goal states, the task is to predict a sequence of high-level actions needed to achieve a goal; see Figure 5.1. This task is particularly challenging as it requires parsing unstructured environments, recognizing human activities and understanding human-object interactions. Yet, the range of applications for such planners is broad, which motivates research efforts on this problem.

Current approaches for procedure planning from instructional videos share one serious

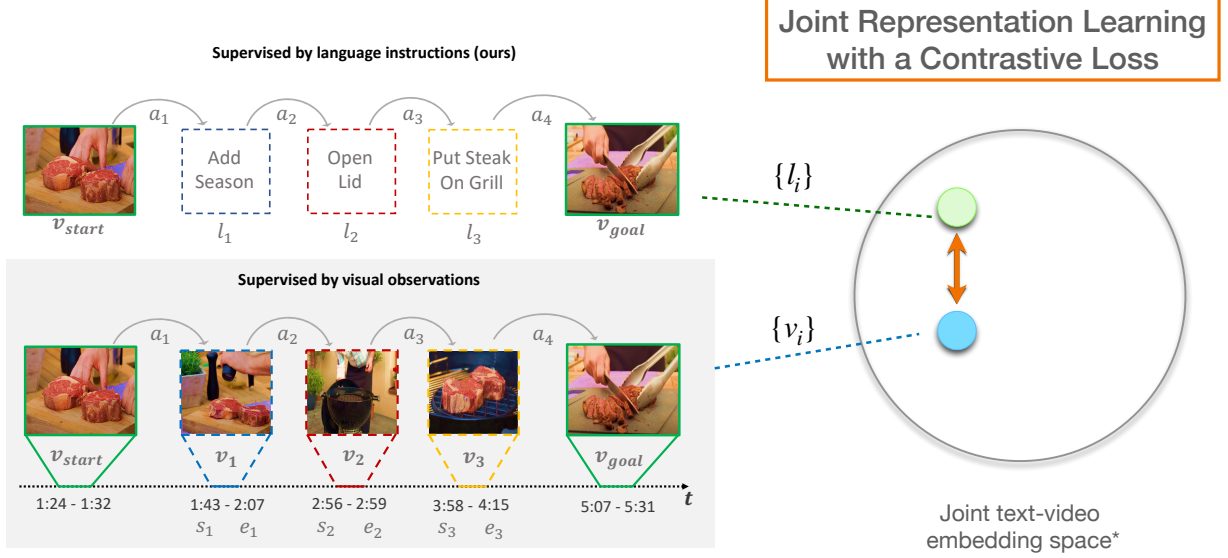


Figure 5.1: **(Left sub-figure)** Illustration of weak language supervision for procedure planning fully supervised approaches (bottom row) learn models from step labels, a_i , and intermediate visual representations, v_i , over T finite steps. This strategy requires knowing the starting, s_i , and ending, e_i , timestamps, for each intermediate step. In contrast, our approach (top row) exploits natural language representations, l_i , of the intermediate labels, a_i , as a surrogate supervision, which only requires labeling the order of events. Note that the action label, a_i , is a discrete variable, whereas the action language representation, l_i , is a pre-trained continuous embedding. **(Right sub-figure)** The learned joint embedding space of video and language. Representations (e.g., colored circles connected with $\{l_i\}$ and $\{v_i\}$) from each modality are clustered much closer if they share the same semantic meaning, e.g., the text of “cook steak” and its RGB image. In this way, features of different modalities become interchangeable.

limitation – reliance on strong supervision with expensive annotations [25, 13, 219]. Specifically, all such methods require access to (i) a list of action labels used to transition from start to the goal state and (ii) the visual representation of the intermediate states. Using such intermediate visual representations entails very expensive annotation of the start and end times of each intermediate instructional step; see Figure 5.1 (bottom). In contrast, our work removes the need for intermediate visual states during training and instead uses their linguistic representation for supervision. Relying on language representations allows us to better leverage instructional videos and significantly reduce the labeling effort; see Figure 5.1 (top). For example, language annotations for the intermediate instructional steps could be extracted from

general procedure descriptions available in recipes or websites, e.g., WikiHow [83]. In contrast, to obtain the timestamps for intermediate instructional steps one must watch the entire instructional video. In addition, a language representation can be a more stable supervisory signal [176], as the description of a given step (e.g., *add seasoning*) remains the same, while its visual observation varies across different videos.

Previous work on procedure planning from video relies on a two-branch auto-regressive approach while adopting different architectures to model these branches [25, 13, 219]. In such models, one branch is dedicated to predicting actions based on the previous observation, while the other approximates the observation given the previous action in a step-by-step manner. Such models are cumbersome and compound errors, especially for longer sequences. In contrast, we rely on a single branch non auto-regressive model, implemented as a transformer [227] that generates all intermediate steps in parallel conditioned on the start and goal observations. Finally, no previous effort has captured multiple possible procedure variants, i.e., given a start observation of a raw steak and a final observation of slicing a cooked steak (Figure 5.1), there are multiple ways to realize the goal from the start. Our approach deals with this important fact about real-world procedures through probabilistic modeling.

5.2 Related work

5.2.1 Procedure planning

Traditionally, goal-conditioned planning has been studied mostly in physical environments, e.g., robotic motion planning [95, 53, 61] and human pedestrian trajectory planning [158]. Recently, the task of procedure planning from instructional videos was introduced [25]. Various approaches have made use of RNNs [25], transformers [219] and adversarial policy planning [13]; all have used two-branches and strong visual supervision. In contrast, we model the actions directly using a non auto-regressive transformer-based architecture. More importantly, we use low-budget weak supervision in the form of language instructions instead of supervising the model with “costly” visual observations as done by all existing approaches.

5.2.2 Supervision with natural language

A common alternative to training visual models using manually defined label sets is to exploit semantic supervision from natural language. Using natural language as supervision has several advantages: (i) language annotations can be collected automatically [163]; (ii) modeling language and vision jointly can produce stronger representations [39]; (iii) such supervision can achieve better generalization to unseen domains [176]. Such benefits resulted in a growing interest in using language as supervision for variety of tasks, e.g., image classification [65, 94, 39], representation learning [162, 250], video retrieval [163, 57], step localization [41, 162] and navigation with instruction following [171]. We use advances in joint video and language modeling for procedure planning. We use pre-trained features [162] to map language and video to a common space and replace expensive video supervision with readily available language instructions

5.2.3 Sequence modeling with transformers

Procedure planning is a task of conditional sequence prediction and thus it directly benefits from recent advances in sequence modeling. One of the strongest recent approaches to sequence modeling is the transformer architecture [227], which has been adopted for a wide variety of tasks, e.g., image [6], video [9] and multi-modal data [147, 149, 250] tasks. Recent work adopted the transformer decoder architecture for fixed-size set prediction via learnable input queries [21, 262]. We build upon similar ideas by setting the first and last queries to correspond to the start and goal observations, while making the intermediate queries learnable. To improve long-range sequence modeling and help overall sequence coherence, recent work augmented transformers with an explicit external memory [127, 244]. For the same reasons, we also integrate a learnable memory module.

5.2.4 Future prediction

The task of procedure planning is closely related to future prediction, where the *start* and *goal* states are given. A key consideration in future prediction is modeling prediction uncertainty from partial initial observations. One common approach for modeling uncertainty is a

Variational Auto Encoder (VAE) [102] that captures the distribution over future actions. Another approach is to use generative adversarial networks (GANs) to forecast multiple, distinct and high quality future activities [267, 175]. In this work, we adopt the generative modeling framework to model distributions over possible plans.

5.3 Overall contribution: Probabilistic procedure planning via weak natural language supervision

Here, we present our approach to procedure planning that relies on three main components. First, we predict all steps in the plan in parallel using a non auto-regressive transformer decoder. To obtain coherent plan predictions, our transformer is augmented with a learned memory shared across all possible tasks in a given dataset (Section 5.3.2). Second, to model the uncertainty inherent to the task, we include a generative component trained with an adversarial loss. As a result, we can infer multiple feasible plans conditioned on start and goal observations (Section 5.3.3). Third, to supervise the transformer’s outputs, we use the cross-entropy loss on action predictions and a contrastive loss to match visual state predictions with the corresponding language descriptions (Section 5.3.4). Figure 5.2 provides an overview of our approach, which we detail next.

5.3.1 Problem formulation

Given a start visual observation, v_{start} , and a desired visual goal, v_{goal} , our task is to predict a *plan* defined as the sequence of T intermediate action steps, $\tilde{\pi} = \tilde{a}_{1:T}$, taken to transition from v_{start} to v_{goal} . We overscore with $\sim$ to indicate our predictions, while the lack of such overscoring indicates ground truth (GT). At training time, given v_{start} and v_{goal} , we predict a plan, $\tilde{\pi}$, and corresponding visual observations, $\tilde{v}_{1:T}$. We use the intermediate action labels, $a_{1:T}$, to train the plan prediction, $\tilde{\pi}$, and corresponding language descriptions (embedded with a pre-trained text encoder), $l_{1:T}$, to supervise the intermediate visual observations, $\tilde{v}_{1:T}$. That is, we substitute the visual information about intermediate instruction steps, $v_{1:T}$, with their language counterparts, $l_{1:T}$, to train the planner; see Figure 5.2. We believe such supervision substitution is meaningful, as we are using a strong pre-trained vision-language encoder [162],

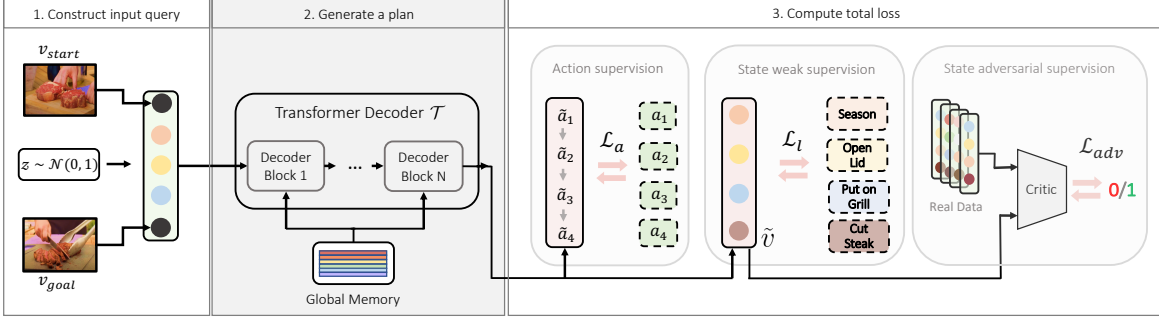


Figure 5.2: Overview of our pipeline for procedure planing. First, we embed the visual observations of the start and goal states (i.e., black nodes), attach them to the sequence of learned queries (i.e., colored nodes) and add random noise to the resulting input sequence. Second, we pass the input to the transformer decoder that interacts with the global memory to generate feasible procedure plans. Third, we produce state and action vectors, $\{\tilde{v}_i, \tilde{a}_i\}$, and use a number of losses, $\mathcal{L}$ (described in Sec 5.3.4), to supervise our architecture.

mapping visual activities and their descriptions in a common space, thus making visual, v_t , and language, l_t , features, corresponding to the same activity, interchangeable for training. In contrast, previous work assumes access to the set of intermediate action-observation pairs (i.e., $a_{1:T}, v_{1:T}$) [25, 219, 13], thereby requiring strong supervision to identify all intermediate visual observations. At inference time, we only use the start and goal observation, to predict a plan, $\tilde{\pi} = \tilde{a}_{1:T}$, for a given time horizon, T .

5.3.2 Memory augmented transformer decoder

To implement our planner, we use a non auto-regressive transformer decoder architecture [21, 262]. Our transformer decoder takes two input types; namely, learnable queries augmented with the start and goal observations and a learned memory component, and outputs action and intermediate state predictions, as illustrated in Figure 5.2.

Conditioned learned-query input. The first input to our approach is a set of queries, i.e., $Q = [q_{start}, q_1, \dots, q_{T-1}, q_{goal}]$, where the first and last inputs correspond to the representations of our initial and goal visual observations, v_{start} and v_{goal} , respectively, while $q_{1:T-1}$ are a set of learned queries. Queries, $[q_1, \dots, q_{goal}]$, are associated with the action labels, $a_{1:T}$, that we wish to predict. To communicate information about the order of elements to the decoder, we

add to each query a fixed cosine positional embedding [11], p_t , as follows

$$\mathbf{Q} = [q_{start} + p_0, \dots, q_t + p_t, \dots, q_{goal} + p_T], \quad (5.1)$$

where q_t and p_t all are encoded as d dimensional embeddings, (i.e., $q_t, p_t \in \mathbb{R}^d$) and $t = 1, \dots, T - 1$.

Learned memory input. The second input to our transformer decoder is a learned memory component that is common across all examples in a given dataset. The memory is defined as a set of d -dimensional vectors

$$\mathbf{M} = [m_1, m_2, \dots, m_n] \in \mathbb{R}^{d \times n}, \quad (5.2)$$

where n is the number of learnable vectors in the memory bank. Notably, the size of the memory (i.e., number of entries, n , in the memory) is a hyperparameter that is independent from the prediction time horizon. We use read-only memory [159] and share it among all layers for simplicity.

Memory-augmented transformer decoder. Our architecture is a stack of standard transformer decoder blocks [227] (see Figure 5.3), where each such block has access to the global learnable memory, (5.2). Specifically, the memory-augmented transformer block consists of two key operations. First, the input is processed with the self-attention operation. Second, the cross-attention module attends to the learnable memory to generate the output. The input to self-attention in the first transformer block corresponds to the query, $\mathbf{Q}$. All cross-attention blocks access the same memory, $\mathbf{M}$. Intuitively, the memory module can be seen as a collection of learnable plan embeddings shared across the entire dataset. Empirically, we show that the memory module plays a key role in our framework.

