Deep Sequence Learning for Video Anticipation:
From Discrete and Deterministic to Continuous and Stochastic

Mohammad Sadegh Aliakbarian

A thesis submitted for the degree of
Doctor of Philosophy
The Australian National University

October 2020
Draft Copy – 12 October 2020
I hereby declare that this thesis is my original work which has been done in collaboration with other researchers. This document has not been submitted to obtain a degree or award in any other university or educational institution. Parts of this thesis have been published in collaboration with other researchers in international conferences and journals as listed in the Publications section.

Mohammad Sadegh Aliakbarian
12 October 2020
To my loving wife and my beloved parents
There are a number of people, without whom this thesis might not have been written.

Firstly, I would like to express my deepest gratitude to my amazing supervisors, Dr. Lars Petersson, Dr. Mathieu Salzmann, Dr. Basura Fernando, and Prof. Steven Gould for their generous support, motivation, and patience throughout my study. I would like to thank Dr. Lars Petersson for his generous and continuous support. He has always been friendly and professional, so that the research environment was always enjoyable and fruitful for the entire team. I remember when I was working on my first paper, he puts a lot of energy and time to make sure that I do not lose my hopes at the beginning of this journey. Near the deadline, he stayed in office until very late at night to help me finish the experiments. Beyond research, he also cares for the well-being of his students. I never forget the fun moments we had in our all year long Friday teas. I also would like to thank Dr. Mathieu Salzmann. Although he was not in Australia, he kindly continued supervising me and attended every meetings we had, even very early in the morning back in Switzerland. I learned a lot about how to approach a problem, how to criticize my solutions to make it perfect, and how to design experiments so that the message is clearly and efficiently conveyed. As a non-English speaker, I also learned a lot how to write clearly and precisely. I am also very grateful to Dr. Basura Fernando, whom I had the chance to work with during my PhD. While being one of the pioneers in the field, he is so humble. I would like to thank him for his support specially in the practical aspects of my research. I want to thank Dr. Stephen Gould who gave detailed insightful comments on my recent research projects during my PhD. His comments, guidance, and ideas helped me a lot improving my works. He also toughed me how to prepare better research presentations, which is going to be a skill that I always need. I also would like to thank him and the ACRV for the generous support for the travels.

I would also show my gratitude to my amazing colleagues during my internship at Qualcomm. Thanks to Amir and Babak who made that four-months internship a memorable one. I want to thank my amazing and professional colleagues at FiveAI when I was an intern there. Thanks to my mentor, Nick, and other colleagues, Stuart, Luca, Tommaso, Oscar, and Steinar. Also, thanks to Prof. Philip Torr and Dr. Puneet Dokania for their insightful discussions on my internship project.

I am also grateful to NICTA/Data61 and the Australian National University for providing my Ph.D. scholarship and enriching my academic experience by providing conference travel funding. I thank Ali C., Ali A., Mehrdad, and Dylan, as well as all other past and the present members of the Data61’s Smart Vision System’s Group especially Lars Andersson and Lachlan Tychsen-Smith for helping me dealing with technical aspects of the work.

I am also very happy that I found such an amazing friends here in Australia. I want to thank Sarah and Mohammad E., my first and two of the best friends, who did not know me when I just arrived to Australia, but helped me unconditionally to settle in. Throughout these years,
they always help me, like siblings, and made sure everything is always alright. To Mohammad N., a great officemate and an amazing friend, who I had really great time with. A special thanks to Alireza and Masoumeh, and adorable little Rose who are just our family and made living here much more enjoyable. To Alireza K., Mehdi, Ehsan, Mousa, Salim, Farshad, Hajar, who is no longer with us but remains in our mind and heart, and my all other friends, I won’t forget your support and unforgettable weekend parties, game nights, and short trips, which we could relax and forget the stress and the deadlines of academia.

I would like to express my gratitude to my parents-in-law for their unfailing emotional support. I also would like to thank my sisters-in-law, specially Fahimeh, with whom I feel we have a bigger family here in Australia.

A very special word of thanks goes for my inspiring and supportive parents. Your love, support and encouragement cannot be expressed by words. This accomplishment would not have been possible without your support. Thanks to my dear sisters Fatemeh and Zahra for their emotional support, endless love and care.

Finally, I would like to thank my special person, my loving wife, who has always been my best friend and my amazing colleague. Fatemeh, thank you so much for your unconditional love, you continued and unfailing support, and your patience and understanding during my study. Words cannot express how I’m grateful for your contributions and suggestions to my research. I am grateful that you never stopped supporting me and believing in me. Every time I fail, you are the one who takes my hand and help me stand again. What can be better than having such a lovely and caring person?
Abstract

Video anticipation is the task of predicting one/multiple future representation(s) given limited, partial observation. This is a challenging task due to the fact that given limited observation, the future representation can be highly ambiguous. Based on the nature of the task, video anticipation can be considered from two viewpoints: the level of details and the level of determinism in the predicted future. In this research, we start from anticipating a coarse representation of a deterministic future and then move towards predicting continuous and fine-grained future representations of a stochastic process. The example of the former is video action anticipation in which we are interested in predicting one action label given a partially observed video and the example of the latter is forecasting multiple diverse continuations of human motion given partially observed one.

In particular, in this thesis, we make several contributions to the literature of video anticipation. The first two contributions mainly focus on anticipating a coarse representation of a deterministic future while the third and fourth contributions focus on predicting continuous and fine-grained future representations of a stochastic process. Firstly, we introduce a general action anticipation framework in which, given very limited observation, the goal is to predict the action label as early as possible. This task is highly critical in scenarios where one needs to react before the action is finalized. This is, for instance, the case of automated driving, where a car needs to, e.g., avoid hitting pedestrians and respect traffic lights. Our work builds on the following observation: a good anticipation model requires (i) a good video representation that is discriminative enough even in presence of partial observation; and (ii) a learning paradigm that not only encourages correct predictions as early as possible, but also accounts for the fact that the future is highly ambiguous. On publicly available action recognition datasets, our proposed method is able to predict markedly accurate action categories given very limited observation, e.g., less than 2% of the videos of UCF-101, outperforming the state of the art methods by a large margin. Secondly, we proposed an action anticipation in driving scenarios. Since there was no anticipation-specific dataset covering generic driving scenarios, as part of our second contribution, we introduced a large-scale video anticipation dataset, covering 5 generic driving scenarios, with a total of 25 distinct action classes. It contains videos acquired in various driving conditions, weathers, daytimes and environments, complemented with a common and realistic set of sensor measurements. This dataset is now publicly available. We then focus on the continuous future prediction problem on tasks that are stochastic in nature; given one observation, multiple plausible futures are likely. In particular, we target the problem of human motion prediction, i.e., the task of predicting future3D human poses given a sequence of observed ones. To this end, in our third contribution, we propose a novel diverse human motion prediction framework based on variational autoencoders (VAEs). In this approach, we particularly propose a novel stochastic conditioning scheme that is well-suited for scenarios where we are dealing with a deterministic datasets, with strong conditioning signals, and expressive

1https://sites.google.com/view/viena2-project/
decoders. Through extensive experiments, we show that our approach performs much better than existing approaches and standard practices in training a conditional VAE. Finally, in the fourth contribution, we propose a conditional VAE framework that solves two main issues of a standard conditional VAE: (i) conditioning and sampling the latent variables are two independent processes, and (ii) the prior distribution is set to be unconditional in practice, however, it should be conditioned on the conditioning signal as elaborated in the evidence lower bound of the data likelihood. In our proposed approach, we address both of these issues that leads to substantial improvement in the quality of generated samples.

All the methods introduced in this thesis are evaluated on standard benchmark datasets. The experiments at the end of each chapter provide compelling evidence that all of our approaches are more efficient than the contemporary baselines.
Contents

Acknowledgments vii

Abstract ix

Publications xxv

1 Introduction 1
  1.1 Problem Definition 1
     Discrete and Deterministic Anticipation. 1
     Continuous and Stochastic Anticipation. 2
  1.2 Thesis Outline and Contributions 2
     1.2.1 General Action Anticipation 2
     1.2.2 Action Anticipation in Driving Scenarios 3
     1.2.3 A Stochastic Conditioning Scheme for Diverse Sequence Generation 3
     1.2.4 Variational Autoencoders with Learned Conditional Priors 4

2 Technical Background 5
  2.1 Convolutional Neural Networks 5
     VGG16. 6
     DenseNet. 7
  2.2 Recurrent Neural Networks 8
     2.2.1 Vanilla RNN. 8
     LSTM. 9
     GRU. 9
  2.3 Variational Autoencoders 10

3 General Action Anticipation 15
  3.1 Introduction 15
  3.2 Related Work 17
     3.2.1 Action Anticipation . 17
     3.2.2 Action Modeling 18
  3.3 Our Approach 19
     3.3.1 A New Loss for Action Anticipation 19
     3.3.2 Multi-stage LSTM Architecture 20
        3.3.2.1 Context- and Action-aware Modeling 20
        Context-Aware Feature Extraction. 20
        Action-Aware Feature Extraction. 21
     3.3.2.2 Sequence Learning for Action Anticipation 21

Draft Copy – 12 October 2020
| Section | Page |
|---------|------|
| Learning Context | 22 |
| Learning Context and Action | 22 |
| Inference | 23 |
| 3.4 Experiments | 23 |
| 3.4.1 Datasets | 23 |
| 3.4.2 Implementation Details | 24 |
| | CNN and LSTM Configuration. | 24 |
| | Training Procedure. | 24 |
| 3.4.3 Comparison to the State-of-the-Art | 24 |
| | JHMDB-21. | 24 |
| | UT-Interaction. | 25 |
| | UCF-101. | 25 |
| 3.4.4 Analysis | 26 |
| 3.4.4.1 Influence of the Loss Function | 26 |
| 3.4.4.2 Influence of the Features | 28 |
| 3.4.4.3 Visualization | 29 |
| 3.5 Comparison to State-of-the-Art Action Recognition Methods | 30 |
| 3.6 Exploiting Optical Flow | 31 |
| 3.7 Effect of Different Feature Types | 32 |
| 3.8 Robustness to the Number of Hidden Units | 33 |
| 3.9 Conclusion | 35 |
| 4 Action Anticipation in Driving Scenarios | 37 |
| 4.1 Introduction | 37 |
| 4.2 VIENA^2 | 39 |
| 4.2.1 Scenarios and Data Collection | 39 |
| 4.2.2 Comparison to Other Datasets | 41 |
| 4.3 Benchmark Algorithms | 42 |
| 4.3.1 Baseline Methods | 42 |
| | Baseline 1: CNN+LSTMs. | 43 |
| | Baseline 2: Two-Stream Networks. | 43 |
| | Baseline 3: Multi-Stage LSTMs. | 43 |
| 4.3.2 A New Multi-Modal LSTM Learning. | 44 |
| 4.3.3 Action Modeling | 45 |
| 4.3.4 Implementation Details | 46 |
| 4.4 Benchmark Evaluation and Analysis | 46 |
| 4.4.1 Action Anticipation on VIENA^2 | 47 |
| 4.4.2 Challenges of VIENA^2 | 47 |
| 4.4.3 Benefits of VIENA^2 for Anticipation from Real Images | 49 |
| 4.4.4 Bias Analysis | 49 |
| 4.5 Benchmark Evaluation on All Splits of VIENA^2 | 51 |
| 4.6 Ablation Study for our MM-LSTM | 51 |
| | Effect of Loss Function | 51 |
## Contents

- Effect of Different Features ........................................ 51
- Effect of the Number of Hidden Units ............................ 53
- Effect of Input Block .................................................. 53
- Comparison to the State of the Art ................................ 53

4.7 VIENA² Visualization: Environmental Diversity .................. 55
4.8 VIENA² Visualization: Scenarios .................................. 56
4.9 Conclusion ............................................................... 56

5 A Stochastic Conditioning Scheme for Diverse Human Motion Prediction 59
5.1 Introduction .............................................................. 59
5.2 Related Work ............................................................ 60
5.3 Proposed Method ....................................................... 62
  5.3.1 Mix-and-Match Perturbation .................................... 62
  5.3.2 M&M Perturbation for Motion Prediction .................... 63
  5.3.3 Curriculum Learning of Variation ............................... 65
  5.3.4 Quality and Diversity Metrics ................................... 66
5.4 Experiments .............................................................. 67
  Baselines ................................................................. 67
  Implementation Details .................................................. 68
  Evaluation Metrics ....................................................... 68
  5.4.1 Comparison to the State-of-the-Art ............................ 70
  5.4.2 Analysis on Diversity and Quality ............................. 70
    Diversity in Hidden State Space. ................................. 71
    Diversity and Quality in Motion Space ......................... 73
    Evaluating the Sampling Error .................................... 74
5.5 Conclusion ............................................................... 74

6 Better Motion Generation via Variational Autoencoders with Learned Conditional Priors 75
6.1 Introduction .............................................................. 75
6.2 Related Work ............................................................ 77
6.3 Unifying Sampling Latent Variable and Conditioning ................ 78
  6.3.1 Stochastically Conditioning the Decoder ..................... 80
  6.3.2 Learning ............................................................ 81
6.4 Experiments .............................................................. 82
  6.4.1 Diverse Human Motion Prediction .............................. 83
    Datasets ............................................................... 83
    Evaluation Metrics ................................................... 84
    Evaluating Stochasticity ............................................ 85
    Evaluating Sampling Quality ..................................... 85
  6.4.2 Diverse Image Captioning .................................... 87
6.5 Conclusion ............................................................... 88
## Contents