Our transformer decoder, $\mathcal{T}$, consists of a stack of N such memory-augmented blocks. We add two output heads (implemented as multi-layer perceptrons (MLPs)) at the final decoding layer. One head, h_a , yields intermediate predicted actions, $\tilde{a}_t$, while the second, h_v , yields corresponding intermediate visual representations, $\tilde{v}_t$, according to

$$\tilde{a}_{1:T} = h_a(\mathcal{T}(\mathbf{Q}, \mathbf{M})), \quad \tilde{v}_{1:T} = h_v(\mathcal{T}(\mathbf{Q}, \mathbf{M})), \quad (5.3)$$

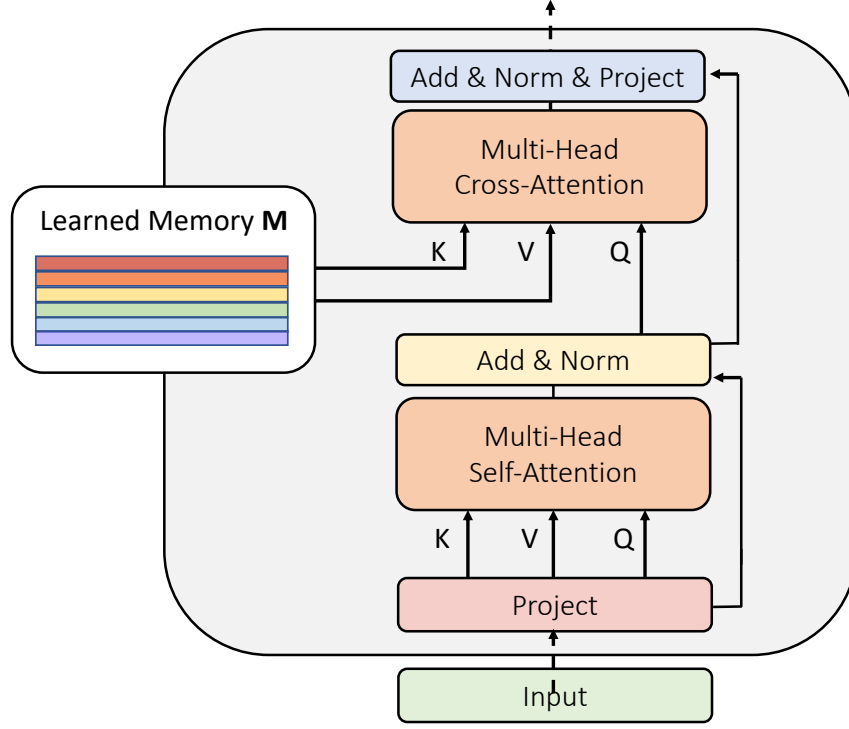


Figure 5.3: Illustration of the transformer block with self-attention and cross-attention with memory. The learned memory, $\mathbf{M}$, serves as an external memory bank and is globally shared across all transformer blocks. K, V and Q symbolize key, value and query, resp.

5.3.3 Adversarial generative modeling

To capture the uncertainty in prediction, where multiple plans from v_{start} to v_{goal} are plausible, we augment our model with a stochastic component using generative adversarial learning [67]. To make the generation process conditional on the input, we augment the entire query input, (5.1), with a random noise vector, $z \sim \mathcal{N}(0, 1)$, $z \in \mathbb{R}^{d'}$, through concatenation. The new query input sequence to our transformer, $\mathcal{T}$, thus becomes

$$\mathbf{Q}^z = \{[q_t; z] \mid q_t \in \mathbf{Q}\}. \quad (5.4)$$

We employ adversarial training wherein the generator, G , is trained to produce realistic action sequences, while the critic, C , provides the supervisory signal for training G [8]. In our case, we treat the memory-augmented transformer, $\mathcal{T}$, as the generator, G (i.e., $G = h_v(\mathcal{T}(\mathbf{Q}^z, \mathbf{M}))$), while the critic is modeled by a simple MLP. More precisely, we pass the output of our transformer, $\tilde{v}_{1:T}$, concatenated along the temporal dimension, to the critic, C ,

which outputs a value between 0 and 1, indicating its ability to discriminate between the predicted and ground-truth sequence, as depicted in Figure 5.2. Notably, to avoid the notorious issue of mode collapse associated with training GANs [213] (i.e., regardless of variations in random latent noise, z), we follow previous work [143, 254, 267] and include the normalized distance regularizing loss, as defined in Equation 3.15.

5.3.4 Training

To supervise our transformer, we rely on two complementary loss functions that enforce our transformer to decode the correct set of action labels in the procedure as well as corresponding visual representations. We also use an adversarial loss to train the stochastic component of our model.

Visual step supervision. One of the outputs of our model at training time is the sequence of visual features, corresponding to the procedure steps, $\tilde{v}_{1:T}$. To supervise the visual features with corresponding language features, $l_{1:T}$, we adopt contrastive learning [74]. For each feature, $\tilde{v}_t$, predicted by the transformer’s head, h_v , we use the corresponding ground-truth language embedding, l_t , as the positive example and all the other embeddings in the language vocabulary, $\{l_j\}$, as negative examples. The contrastive loss is calculated as

$$\mathcal{L}_l = - \sum_{t=1}^T \left[\log \frac{\exp(l_t \cdot \tilde{v}_t)}{\sum_j \exp(l_j \cdot \tilde{v}_t)} \right], \quad (5.5)$$

where $(\cdot)$ denotes the dot-product operator. Note, we use all examples in the language vocabulary as negatives as our vocabulary is typically small ($< 1\text{K}$ elements) and doing so allows for better training compared to per-batch negative sampling, cf. [79].

Action plan supervision. We also enforce the action prediction head, h_a , to output sequences of action probabilities, $\tilde{a}_t$, corresponding to ground-truth one-hot labels, a_t . For this purpose, we use the cross-entropy loss

$$\mathcal{L}_a = - \sum_{t=1}^T a_t \log \tilde{a}_t. \quad (5.6)$$

Adversarial supervision. To model uncertainty, we use adversarial training on the visual state predictions, $\tilde{v}_{1:T}$. The goal is to make predicted visual observation sequences indistinguishable from feature sequences composed of the ground truth language step description, $l_{1:T}$. We optimize the generator, G , (our transformer) and the critic, C , (an MLP) using an adversarial loss [8]

$$\mathcal{L}_{adv} = \min_G \max_C \mathcal{V}(G, C, \mathbf{Q}^z, \mathbf{M}), \quad (5.7)$$

where $\mathcal{V}$ is the standard GAN objective, defined as

$$\mathbb{E}_{l \sim p_{data}} [\log C(l)] + E_{z \sim p_z} [\log(1 - C(h_v(\mathcal{T}(\mathbf{Q}^z, \mathbf{M})))], \quad (5.8)$$

with $l \sim p_{data}$ and $z \sim p_z$ denoting the data distributions of the language representation and random noise, respectively.

Diversity regularizer. In addition to the contrastive and cross-entropy losses, we also use an adversarial loss to train the stochastic component of our model. To effectively capture the different modes present in the data and avoid the notorious mode collapse problem of GANs, we apply the recently proposed latent code normalized distance regularizing loss [254, 143, 258, 267] defined as

$$\mathcal{L}_{reg} = -\mathbb{E}_{z^1, z^2} \left[\frac{\|h_v(\mathcal{T}(\mathbf{Q}^{z^1}, \mathbf{M})) - h_v(\mathcal{T}(\mathbf{Q}^{z^2}, \mathbf{M}))\|_1}{\|z^1 - z^2\|_1} \right]. \quad (5.9)$$

Complete loss. Overall, our full loss function is defined as

$$\mathcal{L}(\theta) = \lambda_1 \mathcal{L}_l + \lambda_2 \mathcal{L}_a + \lambda_3 \mathcal{L}_{adv} + \lambda_4 \mathcal{L}_{reg}, \quad (5.10)$$

where θ refers to the parameters associated with all learnable modules, i.e., queries, memory module, as well as the transformer decoder and discriminator parameters and $\lambda_{1:4}$ are empirically determined loss weights.

5.3.5 Inference

At inference time, we use our transformer as a generative model to sample multiple procedure plans, $\tilde{\pi}^k = \tilde{a}_{1:T}^k$, for the same input start and goal observations. This operation is achieved by drawing $\mathcal{K}$ latent noise vectors, z^k , and forwarding them through our transformer, $\mathcal{T}$, conditioned on a single start-goal observation, as follows

$$\tilde{\pi}^k = h_a(\mathcal{T}(Q^{z^k}, \mathbf{M})), z^k \sim \mathcal{N}(0, 1), \quad (5.11)$$

for $k = 1, \dots, \mathcal{K}$.

To obtain a probability distribution over actions at each timestep, t , of the plan, given by our model, we calculate action frequencies as follows:

$$\bar{\Pi} = \bar{a}_{1:T} = \frac{1}{\mathcal{K}} \sum_{k=1}^{\mathcal{K}} [\tilde{a}_1^k, \dots, \tilde{a}_T^k]. \quad (5.12)$$

Given that a_t^k are one-hot vectors, each $\bar{a}_t$ results in a marginal distribution over actions at a specific timestep, t .

Most standard benchmark metrics for procedure planning, such as Success Rate, Accuracy or IoU (see Section 5.4.2), require a single action sequence output for evaluation. To compute the most probable action sequence, induced by our action distribution, $\bar{\Pi}$, we use the *Viterbi* algorithm [231], as commonly seen in sequential labeling work [188, 106, 105, 187].

Traditionally, the Viterbi algorithm is used to find the most probable path in the first-order Markov model of a dynamic system. For a sequence of length T and N_a possible states of the system, the Viterbi algorithm relies on two main inputs: (i) a transition matrix, $A \in \mathbb{R}^{N_a \times N_a}$, capturing the probability of transitioning from one state, a_i , to another, a_j , (ii) an emission matrix, $B \in \mathbb{R}^{T \times N_a}$, describing the probability of each state, a_i , given a set of observations. In our work, we use the marginal state probabilities for each timestep as the emission matrix, B , and estimate the transition matrix, A , directly from the ground-truth plans. Specifically, the transition matrix, A , is calculated based on the action co-occurrence frequencies in the training data. To calculate the value of $A_{i,j}$, we must find the number of times action a_i is followed by action a_j in the ground-truth plans. We then normalize each row of A to sum to 1, by applying L_1 -normalization followed by a softmax with temperature $\tau = 1$.

5.4 Empirical evaluation

In this section, we evaluate our overall model in comparison to alternative approaches and also evaluate the role of each module of the model. We include evaluation on the largest labeled instructional video dataset, which has not been used previously for the task of procedure planning due to the need of strong supervision in previous work (Section 5.4.3). We also provide prediction uncertainty evaluation in procedure planning for the first time, which sheds light on our approach and the task of planning itself (Section 5.4.4).

5.4.1 Implementation details

Our planner operates on video and language features, pre-extracted by a model trained for joint video-text embedding [162] using the HowTo100M [163] dataset and self supervision. We use a memory-augmented transformer with two layers and eight heads, and optimize it for 200 epochs with ADAM [101] on a single V100 GPU.

Our model uses pre-extracted language and vision features, with a model trained on the HowTo100M dataset for joint text-video embedding [163]. This backbone model embeds both vision and language inputs into 512 dimensional features. For our model, we use a transformer decoder [227] with eight heads, two layers and 128 dimensional hidden states. Note that we provide an ablation on our architecture choices in Section 5.4.5. Since our pre-trained features are of dimension 512, we embed our initial features using a multilayer perceptron (MLP) with shape $[512 \rightarrow 256 \rightarrow 128]$ interspersed with ReLU nonlinearities. A similar MLP is used to project the ground-truth language features, l_i , to the same dimension. For the memory module, we empirically set the number of memory entries, n , to 128. For the critic, C , we use another three layer MLP with shape $[256 \rightarrow 64 \rightarrow 32]$ with ReLU activations in all layers. The dimension of the noise vector, z , is empirically set to 32. We train our model for 200 epochs with an initial learning rate set to 7×10^{-4} and decayed by 0.65 every 40 epochs. The best performing model on the validation set is used to report the final test set results.

5.4.2 Evaluation protocol

Datasets. For evaluation, we use three different instructional video datasets, namely, CrossTask [274], the Narrated Instructional Videos datasets [5] (NIV) and COIN [224]. CrossTask contains 2750 videos, depicting 18 different procedures and an average of 7.6 actions/video; the NIV dataset is much smaller with 150 videos, 5 procedures and 9.5 actions/video on average. COIN is the largest dataset in our evaluation. It contains 11827 videos, 778 procedures and 3.6 actions/video. Depicted procedures vary widely, e.g. *Make Taco Salad* and *Change Car Tire*.

Data curation. Following previous work [25, 13], for each instructional video, we start by extracting the *ordered* list of all actions, $a_{1:N}$, present in the video. We also collect language descriptions, $l_{1:N}$, corresponding to each action, a_i . In addition, to obtain various start and goal observations, we also extract corresponding visual observations, $v_{1:N}$, by locating the start and end times of each action following previous work [25, 13]. *Note that this information is only used for data curation and not for training as done in previous work.* Each video is therefore described as the ordered set of tuples

$$V = [(a_1, l_1, v_1), \dots, (a_N, l_N, v_N)], \quad (5.13)$$

where N is the total number of actions present in the video.

Given the video, V , represented with (5.13), we curate a set of plans with prediction time horizon T by sliding a window of size $T + 1$, such that $a_{t+1:t+T}$ and $l_{t+1:t+T}$ represent the set of action plans we need to predict, whereas v_t and v_{t+T} represent v_{start} and v_{goal} , respectively.