7 Conclusion 89  
7.1 Future Work .............................................. 91  

A Appendix A 93  
A.1 Technical Background on Evidence Lower Bound ................. 93  
A.2 Derivation of LCP-VAE’s KL Divergence Loss .................... 94  
A.3 Mitigating Posterior Collapse: Related Work .................... 95  
A.4 Further Discussion on the Performance of Stochastic Baselines 96  
A.5 Ablation Study on Different Means of Conditioning ............ 97  
A.6 Experimental Results on the Penn Action Dataset ............... 98  
A.7 Stochastic Human Motion Prediction Architecture ............... 98  
A.8 Diverse Image Captioning Architecture ........................ 100  
A.9 Diverse Text Generation: Related Work ........................ 101  
A.10 Ablation Study on Diverse Image Captioning .................... 101  
A.11 Additional Motion Prediction Qualitative Results ............... 102  
A.12 Additional Image Captioning Qualitative Results ............... 102
List of Figures

2.1 VGG16 architecture. ................................................. 6
2.2 Class Activation Maps ............................................... 7
2.3 Densely-connected convolutional network architecture. ............ 8
2.4 In a standard VAE, the model learns stochastic mappings between the observed dataset whose empirical distribution is $p_D(x)$, typically a complicated one, and a latent space whose distribution is relatively simpler, e.g., $\mathcal{N}(0, I)$. ............ 12

3.1 Overview of our approach. Given a small portion of sequential data, our approach is able to predict the action category with very high performance. For instance, in UCF-101, our approach anticipates actions with more than 80% accuracy given only the first 1% of the video. To achieve this, we design a model that leverages action- and context-aware features together with a new loss function that encourages the model to make correct predictions as early as possible. .................. 16

3.2 Overview of our approach. We propose to extract context-aware features, encoding global information about the scene, and combine them with action-aware ones, which focus on the action itself. To this end, we introduce a multi-stage LSTM architecture that leverages the two types of features to predict the action or forecast it. Note that, for the sake of visualization, the color maps were obtained from 3D tensors ($512 \times W \times H$) via an average pooling operation over the 512 channels. .................. 18

3.3 Our feature extraction network. Our CNN model for feature extraction is based on the VGG-16 structure with some modifications. Up to conv5-2, the network is the same as VGG-16. The output of this layer is connected to two sub-models. The first one extracts context-aware features by providing a global image representation. The second one relies on another network to extract action-aware features. .................. 20

3.4 Action-aware feature extraction. Given the fine-tuned feature extraction network, we introduce a new layer that alters the output of conv5-3. This lets us filter out the conv5-3 features that are irrelevant, to focus on the action itself. Our action-aware features are taken as the output of the last fully-connected layer. .................. 22

3.5 Comparison of different losses for action anticipation on UCF-101. We evaluate the accuracy of our model trained with different losses as a function of the number of frames observed. This plot clearly shows the superiority of our loss function. .................. 26
3.6 **Influence of our average pooling strategy.** Our simple yet effective average pooling leverages the predictions of all the frames up to time $t$. As shown on the UCF-101 dataset, this increases anticipation performance, especially at very early stages. .................................................. 27

3.7 Visualization of action-aware and context-aware features on UCF-101. These samples are representative of the data. ................................................................. 29

4.1 **Overview of our data collection.** Using the GTA V environment and driving equipment depicted in the left box, we captured a new dataset covering 5 generic scenarios, illustrated in the right box, each containing multiple action classes. ................................................................. 38

4.2 Samples from different scenarios of VIENA$^2$. ................................. 39

4.3 **Statistics for each scenario of VIENA$^2$.** We plot the number of videos per class, and proportions of different weather conditions (clear in yellow vs rainy/snowy in gray) and different daytime (day in orange vs night in blue). Best seen in color. ................................................................. 41

4.4 (Left) Our Multi-Stage LSTM architecture. (Right) Visualization of our weighting function for the anticipation loss of Eq. 4.1 ................................................. 44

4.5 **Confusion Matrices.** Confusion matrices of all five scenarios after observing 1 second (top) and 5 seconds (bottom) of each video sample. ......................... 48

4.6 **Effect of the amount of real training data for fine-tuning MM-LSTM.** MM-LSTM was pre-trained on VIENA$^2$ in all cases, except for From Scratch w/100% of JAAD (dashed line). Each experiment was conducted with 10 random subsets of JAAD. We report the mean accuracy and standard deviation (error bars) over 10 runs. ................................................................. 50

4.7 Examples of cars used to capture our data. ........................................ 55

4.8 Examples of heavy vehicles (trucks and buses) used to capture data for heavy vehicles’ driver maneuvers. ................................................. 55

4.9 Examples of different daytime conditions. From top-left to bottom-right: Predawn, dawn, sunrise, morning, midday, sunset, dusk, and midnight. ............... 56

4.10 Examples of different weather conditions (from a single view/location), including rainy, snowy, foggy, hazy, and clear. ................................................. 56

4.11 Examples of different locations, including suburbs, downtown, off-road, nature, highway, and industrial locations. ................................................. 57

4.12 Examples of our action classes in different scenarios. Each row represents an action example from one scenario. ................................................. 57
5.1 Mix-and-Match perturbation. (Top) Illustration of the Sampling operation (left) and of the Resampling one (right). Given a sampling rate \( \alpha \) and a vector length \( L \), the Sampling operation samples \( \lceil \alpha L \rceil \) indices, say \( I \). The complementary, unsampled indices are denoted by \( \bar{I} \). Then, given two \( L \)-dimensional vectors and the corresponding \( \lceil \alpha L \rceil \) and \( \lfloor (1 - \alpha)L \rfloor \) indices, the Resampling operation mixes the two vectors to form a new \( L \)-dimensional one. (Middle) Example of Mix-and-Match perturbation. (Bottom) Example of perturbation by concatenation, as in [Yan et al., 2018]. Note that, in Mix-and-Match perturbations, sampling is stochastic; the indices are sampled uniformly randomly for each mini-batch. By contrast, in [Yan et al., 2018], sampling is deterministic, and the indices in \( I \) are fixed and correspond to \( I = \{1, \ldots, \frac{L}{2}\} \).

5.2 Overview of our approach. (Top): Overview of the model during inference. During inference, given past information and a random vector sampled from a Normal distribution, the model generates new motions. (Bottom): Overview of the model during training. During training, we use a future pose autoencoder with a CVAE between the encoder and the decoder. The RNN encoder-decoder network mapping the past to the future then aims to generate good conditioning variables for the CVAE.

5.3 Example of curriculum perturbation of the hidden state.

5.4 Qualitative evaluation of diversity. The first row (black box) shows the ground-truth motion. The next six rows depict six randomly generated motions (not cherry-picked) given the same observations (the first four poses of each motion). The green box shows the last observed frame and the first generated one, illustrating the consistency of the generated motions. The orange boxes show the diversity of the generated motions in different temporal windows. The blue box shows a randomly sampled motion whose poses are similar to the ground-truth ones. Best seen in color and zoomed in.

5.5 Diversity of \( K \) RNN decoder inputs, generated with \( K = 50 \) different random vectors. We report the mean diversity over \( N = 50 \) samples and the corresponding standard deviation.

5.6 (Top) Diversity of our approach and the stochastic baselines. (Middle) Quality of our approach and the stochastic baselines. (Bottom) Comparing classifier-based and human evaluation of quality for our approach and the baselines, where the statistics correspond to evaluation after the models are fully trained. The numbers are provided in the supplementary material to facilitate future comparisons.
6.1 Training and inference for a standard CVAE. (Left) During the training phase, the encoder takes as input the combination of the data and the corresponding condition and compresses it into the latent space. The decoder then samples a latent variable from the approximate posterior computed by the encoder, combines the latent variable with the conditioning signal, and reconstructs the data. The approximate posterior distribution of all training samples, i.e., for all conditioning signals, is encouraged to match the prior distribution. (Right) During inference, the decoder samples different latent variables from the prior distribution, combines them with the condition, and generates samples that all respect the conditioning signal. However, there is no guarantee that the latent variable is sampled from the region of the prior that corresponds to the given condition. 76

6.2 Illustration of a CVAE architecture and an LCP-VAE architecture. (Top) In a standard CVAE architecture, the latent variable $z$ is sampled from a standard Normal distribution, independently of the condition. This leads to the reparameterization trick illustrated on the right. In this case, the approximate posterior is normally distributed. (Bottom) In an LCP-VAE, the sampling of $z$ is conditioned on the CVAE condition via the latent variable $z_c$. Specifically, the posterior distribution of the condition acts as prior on the data posterior. This corresponds to the extended reparameterization trick illustrated on the right. Note that the approximate posterior is not normally distributed anymore. 79

6.3 Inference procedure in an LCP-VAE and a CVAE. (Left) In a standard CVAE, at inference time, since the prior distribution is independent of the conditioning signal, we are unlikely to sample a latent variable that corresponds to the region of the latent space that is related to the observed condition. (Right) In an LCP-VAE, at inference time, since we have access to the conditioning signal, we use the CS-VAE encoder to approximate the posterior of each condition. To generate a sample given a condition, the LCP-VAE decoder then samples a latent variable from the posterior of its condition, instead of using a general prior distribution as in CVAEs, and generate a sample. For instance, $\hat{x}_j$ and $\hat{x}_l$ are conditioned on $C_4$ and their corresponding latent variables are sampled from a very similar region. By contrast, $\hat{x}_k$ is generated by a latent variable sampled from a completely different region, i.e., the region corresponding to the approximate posterior of $C_5$. 81

6.4 Qualitative evaluation of the diversity in human motion. The first row depicts the ground-truth motion. The first six poses of each row correspond to the observation (the condition) and the rest are sampled from our model. Each row is a randomly sampled motion (not cherry picked). Note that all sampled motions are natural, with a smooth transition from the observed poses to the generated ones. Diversity increases with the length of the generated sequence. 83
6.5 t-SNE plots of the posterior mean for 3750 test motions. With MT-VAE [Yan et al., 2018], all classes are mixed, suggesting that the latent variable carries little information about the motions. By contrast, our condition-dependent sampling allows LCP-VAE to better preserve context. Note that some actions, such as discussion and directions, are very hard to identify and are thus spread over other actions. Others, such as walking, walking with dog, and walking together, or sitting and sitting down overlap due to their similarity. 85

6.6 Qualitative evaluation of the diversity in generated captions. While captions generated by our approach are diverse, they all describe the image properly. The caption from the mode also typically provides an accurate description. 87

A.1 Qualitative evaluation of the diversity in human motion. The first row illustrates the ground-truth motion. The first six poses of each row depict the observation (the condition) and the rest are sampled from our model. Each row is a randomly sampled motion (not cherry picked). As can be seen, all sampled motions are natural, with a smooth transition from the observed to the generated ones. The diversity increases as we increase the sequence length. 103

A.2 Additional qualitative evaluation of the diversity in human motion. 104
A.3 Additional qualitative evaluation of the diversity in human motion. 104
A.4 Additional qualitative evaluation of the diversity in human motion. 105
A.5 Additional qualitative evaluation of the diversity in human motion. 105
A.6 Qualitative evaluation of the diversity in generated captions. While captions generated by our approach are diverse, they all describe the image properly. The caption from mode also usually achieves a good descriptive caption. 106
A.7 Additional qualitative evaluation of the diversity in generated captions. 106
A.8 Additional qualitative evaluation of the diversity in generated captions. 107
A.9 Additional qualitative evaluation of the diversity in generated captions. 107
A.10 Additional qualitative evaluation of the diversity in generated captions. 108
3.1 Comparison with state-of-the-art baselines on the task of action anticipation on the JHMDB-21 dataset. Note that our approach outperforms all baselines significantly in both settings. ................................................................. 25
3.2 Comparison with state-of-the-art baselines on the task of action anticipation on the UT-Interaction dataset. .................................................................................................................. 25
3.3 Action Anticipation on the UCF-101 dataset. ................................................................................................................................. 25
3.4 Importance of the different feature types. ........................................................................................................................................ 28
3.5 Comparison with state-of-the-art methods on UCF-101 (average accuracy over all training/testing splits). For the comparison to be fair, we focus on the baselines that, as us, only use the RGB frames as input. ................................................................. 30
3.6 Comparison with state-of-the-art methods on JHMDB-21 (average accuracy over all training/testing splits). Note that while the methods of [Gkioxari and Malik, 2015; Wang et al., 2016a; Soomro et al., 2016a,b] use motion/optical flow information and [Soomro et al., 2016a] uses IDT Fisher vector features, our method yields better performance. ................................................................. 31
3.7 Comparison with state-of-the-art methods on UT-Interaction (average accuracy over all training/testing splits). Note that while the methods of [Soomro et al., 2016a] uses motion/optical flow information and IDT Fisher vector features, our method yields better performance. ................................................................. 31
3.8 Comparison with the state-of-the-art approaches that use optical flow. For the comparison to be fair, we focus on the baselines that, as us, use RGB frames+optical flow as input. ................................................................. 32
3.9 Importance of the different feature types using different losses. Note that combining both types of features consistently outperforms using a single one. Note also that, for a given model, our new loss yields higher accuracies than the other ones. ........................................................................................................................................ 33
3.10 Influence of the number of hidden LSTM units and of our average pooling strategy in our multi-stage LSTM model. These experiments were conducted on the first splits of UCF-101 and JHMDB-21. ................................................................. 34
3.11 Comparison of our multi-stage LSTM model with diverse fusion strategies. We report the results of simple concatenation of the context-aware and action-aware features, their use in two parallel LSTMs with late fusion, and swapping their order in our multi-stage LSTM, i.e., action-aware first, followed by context-aware. Note that multi-stage architectures yield better results, with the best ones achieved by using context first, followed by action, as proposed in this chapter. ........................................................................................................................................ 34
4.1 Statistics comparison with action recognition and anticipation datasets. A * indicates a dataset specialized to one scenario, e.g., driving, as opposed to generic. .................................................. 42