Metrics. Following previous work [25, 13, 219], we evaluate the performance using three increasingly strict metrics. (i) mean Intersection over Union (mIoU) treats the predicted and ground-truth action sequences as sets, and measures the overlap between these sets. mIoU is agnostic to the order of actions and only indicates whether the model captures the correct set of steps needed to complete the plan. (ii) mean Accuracy (mAcc) performs element-wise comparisons between the predicted and ground-truth action sequences, thereby considering the order of the actions as well. (iii) Success Rate (SR) considers a plan successful only if it *exactly* matches the ground truth.

Protocols. There exist two different evaluation protocols on CrossTask, namely “Protocol 1”

and “Protocol 2”. In all our experiments, save one, we follow previous work [25, 13] and use “Protocol 1”, which relies on the sliding window based data curation procedure described above. In addition, “Protocol 1”, uses a 70/30 train/test dataset split. For the long horizon planning experiments on CrossTask, we also adopt an additional protocol, proposed in recent work [219] and referred to as “Protocol 2”. The main differences of “Protocol 2” can be summarized in the following three points: (i) Instead of relying on a sliding window to consider all procedure plans of time horizon T for each video, “Protocol 2” randomly selects one procedure plan of horizon T per video. (ii) “Protocol 2” uses 85/15 train/test split. (iii) “Protocol 2” predicts actions, $a_{1:T-1}$, which concretely means that, for a prediction horizon, T , it actually makes $T - 1$ predictions. For COIN and NIV datasets, we use only “Protocol 1” mentioned earlier in this paragraph, as their evaluation protocols.

Probabilistic metrics. We also evaluate the stochastic nature of our model by measuring the following probabilistic metrics: (i) The Kullback–Leibler (KL) divergence between our predicted plan distributions and ground truth; (ii) how well the ground-truth modes are covered by our results (Mode Recall); (iii) how often our plans correspond to the ground-truth mode (Mode Precision). To this end, for each $\{\text{start}, \text{goal}\}$ observation, we draw ($\mathcal{K} = 1500$) samples from our generative model and explicitly approximate a distribution, as described in Section 5.3.5. For completeness, we also evaluate using more standard probabilistic prediction metrics, including Negative Log Likelihood (NLL) and the cosine distance [32, 161, 267].

Baselines. Here, we provide a more detailed description of the procedure planning comparison approaches considered below. Unlike in previous chapters, we provide these details on alternatives because some have not been used previously for the task at hand (e.g., Random and Retrieval-Based).

- **Random.** As an elementary baseline, we randomly select action steps from the entire vocabulary with equal probability (i.e., uniform distribution) for evaluation.

- **Retrieval-Based.** For each start and goal observation pairing, $\{v_{start}, v_{goal}\}$, in the test set, this method retrieves the nearest neighbor in the train-set based on visual feature similarity. The plan labels associated with the retrieved nearest neighbor is used for evaluation.

- **WLTD0** [42] and **UAAA** [1]. These two frameworks use recurrent neural networks (RNN) [191] for action planning from the visual observation input.

Models	Supervision	$T = 3$			$T = 4$		
		SR $\uparrow$	mAcc $\uparrow$	mIoU $\uparrow$	SR $\uparrow$	mAcc $\uparrow$	mIoU $\uparrow$
Random	-	<0.01	0.94	1.66	<0.01	0.83	1.66
Retrieval-Based	-	8.05	23.30	32.06	3.95	22.22	36.97
WLTD0 [42]	-	1.87	21.64	31.70	0.77	17.92	26.43
UAAA [1]	-	2.15	20.21	30.87	0.98	19.86	27.09
UPN [212]	V	2.89	24.39	31.56	1.19	21.59	27.85
DDN [25]	V	12.18	31.29	47.48	5.97	27.10	48.46
Ext-GAILw/o Aug. [13]	V	18.01	43.86	57.16	-	-	-
Ext-GAIL [13]	V	21.27	49.46	61.70	16.41	43.05	60.93
Ours	L	23.34	49.96	73.89	13.40	44.16	70.01

Table 5.1: Evaluation of procedure planning results on CrossTask for prediction horizon $T \in \{3, 4\}$. The column name *Supervision* denotes the type of state supervision applied in training, with V and L denoting visual and language state representation, resp.

- **Universal Planning Networks (UPN)** [212]. UPN is a physical-world path planning algorithm with a continuous action policy. We follow previous work [25] to modify it via appending a softmax function for the discrete action space.

- **Dual Dynamics Networks (DDN)** [25]. DDN models the state-action transition of procedural plans with a two-branch auto-regressive model. The state branch is modeled with an MLP, while the action branch uses an RNN.

- **PlaTe** [219]. This method simply replaces the models used in the two-branch model of DDN with autoregressive transformer modules.

- **Ext-GAIL** [13]. This method uses reinforcement learning techniques for procedure planning and augments the DDN architecture with a context latent variable.

5.4.3 Comparison to alternative approaches

CrossTask (short-horizon). Table 5.1 compares our weakly-supervised approach to a number of alternatives, including the fully-supervised state of the art, across the two prediction horizons typically reported in this task. Our results are consistently better, except for the success rate (SR) at $T = 4$, where we are the second best approach. The performance improvement in SR is especially striking at short-term horizon, $T = 3$, where we outperform the previous best (i.e., Ext-GAIL) by more than 2%, while using weaker supervision. Notably, Ext-GAIL achieves its level of performance via data augmentation, which allows it to

have 30% more training data. In a more similar setup (i.e., when Ext-GAIL does not use data augmentation) the performance gain of our method over “Ext-GAIL w/o Aug” is 5.3%. Importantly, our results are obtained with weaker supervision, which speaks decisively in favor of our approach.

We also notice a larger gain in mIoU compared to previous work, i.e., 73.89% versus 61.70% for $T = 3$ and 70.01% versus 60.93% for $T = 4$. This result suggests that our approach is better at capturing feasible action steps than other approaches (e.g., never producing *pour water* when input observations are related to *making a salad*). We hypothesize that this performance is enabled by the language-based contrastive learning, which is more effective at clustering latent representations than its vision counterpart. For example, while some visual observations can look similar (e.g., *pour water* and *add oil*), the distinction between the two is clearer in natural language. Finally, the level of improvements in mIoU and mAcc is typically higher than the gain in SR. We attribute this result to the uncertainty inherent to the task (i.e., multiple feasible plans for the same start and goal observations). We explore this aspect in greater detail in Section 5.4.4. Moreover, thanks to the non-autoregressive nature, we are **4** times faster at inference, e.g. **6.75ms** (ours) versus **27.34ms** DDN [25] on CrossTask for $T = 3$.

CrossTask (long-horizon). We now evaluate our model’s ability to predict plans for longer time horizons (i.e., $T \in \{3, \dots, 6\}$). We compare to previous approaches that reported results on such horizons. There are two different protocols for these settings. (i) Protocol 1 [25] and (ii) Protocol 2 [219]; see details in Section 5.4.2. For a fair comparison, we present our results using both protocols in Table 5.2 and show that our approach is the most effective on both.

NIV. Following previous work [13], we also evaluate our model on the smaller NIV dataset. The results in Table 5.3 are consistent with our results on CrossTask. Once again, our approach is the top performer across all metrics. This result suggests our language supervision works for smaller datasets as well.

COIN. To show the capability of our method to scale, we evaluate our method on the largest labeled instructional video dataset (i.e., COIN). As we are the first ones to perform procedure planning on such a large-scale dataset, there is a lack of comparison methods. Therefore, we follow existing work [25] and include three baselines: (i) Random selection; (ii) Retrieval-based; (iii) and a re-implementation of the DDN model [25], using our video features, de-

	$T = 3$	$T = 4$	$T = 5$	$T = 6$
Models	SR $\uparrow$	SR $\uparrow$	SR $\uparrow$	SR $\uparrow$
Retrieval-Based	8.05	3.95	2.40	1.10
DDN [25]	12.18	5.97	3.10	1.20
Ours (Protocol 1)	23.34	13.40	7.21	4.40
PlaTe [219]	18.5	14.0	10.0	7.5
Ours (Protocol 2)	24.4	15.8	11.8	8.3

Table 5.2: Success Rate evaluation of procedure planning results on CrossTask [274] that extends to longer prediction horizon, T .

Horizons	Methods	Sup.	NIV			COIN		
			SR $\uparrow$	mAcc $\uparrow$	mIoU $\uparrow$	SR $\uparrow$	mAcc $\uparrow$	mIoU $\uparrow$
$T = 3$	Random	-	2.21	4.07	6.09	<0.01	<0.01	2.47
	Retrieval	-	-	-	-	4.38	17.40	32.06
	DDN [25]	V	18.41	32.54	56.56	13.9	20.19	64.78
	Ext-GAIL [13]	V	22.11	42.20	65.93	-	-	-
	Ours	L	24.68	49.01	74.29	15.4	21.67	76.31
$T = 4$	Random	-	1.12	2.73	5.84	<0.01	<0.01	2.32
	Retrieval	-	-	-	-	2.71	14.29	36.97
	DDN [25]	V	15.97	27.09	53.84	11.13	17.71	68.06
	Ext-GAIL [13]	V	19.91	36.31	53.84	-	-	-
	Ours	L	20.14	38.36	67.29	11.32	18.85	70.53

Table 5.3: Procedure planning results on NIV [5] and COIN [224] for prediction horizon $T \in \{3, 4\}$. The column *Sup.* denotes the type of state supervision applied in training.

scribed in Section 5.4.1. Table 5.3 shows that our model equipped with language supervision consistently outperforms the baselines, even strongly-supervised ones.

5.4.4 Evaluation of probabilistic modeling

To evaluate our probabilistic modeling, we compare plan distributions produced by our model, with the ground-truth distribution over feasible plans. We begin our evaluation of probabilistic modeling on CrossTask as it is the most suitable dataset in terms of variations in the set of feasible plans, e.g., avg. 6.76 different paths for $T = 4$. Subsequently, we provide analogous evaluation using the COIN and NIV datasets.

Plan distribution modeling. Our approach is probabilistic by design as described in Secs. 5.3.3 and 5.3.5. To establish a deterministic baseline, we train our model without the adversarial loss

Metric ↓	Method	$T = 3$	$T = 4$	$T = 5$	$T = 6$
KL-Div	Ours - deterministic	2.31	4.47	6.30	8.81
	Ours - probabilistic	2.11	3.50	4.26	6.89
NLL	Ours - deterministic	5.13	6.25	6.68	8.49
	Ours - probabilistic	4.89	5.48	6.24	7.67

Table 5.4: Evaluation of the plan distributions produced by our probabilistic approach versus the deterministic variant on CrossTask.

Metric ↑	Method	$T = 3$	$T = 4$	$T = 5$	$T = 6$
ModePrec	Ours - deterministic	27.61	17.21	7.41	4.97
	Ours - probabilistic	36.61	18.55	12.48	6.58
ModeRec	Ours - deterministic	56.24	37.33	18.38	8.85
	Ours - probabilistic	66.13	46.56	26.46	12.67
CosDist	Ours - probabilistic	0.384	0.302	0.2471	0.1658

Table 5.5: Evaluation of diversity and accuracy of our samples with respect to ground truth on CrossTask. Our approach improves *both* ModePrec and ModeRec metrics. We further provide averaged pair-wise cosine distance as another indicator for diversity. CosDist for the deterministic model is not provided, since it only produces a single result with no pairs to compare.

and fix the latent noise vector, $z = 0$, both during training and testing. To build the ground-truth distribution over goal-conditioned plans, we retrieve all action sequences of length T in the test set that share the given start and goal state. The plan distribution of our probabilistic model (conditioned on the start and goal observations) is obtained by sampling $\mathcal{K} = 1500$ different action sequences, as discussed in Section 5.3.5. For the deterministic baseline, the model produces only one plan (i.e., $\mathcal{K} = 1$). In all cases, the probability of a plan is defined as its frequency in the obtained sample set. To evaluate the quality of the predicted plans, we measure the (dis-)similarity between plan distributions produced by each model (i.e., ours vs. the deterministic variant) and that of the ground truth using the KL divergence and NLL. Table 5.4 shows that our probabilistic approach better matches the ground-truth plan distribution (i.e., it has lower KL and NLL). These results come about because our model is able to sample multiple *valid* plans with respect to the test set distribution, as opposed to the deterministic model that considers a single feasible plan.

Sample diversity and mode coverage. For given start and goal observations, our approach produces multiple plan hypotheses, using probabilistic sampling described in Section 5.3.5.

Metric ↓	Method	$T = 3$	$T = 4$	$T = 5$	$T = 6$
NLL	Ours - deterministic	6.37	6.38	7.71	8.95
	Ours - probabilistic	6.37	6.37	7.79	8.84
KL-Div	Ours - deterministic	5.34	6.88	6.75	7.06
	Ours - probabilistic	5.75	6.68	6.84	7.15

Table 5.6: Evaluation of the plan distributions produced by our probabilistic approach versus the deterministic variant on COIN.

In this section, we measure the diversity of our samples and their relation to the ground-truth plans distribution. To characterize the ground-truth distribution, we define ground-truth modes as the set of unique action sequences in the test set that share the same start and goal state. To calculate the relation of our samples to the ground-truth modes we define two metrics, Mode Recall (ModeRec) and Mode Precision (ModePrec). ModeRec reflects how well the GT modes are covered by our model and is calculated as the average number of GT modes captured by at least one sample from our model. In complement, ModePrec measures how often a sampled plan is feasible according to the test data; it is calculated as the average number of samples that match at least one GT mode. Intuitively, ModeRec and ModePrec measure not only how diverse the samples from our model are, but also how useful this diversity is with respect to the GT. By measuring the average cosine distance among model samples, i.e., CosDist — a metric widely used in the GAN literature — we also show that our samples are diverse agnostic to the data distribution. The results in Table 5.5 suggest our probabilistic approach can produce both diverse and accurate plans, where it is superior on all metrics.

Probabilistic modeling on COIN and NIV. We now further evaluate the probabilistic modeling abilities of our approach on COIN (in Table 5.6) and NIV (in Table 5.7). In particular, we again consider the KL divergence and NLL measures to quantify our model’s ability to capture the plan distributions in those datasets. Interestingly, different from the results ob-

Metric ↓	Method	$T = 3$	$T = 4$	$T = 5$	$T = 6$
NLL	Ours - deterministic	7.18	7.79	8.49	9.17
	Ours - probabilistic	7.07	7.81	8.36	9.25
KL-Div	Ours - deterministic	5.35	5.60	5.96	8.23
	Ours - probabilistic	4.92	5.76	6.28	9.51

Table 5.7: Evaluation of the plan distributions produced by our probabilistic approach versus the deterministic variant on NIV.

Datasets	$T = 3$	$T = 4$	$T = 5$	$T = 6$
CrossTask	3.26	6.76	8.40	9.29
COIN	1.51	1.93	2.25	2.35
NIV	1.03	1.07	1.28	1.29

Table 5.8: The average number of unique paths that share the same start and goal across multiple horizons and datasets.

tained on CrossTask (i.e., Table 5.4), we observe that there is negligible difference between our approach and the deterministic counterpart on CrossTask, COIN and NIV. We suspect that this happens because the NIV and COIN datasets are lacking variability in goal-conditioned plans. To verify that hypothesis, we conduct a quantitative evaluation of such variability on CrossTask, COIN and NIV. In Table 5.8, we show the average number of distinct plans that can connect the same start and goal observations for each dataset and prediction horizon. It is clear that the plans in CrossTask are much more diverse than those present in the other two datasets. In particular, CrossTask has the largest average number of distinct plans for each time horizon. Notably, for all datasets, it seems longer horizons tend to have larger variability. We believe that this is reasonable because any variations of intermediate steps would result in different paths, which highlights the importance of adopting a probabilistic point of view for procedure planning.

5.4.5 Ablation studies

Impact of different loss functions. We evaluate the role of each loss component by gradually introducing each objective. The results in Table 5.9 show the pivotal role of language-based supervision, as evidenced by increased performance across all metrics. The complementarity of the three objectives is shown in an incremental manner. Note that improvements from the adversarial loss might seem minor (e.g., ≈ 1 in success rate for three datasets), as it only compares a single prediction to a single ground-truth plan. We show its strict superiority to deterministic models for modelling distribution of procedures in Section 5.4.4.

Impact of Viterbi post-processing. Throughout the empirical results, we use the Viterbi algorithm on top of the predicted action probabilities, $\bar{\Pi}$ (5.12), to produce the optimal plan at inference time. Notably, the Viterbi post-processing is optional, and one may directly use the

Datasets	Loss Objective	SR $\uparrow$	mAcc $\uparrow$	mIoU $\uparrow$
CrossTask	L_a	16.90	44.20	57.56
	$L_a + L_l$	22.12	45.57	67.40
	$L_a + L_l + L_{adv}$	23.34	49.96	73.89
	w/o <i>Viterbi</i>	22.66	45.95	67.52
COIN	L_a	8.48	12.19	68.15
	$L_a + L_l$	14.41	20.25	73.49
	$L_a + L_l + L_{adv}$	15.40	21.67	76.31
	w/o <i>Viterbi</i>	14.18	21.01	75.62
NIV	L_a	17.81	42.35	69.42
	$L_a + L_l$	24.05	46.67	73.89
	$L_a + L_l + L_{adv}$	24.68	49.01	74.29
	w/o <i>Viterbi</i>	20.18	47.73	73.09

Table 5.9: Performance of our model trained with different loss functions on three datasets. The last row of each block represents the results of the total loss but without the *Viterbi* algorithm.

set $\tilde{\pi}^k$ to produce the final plan by simply selecting the mode across the set. Comparing the last two rows in each block of Table 5.9 shows the added advantage of using Viterbi to model the optimal action order in procedure plans explicitly, for all datasets. Notably, Viterbi post-processing is especially helpful on the NIV dataset. We hypothesize that the data scarcity in NIV results in a weaker predictive model at training; therefore, explicitly modeling the optimal transition between the actions with Viterbi plays a more important role in this case.

Impact of memory configuration. We also include an ablation evaluating the adopted memory augmented transformer decoder. Table 5.10 shows that the size of our memory plays a key role in our architecture. Indeed, excluding the memory component yields the worst results, while too large a memory degrades performance. These results suggest that the memory component helps capture dataset content, where it yields stronger results when the number of memory entries is large enough to properly span the actions present in the entire dataset. Notably, while tuning the memory size for each dataset might yield better results, for simplicity, we elect to use the best setting of CrossTask for all datasets.

Impact of the regularization loss. In this experiment we further evaluate our model’s capability to model plan distributions. For this purpose we follow the evaluation procedure adopted in Section 5.3.4. The effectiveness of the diversity regularization loss, (5.9), for our approach is studied in Table 5.11. It is clear that training without the regularization leads to worse NLL and KL-div values. These results confirm the important role of the regularization

Memory Size	SR $\uparrow$	mAcc $\uparrow$	mIoU $\uparrow$
0	7.49	22.76	31.33
64	16.30	43.62	55.66
128	23.34	49.96	73.89
256	20.81	44.61	59.70

Table 5.10: Ablation study on the impact of external memory sizes for prediction horizon, $T = 3$, with CrossTask. All results are obtained using our transformer, with two layers and eight heads.

Metric $\downarrow$	Method	$T = 3$	$T = 4$	$T = 5$	$T = 6$
NLL	Ours - prob. w/o $\mathcal{L}_{reg}$	5.07	6.61	7.57	8.28
	Ours - prob.	4.89	5.48	6.24	7.67
KL-Div	Ours - prob. w/o $\mathcal{L}_{reg}$	2.37	4.45	6.49	7.74
	Ours - prob.	2.11	3.50	4.26	6.89

Table 5.11: The effect of diversity regularization loss on the produced plans in the CrossTask dataset.

in the probabilistic model, which indeed seems to suffer from mode dropping in the absence of such regularization.

Impact of the Viterbi post-processing. In Table 5.9, we have shown that using Viterbi post-processing, while relying on our predicted plan distribution, $\bar{\Pi}$, as an emission matrix, leads to overall better plan predictions. Here, we further study the role of emission, B , versus transition, A , matrices, in the Viterbi-based plan inference. In particular, to show the importance of using our predicted distribution, $\bar{\Pi}$, as an emission matrix, we substitute it with a uniform matrix, *i.e.*, a matrix with all values equal to $\frac{1}{N_a}$, and perform plan inference with a correspondingly modified Viterbi algorithm. In other words, in these settings, the plan inference will be driven purely by the transition matrix, A . We also experiment with setting A to the uniform matrix, and only using B to drive Viterbi’s inference. Table 5.12 shows that our original formulation gives the best results and highlights the importance of estimating the emission and transition matrices properly from data, as handled by our model.

Impact of the size of the sample set. For the results shown elsewhere in this chapter, we provide inference results obtained by sampling $\mathcal{K} = 1500$ procedure plans from our probabilistic model. Here, we ablate this parameter and show its influence on how well we fit the

Emission matrix	SR $\uparrow$	mAcc $\uparrow$	mIoU $\uparrow$
B Uniform	0.12	7.25	<0.01
A Uniform	22.61	46.12	70.24
Ours	23.34	49.96	73.89

Table 5.12: The effect of estimating transition matrix, A , and emission matrix, B , of *Viterbi* post-processing from data. The results are for prediction horizon, $T = 3$, on the CrossTask dataset.

Metric $\downarrow$	$\mathcal{K}$	$T = 3$	$T = 4$	$T = 5$	$T = 6$
NLL	150	5.03 ± 0.21	5.85 ± 0.31	6.78 ± 0.25	8.73 ± 0.43
	500	4.91 ± 0.11	5.71 ± 0.20	6.57 ± 0.13	8.33 ± 0.27
	1500	4.89 ± 0.10	5.48 ± 0.11	6.24 ± 0.09	7.67 ± 0.18
	2500	4.89 ± 0.04	5.45 ± 0.06	6.22 ± 0.05	7.63 ± 0.08
KL-Div	150	2.51 ± 0.31	4.20 ± 0.44	4.66 ± 0.29	7.23 ± 0.14
	500	2.48 ± 0.24	3.58 ± 0.12	4.51 ± 0.20	6.94 ± 0.10
	1500	2.11 ± 0.10	3.50 ± 0.06	4.26 ± 0.08	6.89 ± 0.03
	2500	2.01 ± 0.06	3.27 ± 0.02	4.18 ± 0.02	6.83 ± 0.01

Table 5.13: Ablation study of the number of samples, $\mathcal{K}$, used by our approach for probabilistic inference. We show NLL and KL-Div results with corresponding variances obtained from 20 runs.

ground-truth plan distribution. Table 5.13 shows that larger value of $\mathcal{K}$ generally leads to better results as it can better approximate the true distribution. However, it should be noted that larger value of $\mathcal{K}$ require more computation. We therefore chose $\mathcal{K} = 1500$ for our experiments as a reasonable trade-off between performance and computation cost.

Impact of transformer architecture. Here, we examine the impact of the transformer architecture on the plan prediction performance. To save on the computations, we limit this study to a prediction horizon of $T = 3$. Table 5.14 shows the performance of alternative transformer configurations with different number of layers and/or heads. The final model used elsewhere in this chapter, with two layers and eight heads, performs best on CrossTask.

Alternative uncertainty baselines. Our approach captures the uncertainty of plans by learning a stochastic model (GAN), which allows it to produce distinct procedure plans when seeded with different random noise vectors. On the other hand, one can follow a more naive approach and obtain diverse outputs from a model trained deterministically (i.e., no random-

layers/heads	SR $\uparrow$	mAcc $\uparrow$	mIoU $\uparrow$
2/1	17.81	40.67	70.85
2/4	20.54	48.32	73.05
2/8	23.34	49.96	73.89
3/4	19.21	45.51	72.47
3/8	14.85	39.48	69.72

Table 5.14: Ablation study on the number of layers and heads for prediction horizon, $T = 3$, with CrossTask.

Datasets	SR $\uparrow$	mAcc. $\uparrow$	mIoU $\uparrow$	KL-Div $\downarrow$	NLL $\downarrow$	MCPrec $\uparrow$	MCRec $\uparrow$
Ours-prob. (HowTo100M pretrained)	23.34	48.96	73.89	2.11	4.89	36.61	66.13
Ours-prob. (CrossTask pretrained)	10.82	35.11	58.32	3.78	5.56	17.19	12.40
Ours-deter+noise ($\sigma = 1$)	20.02	44.76	72.96	2.37	5.77	24.74	67.21
Ours-deter+noise ($\sigma = 3$)	10.94	42.67	63.63	3.53	7.51	4.290	58.96
Ours-deter+noise ($\sigma = 5$)	5.83	35.53	55.43	5.52	8.53	1.169	37.01
Ours-deter+dropout ($p = 0.1$)	22.27	45.43	73.40	2.34	5.49	33.79	44.17
Ours-deter+dropout ($p = 0.3$)	22.04	43.64	70.97	2.46	5.83	30.61	55.32
Ours-deter+dropout ($p = 0.5$)	20.77	44.89	70.21	2.73	6.40	24.08	64.61
Ours-prob. (Strong Sup.)	24.41	45.17	73.83	2.12	4.71	36.89	62.69

Table 5.15: (top rows) Results using features extracted from a model pre-trained on HowTo100M vs. features finetuned on CrossTask. (middle rows) Ablation study on two alternative ways for uncertainty modeling. (last row) Strong visual supervision results. **Blue** indicates the top performer and **red** the 2nd-best result.

ness at training), by adding random noise to the system only at inference, e.g., injecting random noise in the input or using dropout at inference. Here, we compare our GAN-based formulation to the aforementioned naive approaches for introducing diversity in the predicted plans; results are summarized in Table 5.15. In particular, we evaluate adding Gaussian random noise of different intensity to the input, i.e., $N(0, \sigma)$, and applying dropout (of probability p) to the hidden representation of the network at inference time. We see that both baselines can increase diversity at the cost of accuracy. The best naive configuration is Ours-deter+dropout ($p = 0.3$), which is still inferior to our full GAN-based approach on both deterministic and probabilistic metrics. This result shows the importance of learning the uncertainty model during training, as opposed to simply adding it at inference, confirming that the GAN is useful for modelling uncertainty in our scenario.

Domain shift in joint video-text embedding. In this study, we choose video+language features pretrained on HowTo100M because recent work (e.g., [163] show state-of-the-art results

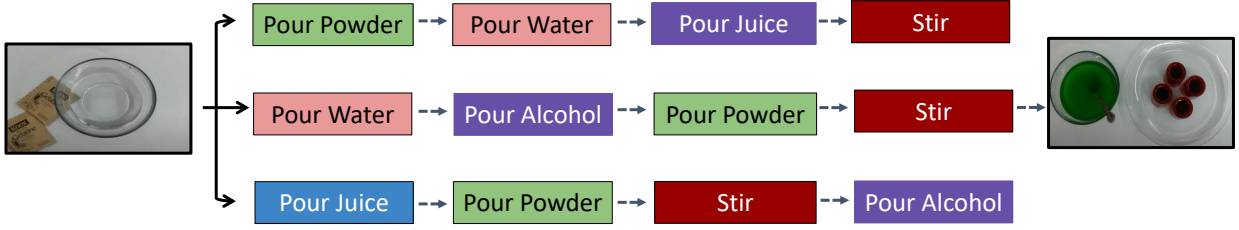


Figure 5.4: Sample *plausible* plans (seen in the test set) for *making beverage* produced by our probabilistic model, for the same {start, goal} observations.