4.2 Results on the Random split of VIENA$^2$ for the original versions our three baselines: CNN+LSTM [Donahue et al., 2015a] with only appearance, Two-Stream [Feichtenhofer et al., 2016a] with appearance and motion, and MS-LSTM (introduced in Chapter 3) with action-aware and context-aware features. 47

4.3 Results on the Random split of VIENA$^2$ for our three baselines with our action descriptors and for our approach. ................................................................. 47

4.4 Per-class accuracy of our approach on all scenarios of VIENA$^2$ (Random). . . . 48

4.5 Anticipating actions on real data. Pre-training our MM-LSTM with our VIENA$^2$ dataset yields higher accuracy than training from scratch on real data. 49

4.6 Accuracy of MM-LSTM on the Driver Maneuver scenario when the training and test sets were captured by different users. ................................................. 51

4.7 Results on the Weather split of VIENA$^2$ for all scenarios and for our three baselines and our approach. ................................................................. 51

4.8 Results on the Daytime split of VIENA$^2$ for all scenarios and for our three baselines and our approach. ................................................................. 52

4.9 Influence of the loss function. We compare the use of the standard cross-entropy, the loss introduced in Chapter 3 and our loss within our MM-LSTM model using the Accidents of VIENA$^2$. Note that our loss consistently yields higher accuracy, particularly in the early stages of the videos. ..................... 52

4.10 Influence of different features. We evaluate the accuracy obtained by using our different features individually on the Accidents and Front car intention scenarios for the Random partition. See the discussion in the text for more detail. 52

4.11 Effect of number of LSTM hidden units on anticipation performance. Using 1024 units represents a good trade-off between memory consumption and accuracy. .................................................. 53

4.12 Importance of the Input Block to learn a good combination of the modalities. We report the mean accuracy and standard deviation over the 24 permutations of the 4 input modalities for the Accident scenario. Note that using both modality-specific LSTMs (L) and the FC-Pool (P) layer yields not only the highest accuracies, but also very low std, thus indicating that the modality order has virtually no impact on our results. ................................. 54

4.13 Comparison to the state-of-the-art anticipation techniques on JHMDB-21. Note that our MM-LSTM architecture outperforms all the baselines, thus achieving state-of-the-art performance on this dataset. ..................... 54

5.1 Comparison of our approach with the stochastic motion prediction baselines on Human3.6M dataset (left) and CMU Mocap dataset (right). Tr KL stands for KL term at training convergence. .................................................. 67

5.2 Quantitative comparison of the S-MSE against stochastic baselines for four actions of the Human3.6M dataset. .................................................. 71
5.3 Comparison against deterministic motion prediction techniques for four actions of the Human3.6M dataset. ................................................................. 72

6.1 Comparison of LCP-VAE with the stochastic motion prediction baselines on Human3.6M (top) and CMU MoCap (bottom). .............................. 84

6.2 Comparison of the generated motions with the ground-truth future motions in terms of context. The gap between the performance of the state-of-the-art pose-based action classifier [Li et al., 2018b] with and without true future motions is 0.22 / 0.54. Using our predictions, this gap decreases to 0.06 / 0.08, showing that our predictions reflect the class label. H3.6M / CMU represent Human3.6M and CMU MoCap, respectively. .............................................. 84

6.3 Comparison with the state-of-the-art stochastic motion prediction models for 4 actions of Human3.6M (all methods use the best of $K = 50$ sampled motions). ................................................................. 86

6.4 Comparison with the state-of-the-art deterministic models for 4 actions of Human3.6M. ................................................................. 86

6.5 Quantitative evaluation of stochastic image captioning on the MSCOCO Captioning dataset. ................................................................. 87

A.1 Evaluation of various architecture designs for a CVAE. A smaller KL value, indicating posterior collapse, leads to less diversity. ......................... 98

A.2 Quantitative evaluation on the Penn Action dataset. Note that a diversity of 1.21 is reasonably high for normalized 2D joint positions, i.e., values between 0 and 1, normalized with the width and the height of the image. ............... 98

A.3 BLEU scores of different orders for sampled captions from our model as well as the baselines. ........................................................................ 102
The following publications have resulted from the work presented in this thesis:

- Sadegh Aliakbarian, Fatemeh Sadat Saleh, Mathieu Salzmann, Basura Fernando, Lars Petersson, and Lars Andersson. *Encouraging lstms to anticipate actions very early*. In Proceedings of the IEEE International Conference on Computer Vision, pp. 280-289. 2017.

- Sadegh Aliakbarian, Fatemeh Sadat Saleh, Mathieu Salzmann, Basura Fernando, Lars Petersson, and Lars Andersson. *VIENA²: A Driving Anticipation Dataset*. In Asian Conference on Computer Vision, pp. 449-466. Springer, Cham, 2018.

- Sadegh Aliakbarian, Fatemeh Sadat Saleh, Mathieu Salzmann, Lars Petersson, and Stephen Gould. *A Stochastic Conditioning Scheme for Diverse Human Motion Prediction*. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5223-5232. 2020.

- Sadegh Aliakbarian, Fatemeh Sadat Saleh, Mathieu Salzmann, Lars Petersson, and Stephen Gould. *Sampling Good Latent Variables via CPP-VAEs: VAEs with Condition Posterior as Prior*. arXiv preprint arXiv:1912.08521 (2019). To be submitted to CVPR 2021.
Introduction

As for many other computer vision tasks, a recognition model gets as input the whole observation, e.g., an image or a video, and predicts a label. For instance, in video/action recognition, a model gets as input all the information in the video and predicts a label that represents the most likely category/event happened in that video. Although understanding such information given the full extent of videos is key to the success of wide variety of applications, the full observation is not always available in many real-world cases, such as autonomous navigation, surveillance and sports analysis. In these application, it is crucial to reliably predict the future action of a particular sequence. For instance, an autonomous car should always predict the intention of all nearby pedestrians a few seconds ahead to e.g., avoid hitting them.

Video anticipation is one of the key challenges in computer vision and video analysis. Anticipation is the task of predicting future representation(s) given limited, partial observation. Having a reliable anticipation system is highly critical in scenarios where one needs to react before the action is finalized, such as in pedestrian intention forecasting system in autonomous vehicles. Sequence anticipation is a challenging task due to the fact that given limited observation, the future representation can be highly ambiguous. Given the task, one may anticipate a coarse, discrete representation per limited observation, called discrete anticipation, or a fine-grained, continuous representation, called continuous anticipation. There are also cases where the future representation can be considered deterministic, i.e., expecting one single likely outcome for a given observation, or can be considered stochastic, where multiple plausible future representations are likely and correct.