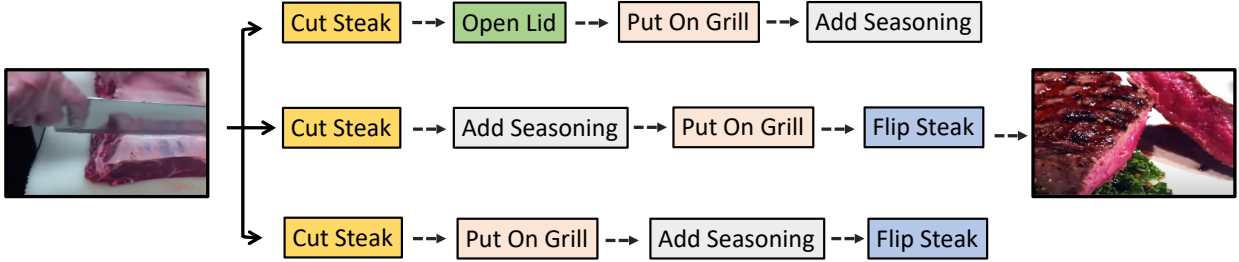


Figure 5.5: Sample *plausible* plans (seen in the test set) for *grilling steak* produced by our probabilistic model, for the same {start, goal} observations.

with such features on the datasets we are experimenting with (i.e., [224, 274]). Still, it is interesting to consider using a dataset’s “native” features as a variation. For this experiment, we use the CrossTask dataset, employ the video features trained on it (provided with the dataset) and train language embeddings (not provided by the dataset) by ourselves; we finetune the language model to align with CrossTask’s video features, using contrastive learning. These features are then used to train our model as described in Section 5.3.4. Table 5.15 (top two rows) shows that using the “native” CrossTask features is still inferior to the large-scale pre-training on HowTo100M [163]. This result could be due to the relative scarcity of data in CrossTask that is insufficient to train strong language+vision features.

Comparison to strong supervision. Our framework can also be trained with strong supervision, *i.e.*, simply swapping the language supervision signals, $\{l_i\}$, with visual ones, $\{v_i\}$. We show the results experimented in this setting in Table 5.15 (bottom row). Accessing the full annotations can modestly improve or hurt the performance, depending on the metric. Thus, our weakly supervised model performs comparably, while being cheaper to train.

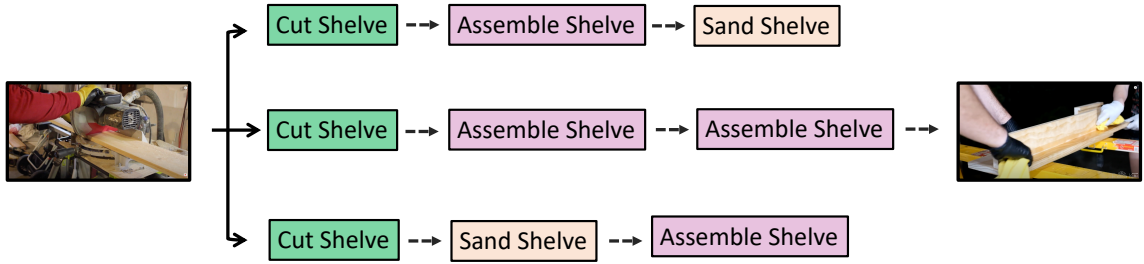


Figure 5.6: Sample *plausible* plans (seen in the test set) for *furniture assembly* produced by our probabilistic model, for the same {start, goal} observations.

5.4.6 Plan visualizations

In this subsection, we provide visualizations of the ability of our model to produce multiple plans given initial and goal video inputs; see Figures 5.4-5.6. In Figure 5.4, our approach generates three plausible plans for the *making beverage* instructional video. The first path contains [*pour powder* $\rightarrow$ *pour water* $\rightarrow$ *pour juice* $\rightarrow$ *stir*]. The other two paths depict possibilities that either uses different ingredients to achieve the same goal, e.g., *pour alcohol* rather than *pour juice*, or operates in different order, e.g., *pour juice* comes before *pour powder*. Similarly, in Figure 5.5 and 5.6, multiple plausible paths are sampled from our approach for the *grill steak* and *furniture assembly* videos, respectively. It is seen that the generated plans are accurate on their own interpretation, but also enjoy variability.

5.5 Conclusion

We have introduced a weakly-supervised approach for probabilistic procedure planning using instructional videos. Different from previous work, we eschew the need for expensive visual supervision in favor of cheaper language supervision by capitalizing on pre-trained text-video embeddings, which, remarkably, leads to superior planning performance. We showed that modeling the interplay between intermediate visual states and actions step-by-step is not a necessity for procedure planning. Instead, we efficiently solve the problem with a “one-shot” transformer decoder architecture. In addition, we demonstrated the crucial role of modeling

uncertainty in obtained plans to yield a principled approach to planning from videos. We introduced a way to evaluate such uncertainty on the test set and show that it is a powerful metric to better understand the model and the task of planning itself. Hopefully, future work will adopt the probabilistic view on procedure planning from instructional videos, not only in training but also in evaluation, such that the next generation of planners can confidently predict multiple feasible plans to achieve the desired goal.

While our approach has shown state-of-the-art performance under the considered evaluation protocols, it would face challenges when being used in real-life systems. First, the model training and testing process presumed the prediction horizon (i.e., $T = 3$), which is infeasible in daily life. There is a need to have a universal estimator that can self-decide the number of prediction steps. Second, to study the future prediction of a certain horizon, we slice long videos into short pieces and assume each of them is an independent video clip, which ignores the long-term history context. There is a need to effectively associate past events to future predictions. Finally, similar to our approach to future activity prediction, explicit use of a normalized distance regularizer to encourage diversity in predictions lends interpretability to our procedure planning approach; however, other learned components the system deserve study of their interpretability.

Chapter 6

Conclusion and future work

6.1 Summary and overall discussion

This dissertation has addressed the topic of video-based prediction with both breadth and depth. Various prediction tasks were introduced along with some representative applications in Chapter 1. Then, we investigated four important sub-areas in detail, namely early action recognition, future activity prediction, trajectory prediction and procedure planning. They are studied in Chapters 2–5, respectively. For each task, we propose a novel solution that greatly improves the performance on multiple public benchmarks and addressed weaknesses of current machine learning techniques, i.e., interpretability, multi-modality learning as well as supervision efficiency.

- For early action recognition in Chapter 2, we focus on studying *what is the best way to propagate deep features* as well as decoding *what has been learned* by the specified deep network. First, to develop an efficient solution that avoids error build-up in long-term predictions we specify a deep network that (1) propagates features into the future via learned motion kernels and (2) borrows ideas from classic state-space functions, i.e., the Kalman filter, which is redesigned with learnable components to allow more expressive power and accommodate high dimensional data (e.g., deep features as large size 3D tensors). Second, after obtaining improved results on the UCF101 [211], JHMDB21 [118] and BIT [110] datasets, we investigate the learnable components by visualizing the learned kernels and the Kalman gain. Our initial exploration suggests

that the learned kernels behave akin to 2D unit impulse kernels, which can perform translational operations (i.e., shifting) on pixels. Moreover, the learned Kalman gain is found to correct poor prior predictions in an adaptive manner, i.e., associating ground truth measurements to the timesteps where non-smooth motion happens.

- For future activity prediction in Chapter 3, *how to generate multiple plausible outcomes* is investigated and enhanced. The well-known trade-off between accuracy and diversity in sequential Gumbel-Softmax based generative adversarial networks is studied. We re-demonstrate the mode collapse issue in discrete GANs with a toy example, systematically illustrate that the temperature parameter in Gumbel-Softmax controls the output distribution shapes and use a distance normalized regularizer to increase the variability as well as keep the accuracy. We demonstrate the superiority of our system on the Breakfast [117], 50Salads [214] and Epic-Kitchens [34] datasets.
- For human trajectory prediction in Chapter 4, a non-parameteric retrieval based framework is proposed to elevate the learning efficiency of the goal position inference. We collect expert goal position examples from the training set, which is augmented by random rotations. In testing, we use a relaxed dynamic time warping technique to compute pairwise similarity between the test input and cached experts. This retrieval operation can run with CUDA support (e.g., real time) and incurs no extra parameters. Further, we associate the retrieved goal by subtracting it from observed motion histories, which further reduces the number of parameters. The overall system is demonstrated to be both effective and efficient on the SDD [189], ETH [174] and UCY [129] datasets.
- For procedure planning in Chapter 5, a language-based weakly supervised training procedure is used to reduce the expansive human efforts on annotating start and end times of instructional videos, which yields state-of-the-art results, even in comparison to strongly supervised alternatives. The weak supervision from language is implemented as a contrastive learning loss. Moreover, we present the first approach that can generate multiple plausible, yet diverse plans at inference via use of a generative adversarial framework. Additionally, we actively explore a better alternative to generate sequential plans in a “one-shot” manner, to avoid built-up errors and speed up the inference step. In this regard, a memory augmented transformer decoder is used to yield the entire plans in parallel. This external memory module incurs the minimum amount of computational effort, as it is shared across multiple transformer layers as well as the entire dataset. We

evaluate our approach with success against both deterministic and probabilistic metrics on the CrossTask [274], NIV [5] and COIN [224] datasets.

6.2 Societal impact

Being able to precisely predict the future information or plan the intermediate steps in videos can bring meaningful (positive) impact for the society.

- Such a predictive vision system can greatly reduce the computational cost and save energy. One of the most urgent needs of video-based vision systems is **energy efficiency**; current approaches might fall short of considerations in this regard. For example, recent action recognition systems rely on the full video observations to produce accurate class labels, which consumes large amounts of resources (e.g., running on GPUs costs electricity as well as releases carbon dioxide), especially when the video volume is large (e.g., minutes long). Existing efforts on alleviating this matter consider either sparse frame sampling [49, 237] or light-weight architectures [248]. Still, these strategies can not handle online streaming videos. The early action recognition framework introduced in Chapter 2 could be a good fit for this purpose: (1) Much less frames are needed as inputs to yield results, (2) the system can run without receiving the full action execution of a video volume.
- Predicting the future can contribute to our daily life by improving the **safety level** in traffic scenes. Road traffic accidents increase dramatically every year and an estimated 1.2 million people are injured in road crashes. The harm mostly happens to vehicles and pedestrians. A vision system capable of predicting the next action/trajectory/intention (e.g., making use of the techniques presented in Chapters 3 and 4) is of great value: (1) Knowing the future information of other agents on the same road allows for prompt (or well-prepared) response, (2) predicting our own future enables early warning for self correction, e.g., when drivers are drowsy-driving or pedestrians are crossing the red light unconsciously.
- The weak supervision introduced in Chapter 5 for procedure planning also brings substantial benefits to our society in terms of **human labor efficiency**. The leading approaches in machine learning are notoriously data-hungry. Academic labs/companies

nowadays are collecting, annotating and publicizing datasets very frequently for better (robust) model learning, which demands huge human annotation efforts. Unfortunately, many application domains or groups of people do not have access to such human power resources. The weak supervision from natural languages is a good example of mitigating such issues. In the future, we hope work in every domain will consider more affordable learning paradigms, e.g., weak or self-supervised learning, to save our energy for other important problems.

However, our research on future prediction might come with potential negative impacts as follows.

- A predictive vision system could create a lack of **privacy**. Arguably, exposing one's plausible future actions to the public can be equal to revealing one's future privacy. This might be harmful, as criminals could take the chance and train machines to predict the behaviors of individuals and profit from this information. For example, a certain pedestrian who is predicted to be heading to the bank by large chance would have money on him/her. If surveillance cameras are hacked, this type of people can be easily targeted by criminals. Also, in social media, being able to forecast one's activity would naturally incur **scams**, e.g., pretending to be a friend of someone if the predictive system suggests a potential interaction is happening in the near future.
- **Wrong predictions** can lead to **life-threatening** events. The performance of current machine learning models are subject to the quality of the data and the associated biases induced during data collection. Therefore, employing predictive vision systems on real life scenarios needs great caution. For instance, for self-driving vehicles, predictive models could potentially lead to dangerous events when bad decisions are made on data it has never seen before, such as, on poor visibility days (i.e., irregular data point), *turning-left* is suggested by the system while the left lane has been occupied already. A key step forward is to improving the prediction accuracy as well as judging on whether or not to trust the predictions. Similarly, as a second example, a wrong prediction on the stock market can cause huge financial losses.
- The selected datasets can cause biases in models trained on them, which leads to **inequity**. More specifically, in the procedure planing work of Chapter 5, most “How To” videos are from the Internet and they are subject to the preference of the video

poster. We observe that most instructional videos on the “furniture assembly” topic use packages bought from IKEA. So, models trained on such data tend to perform better on products from this particular brand, which results in inequity (biases). Perhaps even worse, datasets constructed with biases toward or against certain groups of people could lead to similar biases in predictive systems trained on such data.

6.3 Future work

In this final section, several directions for future work are suggested.

- **Improving prediction through improving action recognition.** Current trending approaches on video predictive understanding greatly rely on success from the video action recognition area, where large scale datasets and powerful video discriminative feature representation are frequently advanced via deep learning. For example in early action recognition, our work uses the Temporal Segment Network (TSN) [236] as a feature extractor and that helps the early recognition baseline from our framework surpass several previous efforts; see results in Table 2.1. Similarly, in procedure planning, we found feature representations of instructional videos has been greatly advanced by large-scale dataset cross modality pre-training, e.g., aligning natural language sentences with vision [163]. This is also true for future activity prediction, where the accuracy of the off-the-shelf action segmentation algorithm, which is needed to pre-process long-term videos into high level abstractions, would decide the quality of future prediction. To the extent that future video activity prediction continues to rely on core action recognition components, future research will need to address improvements to those components, perhaps specialized/fine-tuned to prediction.
- **Adaptive prediction horizons.** Another notable issue in future prediction is that the prediction horizon as well as the length of historical observations are pre-fixed. For example in trajectory prediction, the standard is to predict the next 12 timesteps from eight past trajectory points. Given that trajectories are sampled from videos with a relatively high frequency, e.g., each 0.4 second, the current set-up can reach to trajectories of the next 4.8 seconds at most, which might not be helpful considering humans walk at a fairly low speed. Moreover, the input length is not adaptable, which can harm scenarios

where only very adjacent motion information can be observed due to partial occlusion or narrow camera viewpoint. The current status of procedure planning also has the limitation of fixed prediction horizons. It is assumed that we know how many steps to predict during both training and testing, which is impractical in real-life situations. A breakthrough is needed on making efficient long-term predictions and being capable of working with variable input length. An essential next step is to designing a criteria by which the horizon of prediction may be judged or decided.