In this chapter, we first provide a definition of the problem as well as the motivations behind this research. We then summarize our contributions and the thesis outline.

1.1 Problem Definition

Discrete and Deterministic Anticipation. In a discrete anticipation problem, the goal is to predict the desired outcome $y_T$ given the partial observation $x_{1:t}$ where $T$ is the length of the sequence and $t < T$ is the length of the partial observation. The desired outcome $y_T$ can be a binary value determining whether what will happen in the future is normal or abnormal or can be a categorical determining the action of interest at time $T$. For discrete anticipation, we specifically study the problem of action anticipation in this thesis. In contrast to the widely studied problem of recognizing an action given a complete sequence, as introduced above,
action anticipation aims to identify the action from only partially available videos. As such, it is therefore key to the success of computer vision applications requiring to react as early as possible, such as autonomous navigation. Note that, when addressing discrete anticipation task, we focus on only a deterministic scenario, where we are interested in predicting only one \( y_T \) that has the highest likelihood of occurrence.

**Continuous and Stochastic Anticipation.** In a continuous anticipation task, the goal is to predict a sequence of desired outcomes \( y_{t+1:T} \) given the partial observation \( x_{1:t} \) where \( T \) is the length of the sequence and \( t < T \) is the length of the partial observation. The main difference to discrete anticipation is one need to anticipate the representation at every time-step in the future such that the whole sequence remains natural and plausible. Unlike discrete anticipation where the outcome representation (e.g., action label) differs from the input representation (e.g., consecutive RGB frames), in continuous anticipation these two representation are usually from the same domain, casting video anticipation problem into future representation generation. Examples of continuous anticipation is trajectory prediction, video generation, and human motion prediction, where in this thesis we focus on the latter. Similar to many other real-world tasks in continuous anticipation, human motion prediction is a highly stochastic process: given an observed sequence of poses, multiple future motions are plausible. Thus, it is crucial to take stochasticity of this problem into account. Therefore, in this thesis we focus on stochastic scenario and try to generate multiple likely continuations of an observed human motion.

### 1.2 Thesis Outline and Contributions

This thesis comprises seven chapters. Apart from this chapter that introduces the problem and provides our motivations to address the problem video anticipation, in Chapter 2, we introduce the technical background of the methods we used in this thesis. This background material consists of some concepts and theories that are common to many of the approaches proposed in later chapters, including the convolutional neural network architectures, recurrent neural networks, and variational autoencoders. The next four chapters provide details of our novel techniques for sequence learning for video anticipation. Particularly, in Chapter 3, we introduce our novel techniques for encouraging LSTMs to anticipate actions as early as possible. In Chapter 4, we introduce a new anticipation-specific dataset as well as a novel multi-modal LSTM architecture that generalizes our previous contribution to an arbitrary number of modalities. In Chapter 5, we address the problem of human motion prediction with the focus on the stochastic nature of this problem. Then, in Chapter 6, we propose a novel framework for generating diverse and semantically high quality sequences given partial observation. Finally, in Chapter 7, we summarise the main contributions of the thesis and discuss ongoing and future work stemming from this research.

A brief summary of each contribution is provided below.

#### 1.2.1 General Action Anticipation

We propose a novel action anticipation framework (that can be also seen as an *early recognition* model). In particular, we introduce a novel loss that encourages making correct predictions
very early. Our loss models the intuition that some actions, such as running and high jump, are highly ambiguous after seeing only the first few frames, and false positives should therefore not be penalized too strongly in the early stages. By contrast, we would like to predict a high probability for the correct class as early as possible, and thus penalize false negatives from the beginning of the sequence. Our experiments demonstrate that, for a given model, our new loss yields significantly higher accuracy than existing ones on the task of early prediction. We also propose a novel multi-stage Long Short Term Memory (MS-LSTM) architecture for action anticipation. This model effectively extracts and jointly exploits context- and action-aware features. This is in contrast to existing methods that typically extract either global representations for the entire image or video sequence, thus not focusing on the action itself, or localize the feature extraction process to the action itself via dense trajectories, optical flow or actionness, thus failing to exploit contextual information [Sadegh Aliakbarian et al., 2017].

1.2.2 Action Anticipation in Driving Scenarios

We improve and extend our previous contribution by focusing on driving scenarios, encompassing common the subproblems of anticipating ego car’s driver maneuvers, front car’s driver maneuver, accidents, violating or respecting traffic rules, and pedestrian intention, with a fixed, sensible set of sensors. To this end, we introduce the VIRTUAL Environment for Action Analysis (VIENA$^2$) dataset. Altogether, these subproblems encompass a total of 25 distinct action classes. VIENA$^2$ is acquired using the GTA V video game. It contains more than 15K full HD, 5s long videos, corresponding to more than 600 samples per action class, acquired in various driving conditions, weathers, daytimes, and environments. This amounts to more than 2.25M frames, each annotated with an action label. These videos are complemented by basic vehicle dynamics measurements, reflecting well the type of information that one could have access to in practice$^1$. We then benchmark state-of-the-art action anticipation algorithms on VIENA2, and as another contribution, introduce a new multi-modal, LSTM-based architecture that generalizes out previous contribution to an arbitrary number of modalities, together with a new anticipation loss, which outperforms existing approaches in our driving anticipation scenarios [Aliakbarian et al., 2018a].

1.2.3 A Stochastic Conditioning Scheme for Diverse Sequence Generation

For continuous, stochastic anticipation task, we address the problem of stochastic human motion prediction. As introduced earlier in this thesis, human motion prediction aims to forecast the sequence of future poses of a person given past observations of such poses. To achieve this, existing methods typically rely on recurrent neural networks (RNNs) that encode the person’s motion. While they predict reasonable motions, RNNs are deterministic models and thus cannot account for the highly stochastic nature of human motion; given the beginning of a sequence, multiple, diverse futures are plausible. To correctly model this, it is therefore critical to develop algorithms that can learn the multiple modes of human motion, even when presented with only deterministic training samples. We introduce an approach to effectively learn the stochasticity in human motion. At the heart of our approach lies the idea of Mix-and-

$^1$Our dataset is publicly available at [https://sites.google.com/view/viena2-project](https://sites.google.com/view/viena2-project)
Introduction

Match perturbations: Instead of combining a noise vector with the conditioning variables in a deterministic manner (as usually done in standard practices), we randomly select and perturb a subset of these variables. By randomly changing this subset at every iteration, our strategy prevents training from identifying the root of variations and forces the model to take it into account in the generation process. This is a highly effective conditioning scheme in scenarios when (1) we are dealing with a deterministic dataset, i.e., one sample per condition, (2) the conditioning signal is very strong and representative, e.g., the sequence of past observations, and (3) the model has an expressive decoder that can generate a plausible sample given only the condition. We utilize Mix-and-Match by incorporating it into a recurrent encoder-decoder network with a conditional variational autoencoder (CVAE) block that learns to exploit the perturbations. Mix-and-Match then acts as the stochastic conditioning scheme instead of concatenation that usually appears in standard CVAEs [Aliakbarian et al., 2020a].

1.2.4 Variational Autoencoders with Learned Conditional Priors

In our previous contribution, we identified one limitation of a standard CVAEs when dealing deterministic datasets and strong conditioning signals. In this work, we further investigates this problem from a more theoretical point of view. Specifically, in this contribution we tackle the task of diverse sequence generation in which all the diversely generated sequences carry the same semantic as in the conditioning signal. We observe that in standard CVAE, conditioning and sampling the latent variable are two independent processes, leading to generating samples that are not necessarily carry all the contextual information about the condition. To address this, we propose to explicitly make the latent variables depend on the observations (the conditioning signal). To achieve this, we develop a CVAE architecture that learns a distribution not only of the latent variables, but also of the observations, the latter acting as prior on the former. By doing so, we change the variational family of the posterior distribution of the CVAE, thus, as a side effect, our approach can mitigate posterior collapse to some extent [Aliakbarian et al., 2019].
Technical Background

In this chapter we introduce the background materials, architectures and models that have been used in this thesis. We use Convolutional Neural Networks (CNNs) widely as the feature/representation learning approach for discrete anticipation task. We employ variants of Recurrent Neural Networks (RNNs) to effectively model sequences. Finally, since we cast the problem of continuous anticipation as generating future representation, we use Variational Autoencoders (VAEs) as our backbone generative model in this thesis. Below, we review the utilized CNN architectures, the variants of RNNs used in this thesis, and VAEs to help the reader better understand the following chapters.

### 2.1 Convolutional Neural Networks

Convolutional neural networks [Karpathy et al., 2014a], or CNNs in short and also known as ConvNets, are powerful tools used in many machine learning and computer vision tasks. With the advances in deep learning and compute, CNNs achieve the state-of-the-art performance on various tasks such as image recognition [Simonyan and Zisserman, 2014c; Karpathy et al., 2014a; He et al., 2016; Huang et al., 2017; Krizhevsky et al., 2012], semantic segmentation [Long et al., 2015; Noh et al., 2015; Saleh et al., 2017b; Zhao et al., 2017a], action recognition [Wang et al., 2016b; Carreira and Zisserman, 2017; Simonyan and Zisserman, 2014a; Feichtenhofer et al., 2016a; Wu et al., 2019; Aliakbarian et al., 2016], object detection [Redmon et al., 2016; Tychsen-Smith and Petersson, 2017; Ren et al., 2015; Girshick, 2015; Girshick et al., 2014], etc. CNNs comprise stack of layers of different types, including but not limited to convolutional, pooling, and fully-connected (also known as linear) layers. Common CNN architectures stack blocks of these operations to perform feature representation learning which is done by learning trainable parameters of each layer. Since, in most cases, different layers are stacked on top of each other, deeper layers perform operations on the output of shallower layers, thus learning the feature representation in a hierarchical manner. Hence, it is usually the case where the features computed by the deeper layers carry out more semantically meaningful information while the features of shallower layers provide detailed, but low-level information [Saleh et al., 2016; Bertasius et al., 2015a]. This characteristic of CNNs allows us to use the output features of deeper layers as a good representation of the input image [Wang et al., 2016b; Carreira and Zisserman, 2017; Simonyan and Zisserman, 2014a; Feichtenhofer et al., 2016a; Wu et al., 2019]. In this thesis, we use CNNs as feature extractors for represent-
In this section, we assume that the reader has a knowledge about CNNs building blocks, layers, and optimization, thus, we only briefly introduce the special CNN architectures we used later in this thesis.

There are many powerful CNN architectures that shown effective in many computer vision tasks. Among those, we use VGG16 [Simonyan and Zisserman, 2014c] and DenseNet [Huang et al., 2017] architectures in the following chapters. In this section we describe these two architectures briefly.

**VGG16.** Introduced in 2014, VGG16 [Simonyan and Zisserman, 2014c] considered to be a very deep network. This network, as shown in Fig. 2.1 comprises five convolutional blocks followed by three fully-connected layers. Each convolutional block contains convolutional, pooling, and nonlinearity layers. Pre-trained on ImageNet dataset [Krizhevsky et al., 2012] and fine-tuned on the target dataset, this network can extract discriminative features from each image.

In this thesis, followed by the idea of Class Activation Map (CAM) [Zhou et al., 2016a], we modified the VGG16 architecture such that is becomes capable of generating heatmaps that illustrate the most discriminative part of image in determining a particular class. As depicted in Fig. 2.2, one can add a global average pooling (GAP) layer [Zhou et al., 2016a] after the fifth convolutional block of the VGG16 architecture. The output of GAP is then used as the input features for a fully-connected layer that maps the features to the number of classes. Given such design, one can generate a coarse representation that determines the most discriminative part of the input simply by projecting back the weights of the output layer on to the last convolutional feature maps. More specifically, let \( f_l(x,y) \) represent the activation of unit \( l \) in the last convolutional layer at the spatial location \((x,y)\). A score \( S_k \) for each class \( k \) can be obtained by performing GAP to obtain, or each unit \( l \), a feature \( F^l = \sum_{x,y} f_l(x,y) \), followed by a fully-connected layer with the set of weights \( \{w^l_k\} \). That is,
\[ S_k = \sum_k w_k^k F_l. \] (2.1)

A CAM for a class \( k \) at location \((x, y)\) can be then computed as

\[ M_k(x, y) = \sum_l w_l^k f_l(x, y), \] (2.2)

which indicates the importance of the activations at location \((x, y)\) in the final score for class \( k \).

**DenseNet.** Densely-connected convolutional neural network [Huang et al., 2017], or DenseNet in short, was proposed in 2016 to address the issue of maximum information and gradient flow. To this end, as illustrated in Fig. 2.3, in DenseNet, every layer gets as input the output of all previous layers, thus, is connected to every other layers directly. Thus, since each layer has a direct access to the gradient computed by the loss directly, the vanishing gradient problem is remedied to a certain degree.

Another good characteristic of DenseNet is that, unlike simpler architectures such as VGG16, the output of the deepest layer contains features at multiple semantic level, i.e., very rich semantic features as well as low level features representing edges or texture.
2.2 Recurrent Neural Networks

Throughout this thesis, we widely use RNNs to learn the sequences in both cases of discrete and continuous anticipation tasks. In this section, we first describe the main concept of RNNs briefly, then discuss the two variants of RNNs, namely Long-Short Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997] and Gated Recurrent Unit (GRU) [Cho et al., 2014], which we later use in the following chapters.