- **Better modelling of long-term correlation.** It is necessary to have a better model for capturing long-term correlations to achieve scalable prediction of activities. Take a movie video as an example, the cue that can precisely indicate the future could possibly hide in details that are far away from current observations [241]; thus, it is essential to keep logging historical information and effectively spot the correlations between minor details and future events. Though some efforts have been made for action recognition (e.g., design of a long-term filter bank to keep track of historically remote features [240]), its effects on other areas stays unknown and it is likely beneficial to revisit such an ability for future action prediction.
- **Stronger temporal sequence modelling tools.** Temporal modelling tools are always favorable to video understanding. The majority of mathematical tools used in this dissertation are theoretically grounded and widely adopted. Recently, however, it has been proposed to explore other advances, such as temporal convolution [124], which challenges the use of memory hidden states, or transformers [62], which get rid of the recurrent structure and yet demonstrates superior results, as alternatives to classic sequence modelling tools. These innovate findings break the traditional design where the temporal modelling has to be sequential and mostly Markovian. Such directions are worth further investigation.
- **Learning efficiency.** Contemporary machine learning algorithms mostly are data-driven and consume tremendous amounts of computational resources. AI companies have built up giant GPU clusters and trained models with billions of parameters, which is not affordable to academic labs and developing countries. It is a concern that closely associates with most recent learning approaches. Moreover, most studied approaches prerequisite sufficient size of datasets (e.g., HowTo100M [163]), which incurs huge amounts of human labeling efforts, data cleaning and unavoidable data biases. So, how

to learn meaningful information without strong supervision or with noisy labels awaits to be answered. Besides, there is also a need for adaptive learning frameworks that can assimilate unseen data and new action categories (e.g., open set datasets), since people’s daily activities would evolve along with their own development. An active and continuous learning framework would be of great benefit. Unsupervised approaches to representational learning, e.g., as introduced under the topic of predictive video coding in Chapter 1, provide a potentially useful avenue to explore to address these concerns. As another example, the weak supervision from different modalities (e.g., natural languages discussed in Chapter 5) could be useful.

- **Creation of better datasets and experimental protocols.** Development of improved datasets and experimental protocols is necessary to better validate prediction algorithms. From the perspective of datasets and evaluation metrics, it is worth understanding the specific observation ratios that are practically valuable in real-life conditions with regard to actions, and therefore re-evaluating existing approaches using this new criterion. Moreover, a better dataset that is free of class imbalance issue is much needed. In our last work on procedure planning, we have spotted that the class categories are imbalanced in the used datasets. For example, the presence of *pour water* and *pour milk* is much more prevalent than some others, e.g., *pour alcohol*, which causes a bias. It is also worth investigating the correlation between the video observation ratio and action evolution ratio of popular datasets, in order to examine the validation of current early recognition evaluation metrics. Finally, better dataset collection can mitigate some of the potential negative societal impact discussed in the previous sections, e.g., through curation that ensures equity across interested parties.
- **Further interpretability investigation.** Our work on early action recognition included a detailed interpretability study. Future work should seek to interpret the processing embodied in the other architectures presented in this dissertation. To some extent, these architectures already are interpretable: The probabilistic modeling of future activity prediction and procedure planning use distance regularization to yield predictions that are both realistic and diverse; trajectory prediction uses an expert repository to produce goals from a previously seen training data. Nonetheless, other learned components of the systems require further understanding. The methodology used to investigate early action recognition, i.e., visualization of intermediate representations followed by analytic modeling and empirical validation, should be equally applicable to the other de-

veloped architectures.

Bibliography

- [1] Y. Abu Farha and J. Gall. Uncertainty-aware anticipation of activities. In *Proceedings of the IEEE International Conference on Computer Vision Workshop on Human Behaviour Understanding*, 2019.
- [2] Y. Abu Farha, A. Richard, and J. Gall. When will you do what? -Anticipating temporal occurrences of activities. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
- [3] P. Adam, G. Sam, M. Francisco, L. Adam, B. James, C. Gregory, K. Trevor, L. Zeming, G. Natalia, A. Luca, D. Alban, K. Andreas, Y. Edward, D. Zach, R. Martin, T. Alykhan, C. Sasank, S. Benoit, F. Lu, B. Junjie, and C. Soumith. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019.
- [4] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social LSTM: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [5] J.-B. Alayrac, P. Bojanowski, N. Agrawal, J. Sivic, I. Laptev, and S. Lacoste-Julien. Unsupervised learning from narrated instruction videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [6] D. Alexey, B. Lucas, K. Alexander, W. Dirk, Z. Xiaohua, U. Thomas, D. Mostafa, M. Matthias, H. Georg, G. Sylvain, U. Jakob, and H. Neil. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [7] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2(3):283–310, 1989.

- [8] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *Proceedings of the International Conference on Machine Learning*, 2017.
- [9] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid. ViVit: A video vision transformer. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021.
- [10] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 1(11):11–73, 1997.
- [11] I. Bello, B. Zoph, A. Vaswani, J. Shlens, and Q. V. Le. Attention augmented convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.
- [12] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2015.
- [13] J. Bi, J. Luo, and C. Xu. Procedure planning in instructional videos via contextual modeling and model-based policy learning. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021.
- [14] N. Bisagno, B. Zhang, and N. Conci. Group LSTM: Group trajectory prediction in crowded scenarios. In *Proceedings of the European Conference on Computer Vision Workshop on Anticipating Human Behavior*, 2018.
- [15] S. Z. Bokhari and K. M. Kitani. Long-term activity forecasting using first-person vision. In *Proceedings of the Asian Conference on Computer Vision*, 2016.
- [16] R. N. Bracewell. *The Fourier Transform and its Applications*. McGraw Hill, 1986.
- [17] W. Byeon, Q. Wang, R. K. Srivastava, and P. Koumoutsakos. ContextVP: Fully context-aware video prediction. In *Proceedings of the European Conference on Computer Vision*, 2018.
- [18] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary robust independent elementary features. In *Proceedings of the European Conference on Computer Vision*, 2010.

- [19] K. J. Cannons and R. P. Wildes. The applicability of spatiotemporal oriented energy features to region tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(4):784–796, 2013.
- [20] Y. Cao, D. Barrett, A. Barbu, S. Narayanaswamy, H. Yu, A. Michaux, Y. Lin, S. Dickinson, J. Mark Siskind, and S. Wang. Recognize human activities from partially observed videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.
- [21] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *Proceedings of the European Conference on Computer Vision*, 2020.
- [22] J. Carreira and A. Zisserman. Quo vadis, action recognition? A new model and the Kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [23] A. Chakraborty and A. K. Roy-Chowdhury. Context-aware activity forecasting. In *Proceedings of the Asian Conference on Computer Vision*, 2014.
- [24] C.-Y. Chang, D.-A. Huang, Y. Sui, L. Fei-Fei, and J. C. Niebles. D3TW: Discriminative differentiable dynamic time warping for weakly supervised action alignment and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [25] C.-Y. Chang, D.-A. Huang, D. Xu, E. Adeli, L. Fei-Fei, and J. C. Niebles. Procedure planning in instructional videos. In *Proceedings of the European Conference on Computer Vision*, 2020.
- [26] R. Chaudhry, A. Ravichandran, G. Hager, and R. Vidal. Histograms of oriented optical flow and Binet-Cauchy kernels on nonlinear dynamical systems for the recognition of human actions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
- [27] L. Chen, J. Lu, Z. Song, and J. Zhou. Part-activated deep reinforcement learning for action prediction. In *Proceedings of the European Conference on Computer Vision*, 2018.

- [28] M. Chen, Z. Xu, K. Weinberger, and F. Sha. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the International Conference on Machine Learning*, 2012.
- [29] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.
- [30] H. Coskun, F. Achilles, R. DiPietro, N. Navab, and F. Tombari. Long short-term memory Kalman filters: Recurrent neural estimators for pose regularization. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.
- [31] M. Cuturi and M. Blondel. Soft-DTW: A differentiable loss function for time-series. In *Proceedings of the International Conference on Machine Learning*, 2017.
- [32] B. Dai, S. Fidler, R. Urtasun, and D. Lin. Towards diverse and natural image descriptions via a conditional GAN. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.
- [33] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.
- [34] D. Damen, H. Doughty, G. Maria Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, and W. Price. Scaling egocentric vision: The Epic-Kitchens dataset. In *Proceedings of the European Conference on Computer Vision*, 2018.
- [35] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. In *Proceedings of the International Conference on International Conference on Machine Learning*, 2017.
- [36] P. Dendorfer, S. Elflein, and L. Leal-Taixé. MG-GAN: A multi-generator model preventing out-of-distribution samples in pedestrian trajectory prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021.
- [37] P. Dendorfer, A. Osep, and L. Leal-Taixé. Goal-GAN: Multimodal trajectory prediction based on goal position estimation. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

- [38] K. G. Derpanis, M. Lecce, K. Daniilidis, and R. P. Wildes. Dynamic scene understanding: The role of orientation features in space and time in scene classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012.
- [39] K. Desai and J. Johnson. VirTex: Learning visual representations from textual annotations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.
- [40] Y. Ding, C. Florensa, P. Abbeel, and M. Phielipp. Goal-conditioned imitation learning. In *Advances in Neural Information Processing Systems*, 2019.
- [41] N. Dvornik, I. Hadji, K. Derpanis, A. Garg, and A. Jepson. Drop-DTW: Aligning common signal between sequences while dropping outliers. In *Advances in Neural Information Processing Systems*, 2021.
- [42] K. Ehsani, H. Bagherinezhad, J. Redmon, R. Mottaghi, and A. Farhadi. Who let the dogs out? Modeling dog behavior from visual data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
- [43] F. W. Fairman. *Linear Control Theory: The State Space Approach*. John Wiley & Sons, 1998.
- [44] G. Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian Conference on Image Analysis*, 2003.
- [45] W. Fedus, I. Goodfellow, and A. M. Dai. MaskGAN: Better text generation via filling in the_. *arXiv preprint arXiv:1801.07736*, 2018.
- [46] C. Feichtenhofer, H. Fan, J. Malik, and K. He. Slowfast networks for video recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.
- [47] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Spatiotemporal multi-scale networks for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [48] C. Feichtenhofer, A. Pinz, R. P. Wildes, and A. Zisserman. Deep insights into convolutional networks for video recognition. *International Journal of Computer Vision*, 128(2):420–437, 2020.

- [49] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [50] B. Fernando and S. Herath. Anticipating human actions by correlating past with the future with Jaccard similarity measures. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2021.
- [51] C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in Neural Information Processing Systems*, 2016.
- [52] C. Finn and S. Levine. Deep visual foresight for planning robot motion. In *IEEE International Conference on Robotics and Automation*, 2017.
- [53] C. Florensa, D. Held, X. Geng, and P. Abbeel. Automatic goal generation for reinforcement learning agents. In *Proceedings of the International Conference on Machine Learning*, 2018.
- [54] A. Furnari, S. Battiato, K. Grauman, and G. M. Farinella. Next-active-object prediction from egocentric videos. *Journal of Visual Communication and Image Representation*, 49:401–411, 2017.
- [55] A. Furnari, S. Battiato, and G. Maria Farinella. Leveraging uncertainty to rethink loss functions and evaluation measures for egocentric action anticipation. In *Proceedings of the European Conference on Computer Vision*, 2018.
- [56] A. Furnari and G. M. Farinella. What would you expect? Anticipating egocentric actions with rolling-unrolling LSTM and modality attention. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.
- [57] V. Gabeur, C. Sun, K. Alahari, and C. Schmid. Multi-modal transformer for video retrieval. In *Proceedings of the European Conference on Computer Vision*, 2020.
- [58] H. Gammulle, S. Denman, S. Sridharan, and C. Fookes. Forecasting future action sequences with neural memory networks. *arXiv preprint arXiv:1909.09278*, 2019.
- [59] H. Gammulle, S. Denman, S. Sridharan, and C. Fookes. Predicting the future: A jointly learnt model for action anticipation. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

- [60] J. Gao, Z. Yang, and R. Nevatia. RED: Reinforced encoder-decoder networks for action anticipation. *Proceedings of the European Conference on Computer Vision*, 2017.
- [61] D. Ghosh, A. Gupta, and S. Levine. Learning actionable representations with goal-conditioned policies. *Proceedings of the International Conference on Learning Representations*, 2018.
- [62] R. Girdhar, J. Carreira, C. Doersch, and A. Zisserman. Video action transformer network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [63] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2011.
- [64] J. Goldberger, G. E. Hinton, S. Roweis, and R. R. Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems*, 2004.
- [65] L. Gomez, Y. Patel, M. Rusiñol, D. Karatzas, and C. Jawahar. Self-supervised learning of visual features through embedding images into text topic spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [66] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*. MIT Press, Cambridge, 2016.
- [67] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014.
- [68] J. Gu, D. J. Im, and V. O. Li. Neural machine translation with Gumbel-greedy decoding. In *Proceedings of the AAAI conference on artificial intelligence*, 2018.
- [69] J. Gu, X. Yang, S. De Mello, and J. Kautz. Dynamic facial analysis: From Bayesian filtering to recurrent neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [70] E. J. Gumbel. *Statistical theory of extreme values and some practical applications: A series of lectures*, volume 33. US Government Printing Office, 1948.

- [71] J. Guo, S. Lu, H. Cai, W. Zhang, Y. Yu, and J. Wang. Long text generation via adversarial training with leaked information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.
- [72] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi. Social GAN: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
- [73] A. Gupta, P. Srinivasan, J. Shi, and L. S. Davis. *Understanding Videos, Constructing Plots* Learning a visually grounded storyline model from annotated videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
- [74] U. M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *International Conference on Artificial Intelligence and Statistics*, 2010.
- [75] I. Hadji and R. P. Wildes. A spatiotemporal oriented energy network for dynamic texture recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.
- [76] I. Hadji and R. P. Wildes. A new large scale dynamic texture dataset with application to convnet understanding. In *Proceedings of the European Conference on Computer Vision*, 2018.
- [77] R. Hamid, S. Maddi, A. Johnson, A. Bobick, I. Essa, and C. Isbell. A novel sequence representation for unsupervised analysis of human activities. *Artificial Intelligence*, 173(14):1221–1244, 2009.
- [78] W. L. Hamilton, R. Ying, and J. Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.
- [79] T. Han, W. Xie, and A. Zisserman. Video representation learning by dense predictive coding. In *Proceedings of the IEEE International Conference on Computer Vision Workshop on Human Behaviour Understanding*, 2019.
- [80] C. G. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the Alvey Vision Conference*, 1988.

- [81] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [82] D. Helbing and P. Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.
- [83] J. Herrick. wikiHow. www.wikihow.com, 2005. [Online how-to website].
- [84] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.
- [85] M. Hoai and F. De la Torre. Max-margin early event detectors. *International Journal of Computer Vision*, 107(2):191–202, 2014.
- [86] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- [87] J. Hou, X. Wu, R. Wang, J. Luo, and Y. Jia. Confidence-guided self refinement for action prediction in untrimmed videos. *IEEE Transactions on Image Processing*, 29(4):6017–6031, 2020.
- [88] B. Ivanovic and M. Pavone. The trajetron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.
- [89] A. Jain, H. S. Koppula, B. Raghavan, S. Soh, and A. Saxena. Car that knows before you do: Anticipating maneuvers via learning temporal driving models. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.
- [90] A. Jain, A. Singh, H. S. Koppula, S. Soh, and A. Saxena. Recurrent neural networks for driver activity anticipation via sensory-fusion architecture. In *IEEE International Conference on Robotics and Automation*, 2016.
- [91] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with Gumbel-softmax. In *Proceedings of the International Conference on Learning Representations*, 2016.
- [92] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black. Towards understanding action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, 2013.

- [93] J. Jiao, W.-C. Tu, S. He, and R. W. Lau. FormResNet: Formatted residual learning for image restoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop on New Trends in Image Restoration and Enhancement*, 2017.
- [94] A. Joulin, L. Van Der Maaten, A. Jabri, and N. Vasilache. Learning visual features from large weakly supervised data. In *Proceedings of the European Conference on Computer Vision*, 2016.
- [95] L. P. Kaelbling. Hierarchical learning in stochastic domains: Preliminary results. In *Proceedings of the International Conference on Machine Learning*, 1993.
- [96] L. P. Kaelbling. Learning to achieve goals. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1993.
- [97] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 03 1960.
- [98] S. M. Kang and R. P. Wildes. Review of action recognition and detection methods. *arXiv preprint arXiv:1610.06906*, 2016.
- [99] Q. Ke, M. Fritz, and B. Schiele. Time-conditioned action anticipation in one shot. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [100] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.
- [101] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2014.
- [102] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *Proceedings of the International Conference on Machine Learning*, 2014.
- [103] J. Kingman. *Poisson Processes*. Oxford Science Publications. Clarendon Press, Oxford, 1993.

- [104] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations*, 2017.
- [105] O. Koller, H. Ney, and R. Bowden. Deep hand: How to train a CNN on 1 million hand images when your data is continuous and weakly labelled. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [106] O. Koller, S. Zargaran, and H. Ney. Re-Sign: Re-aligned end-to-end sequence modelling with deep recurrent CNN-HMMs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [107] Y. Kong and Y. Fu. Max-margin action prediction machine. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(9):1844–1858, 2015.
- [108] Y. Kong and Y. Fu. Human action recognition and prediction: A survey. *arXiv preprint arXiv:1806.11230*, 2018.
- [109] Y. Kong and S. Gao. Action prediction from videos via memorizing hard-to-predict samples. In *Proceedings of the AAAI conference on artificial intelligence*, 2018.
- [110] Y. Kong, Y. Jia, and Y. Fu. Interactive phrases: Semantic descriptions for human interaction recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(9):1775–1788, 2014.
- [111] Y. Kong, D. Kit, and Y. Fu. A discriminative model with multiple temporal scales for action prediction. In *Proceedings of the European Conference on Computer Vision*, 2014.
- [112] Y. Kong, Z. Tao, and Y. Fu. Deep sequential context networks for action prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [113] Y. Kong, Z. Tao, and Y. Fu. Adversarial action prediction networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(3):539–553, 2018.
- [114] H. S. Koppula and A. Saxena. Anticipating human activities using object affordances for reactive robotic response. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):14–29, 2015.

- [115] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. Reid, H. Rezatofighi, and S. Savarese. Social-BiGAT: Multimodal trajectory forecasting using bicycle-GAN and graph attention networks. In *Advances in Neural Information Processing Systems*, 2019.
- [116] P. Kothari, S. Kreiss, and A. Alahi. Human trajectory forecasting in crowds: A deep learning perspective. *arXiv preprint arXiv:2007.03639*, 2020.
- [117] H. Kuehne, A. Arslan, and T. Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.
- [118] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, 2011.
- [119] J. Lafferty, A. McCallum, and F. C. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, 2001.
- [120] S. Lai, L. Xu, K. Liu, and J. Zhao. Recurrent convolutional neural networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, 2015.
- [121] S. Lai, W.-S. Zheng, J.-F. Hu, and J. Zhang. Global-local temporal saliency action prediction. *IEEE Transactions on Image Processing*, 27(5):2272–2285, 2017.
- [122] T. Lan, T.-C. Chen, and S. Savarese. A hierarchical representation for future action prediction. In *Proceedings of the European Conference on Computer Vision*, 2014.
- [123] I. Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2-3):107–123, 2005.
- [124] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager. Temporal convolutional networks for action segmentation and detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [125] A. X. Lee, R. Zhang, F. Ebert, P. Abbeel, C. Finn, and S. Levine. Stochastic adversarial video prediction. *arXiv preprint arXiv:1804.01523*, 2018.
- [126] K. Lee, K. Guu, L. He, T. Dozat, and H. W. Chung. Neural data augmentation via example extrapolation. *arXiv preprint arXiv:2102.01335*, 2021.

- [127] J. Lei, L. Wang, Y. Shen, D. Yu, T. Berg, and M. Bansal. MART: Memory-augmented recurrent transformer for coherent video paragraph captioning. In *Proceedings of the Association for Computational Linguistics*, 2020.
- [128] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPNP: An accurate $O(n)$ solution to the PNP problem. *International Journal of Computer Vision*, 81(2):155, 2009.
- [129] A. Lerner, Y. Chrysanthou, and D. Lischinski. Crowds by example. In *Computer Graphics Forum*, volume 26, 2007.
- [130] J. Li, H. Ma, and M. Tomizuka. Conditional generative neural system for probabilistic trajectory prediction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019.
- [131] J. Li, F. Yang, M. Tomizuka, and C. Choi. EvolveGraph: Multi-agent trajectory prediction with dynamic relational reasoning. *Advances in Neural Information Processing Systems*, 2020.
- [132] K. Li and Y. Fu. Prediction of human activity by discovering temporal sequence patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1644–1657, 2014.
- [133] K. Li, J. Hu, and Y. Fu. Modeling complex temporal composition of actionlets for activity prediction. In *Proceedings of the European Conference on Computer Vision*, 2012.
- [134] M. Li, Y.-X. Wang, and D. Ramanan. Towards streaming perception. In *Proceedings of the European Conference on Computer Vision*, 2020.
- [135] Y. Li, N. Du, and S. Bengio. Time-dependent representation for neural event sequence prediction. *arXiv preprint arXiv:1708.00065*, 2017.
- [136] Y. Li, P. Wang, and C.-Y. Chan. Restep into the future: relational spatio-temporal learning for multi-person action forecasting. *IEEE Transactions on Multimedia*, 2021.
- [137] Y. Li, P. Wang, L. Chen, Z. Wang, and C.-Y. Chan. ELMA: Energy-based learning for multi-agent activity forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, 2022.

- [138] Y.-K. Li, P. Wang, M. Ye, and C.-Y. Chan. Imitative learning for multi-person action forecasting. In *Proceedings of the ACM International Conference on Multimedia*, 2021.
- [139] J. Liang, L. Jiang, and A. Hauptmann. SimAug: Learning robust representations from 3D simulation for pedestrian trajectory prediction in unseen cameras. In *Proceedings of the European Conference on Computer Vision*, 2020.
- [140] J. Liang, L. Jiang, J. C. Niebles, A. G. Hauptmann, and L. Fei-Fei. Peeking into the future: Predicting future person activities and locations in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [141] J. Liu, A. Shahroudy, G. Wang, L.-Y. Duan, and A. C. Kot. SSNet: Scale selection network for online 3D action prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
- [142] M. Liu, S. Tang, Y. Li, and J. Rehg. Forecasting human object interaction: Joint prediction of motor attention and egocentric activity. *Proceedings of the European Conference on Computer Vision*, 2019.
- [143] S. Liu, X. Zhang, J. Wangni, and J. Shi. Normalized diversification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [144] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision*, 2016.
- [145] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the IEEE International Conference on Computer Vision*, 1999.
- [146] G. Lu, W. Ouyang, D. Xu, X. Zhang, Z. Gao, and M.-T. Sun. Deep Kalman filtering network for video compression artifact reduction. In *Proceedings of the European Conference on Computer Vision*, 2018.
- [147] J. Lu, D. Batra, D. Parikh, and S. Lee. ViLBERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems*, 2019.
- [148] P. Luc, N. Neverova, C. Couprie, J. Verbeek, and Y. LeCun. Predicting deeper into the future of semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

- [149] H. Luo, L. Ji, B. Shi, H. Huang, N. Duan, T. Li, J. Li, T. Bharti, and M. Zhou. UniVL: A unified video and language pre-training model for multimodal understanding and generation. *arXiv preprint arXiv:2002.06353*, 2020.
- [150] S. Ma, L. Sigal, and S. Sclaroff. Learning activity progression in LSTMs for activity detection and early detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [151] J. K. MacKie-Mason, A. V. Osepayshvili, D. M. Reeves, and M. P. Wellman. Price prediction strategies for market-based scheduling. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 2004.
- [152] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of the International Conference on Learning Representations*, 2017.
- [153] M. Maghoumi, E. M. Taranta II, and J. J. LaViola Jr. DeepNAG: Deep non-adversarial gesture generation. *arXiv preprint arXiv:2011.09149*, 2020.
- [154] T. Mahmud, M. Billah, M. Hasan, and A. K. Roy-Chowdhury. Captioning near-future activity sequences. *arXiv preprint arXiv:1908.00943*, 2019.
- [155] T. Mahmud, M. Hasan, A. Chakraborty, and A. K. Roy-Chowdhury. A Poisson process model for activity forecasting. In *Proceedings of the IEEE International Conference in Image Processing*, 2016.
- [156] T. Mahmud, M. Hasan, and A. K. Roy-Chowdhury. Joint prediction of activity labels and starting times in untrimmed videos. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.
- [157] M. Maimone, Y. Cheng, and L. Matthies. Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics*, 24(3):169–186, 2007.
- [158] K. Mangalam, H. Girase, S. Agarwal, K.-H. Lee, E. Adeli, J. Malik, and A. Gaidon. It is not the journey but the destination: Endpoint conditioned trajectory prediction. In *Proceedings of the European Conference on Computer Vision*, 2020.
- [159] F. Marchetti, F. Becattini, L. Seidenari, and A. D. Bimbo. MANTRA: Memory augmented networks for muliee transactions on image processingingle trajectory prediction.

In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

- [160] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.
- [161] N. Mehra, A. A. Jyothi, T. Durand, J. He, L. Sigal, and G. Mori. A variational auto-encoder model for stochastic point processes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [162] A. Miech, J.-B. Alayrac, L. Smaira, I. Laptev, J. Sivic, and A. Zisserman. End-to-end learning of visual representations from uncurated instructional videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.
- [163] A. Miech, D. Zhukov, J.-B. Alayrac, M. Tapaswi, I. Laptev, and J. Sivic. HowTo100M: Learning a text-video embedding by watching hundred million narrated video clips. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [164] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.
- [165] A. Mohamed, K. Qian, M. Elhoseiny, and C. Claudel. Social-STGCNN: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.
- [166] M. Muja and D. G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2227–2240, 2014.
- [167] A. V. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, 2018.
- [168] L. Neumann, A. Zisserman, and A. Vedaldi. Future event prediction: If and when. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop on Precognition: Seeing through the Future*, 2019.

- [169] W. Nie, N. Narodytska, and A. Patel. RelGAN: Relational generative adversarial networks for text generation. In *Proceedings of the International Conference on Learning Representations*, 2019.
- [170] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.
- [171] A. Pashevich, C. Schmid, and C. Sun. Episodic transformer for vision-and-language navigation. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021.
- [172] D. Pathak, P. Mahmoudieh, G. Luo, P. Agrawal, D. Chen, Y. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell. Zero-shot visual imitation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
- [173] M. Pei, Y. Jia, and S.-C. Zhu. Parsing video events with goal inference and intent prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 487–494, 2011.
- [174] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool. You’ll never walk alone: Modeling social behavior for multi-target tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, 2009.
- [175] A. Piergiovanni, A. Angelova, A. Toshev, and M. S. Ryoo. Adversarial generative grammars for human activity prediction. In *Proceedings of the European Conference on Computer Vision*, 2020.
- [176] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, K. Gretchen, and S. Ilya. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.
- [177] A. Ranjan and M. J. Black. Optical flow estimation using a spatial pyramid network. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.
- [178] M. Raptis and L. Sigal. Poselet key-framing: A model for human activity recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2650–2657, 2013.

- [179] A. Rasouli, I. Kotseruba, T. Kunic, and J. K. Tsotsos. PIE: A large-scale dataset and models for pedestrian intention estimation and trajectory prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.
- [180] A. Rasouli, I. Kotseruba, and J. K. Tsotsos. Are they going to cross? A benchmark dataset and baseline for pedestrian crosswalk behavior. In *Proceedings of the IEEE International Conference on Computer Vision Workshop on Computer Vision for Road Scene Understanding and Autonomous Driving*, 2017.
- [181] A. Rasouli, I. Kotseruba, and J. K. Tsotsos. Pedestrian action anticipation using contextual feature fusion in stacked RNNs. *arXiv preprint arXiv:2005.06582*, 2020.
- [182] A. Razavi, A. Van den Oord, and O. Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. In *Advances in Neural Information Processing Systems*, 2019.
- [183] F. A. Reda, G. Liu, K. J. Shih, R. Kirby, J. Barker, D. Tarjan, A. Tao, and B. Catanzaro. SDC-Net: Video prediction using spatially-displaced convolution. In *Proceedings of the European Conference on Computer Vision*, 2018.
- [184] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [185] N. Rhinehart and K. M. Kitani. First-person activity forecasting with online inverse reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.
- [186] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine. PRECOG: Prediction conditioned on goals in visual multi-agent settings. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.
- [187] A. Richard, H. Kuehne, and J. Gall. Weakly supervised action learning with RNN based fine-to-coarse modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [188] A. Richard, H. Kuehne, A. Iqbal, and J. Gall. NeuralNetwork-Viterbi: A framework for weakly supervised video learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

- [189] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *Proceedings of the European Conference on Computer Vision*, 2016.
- [190] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras. Human motion trajectory prediction: A survey. *The International Journal of Robotics Research*, 39(8):895–935, 2020.
- [191] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.
- [192] M. S. Ryoo. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *Proceedings of the IEEE International Conference on Computer Vision*, 2011.
- [193] S. Ryoo, M., T. J. Fuchs, L. Xia, J. K. Aggarwal, and L. Matthies. Robot-centric activity prediction from first-person videos: What will they do to me? In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction*, 2015.
- [194] M. Sadegh Aliakbarian, F. Sadat Saleh, M. Salzmann, B. Fernando, L. Petersson, and L. Andersson. Encouraging LSTM to anticipate actions very early. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.
- [195] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, H. Rezatofighi, and S. Savarese. Sophie: An attentive GAN for predicting paths compliant to social and physical constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [196] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken processing recognition. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, 1978.
- [197] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.
- [198] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone. Trajectron++: Multi-agent generative trajectory forecasting with heterogeneous data for control. *Proceedings of the European Conference on Computer Vision*, 2020.

- [199] A. Santoro, R. Faulkner, D. Raposo, J. Rae, M. Chrzanowski, T. Weber, D. Wierstra, O. Vinyals, R. Pascanu, and T. Lillicrap. Relational recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2018.
- [200] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. In *Advances in Neural Information Processing Systems*, 2017.
- [201] B. Schölkopf, K. Tsuda, and J.-P. Vert. *Kernel Methods in Computational Biology*. MIT press, 2004.
- [202] C. Schöller, V. Aravantinos, F. Lay, and A. Knoll. What constant velocity model can teach us about pedestrian motion prediction. *IEEE Robotics and Automation Letters*, 5(2):1696–1703, 2020.
- [203] O. Shchur, M. Biloš, and S. Günnemann. Intensity-free learning of temporal point processes. *arXiv preprint arXiv:1909.12127*, 2019.
- [204] Y. Shen, B. Ni, Z. Li, and N. Zhuang. Egocentric activity prediction via event modulated attention. In *Proceedings of the European Conference on Computer Vision*, 2018.
- [205] Y. Shi, B. Fernando, and R. Hartley. Action anticipation with RBF kernelized feature mapping RNN. In *Proceedings of the European Conference on Computer Vision*, 2018.
- [206] G. Singh, S. Saha, M. Sapienza, P. H. Torr, and F. Cuzzolin. Online real-time mulieeee transactions on image processingle spatiotemporal action localisation and prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.
- [207] J. Sivic and A. Zisserman. Efficient visual search of videos cast as text retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):591–606, 2008.
- [208] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 2017.
- [209] K. Soomro, H. Idrees, and M. Shah. Predicting the where and what of actors and actions through online action localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

- [210] K. Soomro, H. Idrees, and M. Shah. Online localization and prediction of actions and interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):459–472, 2018.
- [211] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.
- [212] A. Srinivas, A. Jabri, P. Abbeel, S. Levine, and C. Finn. Universal planning networks: Learning generalizable representations for visuomotor control. In *Proceedings of the International Conference on Machine Learning*, 2018.
- [213] A. Srivastava, L. Valkov, C. Russell, M. U. Gutmann, and C. Sutton. VEEGAN: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems*, 2017.
- [214] S. Stein and S. J. McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2013.
- [215] S. Su, C. Peng, J. Shi, and C. Choi. Potential field: Interpretable and unified representation for trajectory prediction. *arXiv preprint arXiv:1911.07414*, 2019.
- [216] S. Su, J. Pyo Hong, J. Shi, and H. Soo Park. Predicting behaviors of basketball players from first person videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [217] C. Sun, A. Shrivastava, C. Vondrick, R. Sukthankar, K. Murphy, and C. Schmid. Relational action forecasting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [218] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.
- [219] J. Sun, D.-A. Huang, B. Lu, Y.-H. Liu, B. Zhou, and A. Garg. PlaTe: Visually-grounded planning with transformers in procedural tasks. *arXiv preprint arXiv:2109.04869v1*, 2021.

- [220] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, 2014.
- [221] T. Suzuki, H. Kataoka, Y. Aoki, and Y. Satoh. Anticipating traffic accidents with adaptive loss and large-scale incident DB. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
- [222] C. Tang and R. R. Salakhutdinov. Multiple futures prediction. In *Advances in Neural Information Processing Systems*, 2019.
- [223] L. Tang. Methods for encrypting and decrypting MPEG video data efficiently. In *Proceedings of the fourth ACM International Conference on Multimedia*, 1997.
- [224] Y. Tang, D. Ding, Y. Rao, Y. Zheng, D. Zhang, L. Zhao, J. Lu, and J. Zhou. COIN: A large-scale dataset for comprehensive instructional video analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [225] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.
- [226] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of machine learning research*, 6(9):1453–1484, 2005.
- [227] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.
- [228] K. Veselý, A. Ghoshal, L. Burget, and D. Povey. Sequence-discriminative training of deep neural networks. In *Proceedings of the International Speech Communication Association*.
- [229] R. Villegas, J. Yang, S. Hong, X. Lin, and H. Lee. Decomposing motion and content for natural video sequence prediction. *Proceedings of the International Conference on Learning Representations*, 2017.
- [230] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, 2016.

- [231] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.
- [232] C. Vondrick, H. Pirsiavash, and A. Torralba. Anticipating the future by watching unlabeled video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, 2016.
- [233] C. Vondrick, H. Pirsiavash, and A. Torralba. Anticipating visual representations from unlabeled video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [234] D. Wang, Y. Yuan, and Q. Wang. Early action prediction with generative adversarial networks. *IEEE Access*, pages 35795–35804, 2019.
- [235] H. Wang and C. Schmid. Action recognition with improved trajectories. In *Proceedings of the IEEE International Conference on Computer Vision*, 2013.
- [236] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards good practices for very deep two-stream ConvNets. In *Proceedings of the European Conference on Computer Vision*, 2016.
- [237] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. V. Gool. Temporal segment networks: Towards good practices for deep action recognition. In *Proceedings of the European Conference on Computer Vision*, 2016.
- [238] X. Wang, J.-F. Hu, J.-H. Lai, J. Zhang, and W.-S. Zheng. Progressive teacher-student learning for early action prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [239] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.
- [240] C.-Y. Wu, C. Feichtenhofer, H. Fan, K. He, P. Krahenbuhl, and R. Girshick. Long-term feature banks for detailed video understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [241] C.-Y. Wu and P. Krahenbuhl. Towards long-form video understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

- [242] C.-Y. Wu, M. Zaheer, H. Hu, R. Manmatha, A. J. Smola, and P. Krähenbühl. Compressed video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
- [243] H. Wu, J. Zhang, Z. Gao, J. Wang, P. S. Yu, M. Long, et al. PredRNN: A recurrent neural network for spatiotemporal predictive learning. *arXiv preprint arXiv:2103.09504*, 2021.
- [244] Q. Wu, Z. Lan, J. Gu, and Z. Yu. Memformer: The memory-augmented transformer. *arXiv preprint arXiv:2010.06891*, 2020.
- [245] X. Wu, R. Wang, J. Hou, H. Lin, and J. Luo. Spatial–temporal relation reasoning for action prediction in videos. *International Journal of Computer Vision*, 129(5):1484–1505, 2021.
- [246] S. Xiao, M. Farajtabar, X. Ye, J. Yan, L. Song, and H. Zha. Wasserstein learning of deep generative point process models. In *Advances in Neural Information Processing Systems*, 2017.
- [247] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le. Self-training with noisy student improves ImageNet classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.
- [248] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision*, 2018.
- [249] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, 2015.
- [250] H. Xu, G. Ghosh, P.-Y. Huang, D. Okhonko, A. Aghajanyan, F. Metze, L. Zettlemoyer, and C. Feichtenhofer. VideoCLIP: Contrastive pre-training for zero-shot video-text understanding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2021.
- [251] J. Xu, H. Xu, B. Ni, X. Yang, and T. Darrell. Video prediction via example guidance. In *Proceedings of the International Conference on Machine Learning*, 2020.

- [252] W. Xu, J. Yu, Z. Miao, L. Wan, and Q. Ji. Prediction-CGAN: Human action prediction with conditional generative adversarial networks. In *Proceedings of the ACM International Conference on Multimedia*, 2019.
- [253] C. Yang, K. Yuan, Q. Zhu, W. Yu, and Z. Li. Multi-expert learning of adaptive legged locomotion. *Science Robotics*, 5(49):DOI:10.1126/scirobotics.abb2174, 2020.
- [254] D. Yang, S. Hong, Y. Jang, T. Zhao, and H. Lee. Diversity-sensitive conditional generative adversarial networks. In *Proceedings of the International Conference on Learning Representations*, 2019.
- [255] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4):13–45, 2006.
- [256] C. Yu, X. Ma, J. Ren, H. Zhao, and S. Yi. Spatio-temporal graph transformer networks for pedestrian trajectory prediction. In *Proceedings of the European Conference on Computer Vision*, 2020.
- [257] L. Yu, W. Zhang, J. Wang, and Y. Yu. SeqGAN: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.
- [258] Y. Yuan and K. Kitani. DLow: Diversifying latent flows for diverse human motion prediction. In *Proceedings of the European Conference on Computer Vision*, 2020.
- [259] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Proceedings of the European Conference on Computer Vision*, 2014.
- [260] K.-H. Zeng, S.-H. Chou, F.-H. Chan, J. Carlos Niebles, and M. Sun. Agent-centric risk assessment: Accident anticipation and risky region localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [261] K.-H. Zeng, W. B. Shen, D.-A. Huang, M. Sun, and J. Carlos Niebles. Visual forecasting by imitating dynamics in natural sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.
- [262] C. Zhang, A. Gupta, and A. Zisserman. Temporal query networks for fine-grained video understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

- [263] M. Zhang, K. Teck Ma, J. Hwee Lim, Q. Zhao, and J. Feng. Deep future gaze: Gaze anticipation on egocentric videos using adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [264] P. Zhang, W. Ouyang, P. Zhang, J. Xue, and N. Zheng. SR-LSTM: State refinement for LSTM towards pedestrian trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [265] H. Zhao, I. Hadji, N. Dvornik, K. G. Derpanis, R. P. Wildes, and A. D. Jepson. P3iv: Probabilistic procedure planning from instructional videos with weak supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022.
- [266] H. Zhao and R. P. Wildes. Spatiotemporal feature residual propagation for action prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.
- [267] H. Zhao and R. P. Wildes. On diverse asynchronous activity anticipation. In *Proceedings of the European Conference on Computer Vision*, 2020.
- [268] H. Zhao and R. P. Wildes. Where are you heading? dynamic trajectory prediction with expert goal examples. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021.
- [269] J. Zhao and L. Itti. ShapeDTW: Shape dynamic time warping. *Pattern Recognition*, 74:171–184, 2018.
- [270] H. Zhong and R. P. Wildes. Egomotion estimation using binocular spatiotemporal oriented energy. In *Proceedings of the European Conference on Computer Vision*, 2013.
- [271] B. Zhou, A. Andonian, A. Oliva, and A. Torralba. Temporal relational reasoning in videos. In *Proceedings of the European Conference on Computer Vision*, 2018.
- [272] X. Zhu, X. Jia, and K.-Y. K. Wong. Pixel-level hand detection with shape-aware structured forests. In *Proceedings of the Asian Conference on Computer Vision*, 2014.
- [273] X. Zhu, A. Toisoul, J.-M. Prez-Ra, L. Zhang, B. Martinez, and T. Xiang. Few-shot action recognition with prototype-centered attentive learning. *arXiv preprint arXiv:2101.08085*, 2021.

- [274] D. Zhukov, J.-B. Alayrac, R. G. Cinbis, D. Fouhey, I. Laptev, and J. Sivic. Cross-task weakly supervised learning from instructional videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.