2.2.1 Vanilla RNN.

There are many tasks that are sequential in nature and often cannot be casted as the ones with fixed-length/fixed-size inputs and outputs, e.g., speech recognition [Graves et al., 2013], machine translation [Sutskever et al., 2014; Maruf et al., 2019; Chung et al., 2014], and video analysis [Donahue et al., 2015b]. In a sequential problem it is crucial that the decision at each time-step being made based on both the observation at that given time-step and all previous observations since samples at different time-steps are not independent of each other. RNNs perform exactly the same task by taking a representation of previous outputs into account while processing the input of the current time-step by updating and using its hidden state. At a given time, each RNN cell takes as input the representation of the current time-step and the hidden state that has been updated given the representation of all previous time-steps. Then, it updates the hidden state given the previous hidden state and the current input, and optionally can generate an output. Since RNNs require updating the hidden state at each time-step, they work sequentially.

More specifically, RNNs learn the temporal relations by mapping the input sequence to hidden states, and then mapping the hidden states to the desired output. Given \( x_t \), the input at time \( t \), the hidden state is updated as

\[
h_t = g(W_{hh}h_{t-1} + W_{xh}x_t),
\]

where \( h_{t-1} \) is the hidden state at the previous time-step, \( g \) is the nonlinearity function such as \( \tanh \), and \( W_{hh} \) and \( W_{xh} \) are the set of learnable weights. One can also generate an output, \( y_t \), at each time-step by computing

\[
y_t = g(W_{hy}h_t),
\]
where $W_{hy}$ is a set of learnable weights that maps the hidden state to the output.

Although shown effective in many sequence learning problems, RNNs suffer from the vanishing/exploding gradient that makes them hard to train for the task requires learning long-range dependencies. To remedy this issue LSTM has been proposed.

**LSTM.** Long-short term memory unit, or LSTM in short, is a variant of RNN which is proposed to address the problem of vanishing/exploding gradient when dealing with long sequences. What makes LSTMs applicable for learning longer-range dependencies is the concept of memory, in which it allows the model decide when to forget and when to use and update the previous hidden state given the input at the current time-step. LSTM comes with a gating of the form

$$c_t = c_{t-1} + i_t \odot g_t,$$

(2.5)

where $c_t$ is a recurrent state, $i_t$ is a gating function, and $g_t$ is a full update. Unlike vanilla RNN, such gating assures that the derivatives of the loss with respect to the recurrent state $c_t$ does not vanish.

The recurrent states comprises a cell state, $c_t$, and an output state, $h_t$. The gate $f_t$ modulates if the cell state should be forgotten and the gate $i_t$ modulates if the new update should be taken into account. The gate $o_t$ modulates in the case that output state should be reset. The structure of these gates and states are

- **Forget gate:** $f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$
- **Input gate:** $i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$
- **Full cell state update:** $g_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$
- **Cell state:** $c_t = f_t \odot c_{t-1} + i_t \odot g_t$
- **Output gate:** $o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$
- **Output state:** $h_t = o_t \odot \tanh(c_t)$

where $\sigma$ is the sigmoid activation function.

In standard practice, the forget gate bias $b_f$ should be initialized with a large value such that $f_t \approx 1$ and the gating has no effect [Gers et al., 1999]. Note that a prediction $y_t$ can be made given the hidden state $h_t$, so that is why the $h_t$ is called the output state.

**GRU.** The LSTM can be simplified in the a gated recurrent unit, or GRU in short, with a gating for the recurrent state and a reset gate. Therefore, one can easily have only the following gates and updates

- **Reset gate:** $r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r)$
- **Forget gate:** $z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z)$
Technical Background

- Full update: $\tilde{h}_t = \tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h)$
- Hidden update: $h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t$

### 2.3 Variational Autoencoders

Consider we have a model $p_\theta(x, z)$, that works with a set of observed variables $x$, which we have access to given a dataset, and a set of unobserved latent variables $z$, which typically are not available in the dataset. This model $p_\theta(x, z)$ aims to model the observations, i.e., $x$ through a neural network parameterized with $\theta$. The likelihood of the data can then be computed as the marginal distribution over the observed variables

\[
p_\theta(x) = \int_z p_\theta(x, z) dz, \tag{2.6}
\]

which is also called the marginal likelihood or the evidence of a model. $p_\theta(x, z)$ is called a deep latent variable model, or DLVM in short, since its distributions are parameterized by a neural network $\theta$. In a simple case, one can define $p_\theta(x, z)$ as a factorization,

\[
p_\theta(x, z) = p_\theta(z)p_\theta(x|z). \tag{2.7}
\]

In such case, $p_\theta(z)$ is called the prior distribution over the latent variable $z$. Learning the maximum likelihood in such model is not straightforward since $p_\theta(x)$ is not tractable [Kingma and Welling, 2013]. This is related to the intractability of the posterior distribution $p_\theta(z|x)$, which can be computed as

\[
p_\theta(z|x) = \frac{p_\theta(x, z)}{p_\theta(x)}. \tag{2.8}
\]

To solve the maximum likelihood problem, we would like to have $p_\theta(x|z)$ and $p_\theta(z)$. Using Variational Inference, we aim to approximate the true posterior $p_\theta(z|x)$ with another distribution $q_\phi(z|x)$ which is computed with another neural network parameterized with $\phi$ (called variational parameters), such that $q_\phi(z|x) \approx p_\theta(z|x)$. Using such approximation, Variational Autoencoders [Kingma and Welling, 2013], or VAEs in short, are able to optimize the marginal likelihood in a tractable way. The optimization objective of the VAEs is variational lower bound, also known as evidence lower bound, or ELBO in short. Recall that variational inference aims to find an approximation of the posterior that represent the true one. One way to do this is to minimize the divergence between the approximate and true posterior using Kullback–Leibler divergence [Kullback and Leibler, 1951], or KL divergence in short, that is,
\[
D_{KL} [q_\phi(z|x) || p_\theta(z|x)] = \sum_{z \sim q_\phi(z|x)} q_\phi(z|x) \log \frac{q_\phi(z|x)}{p_\theta(z|x)}, \tag{2.9}
\]

This can be seen as an expectation,

\[
D_{KL} [q_\phi(z|x) || p_\theta(z|x)] = E_{z \sim q_\phi(z|x)} \left[ \log \frac{q_\phi(z|x)}{p_\theta(z|x)} \right] = E_{z \sim q_\phi(z|x)} \left[ \log q_\phi(z|x) - \log p_\theta(z|x) \right]. \tag{2.10}
\]

The second term above, i.e., the true posterior, according to the Bayes’ theorem, can be written as \( p_\theta(z|x) = \frac{p(x|z)p(z)}{p(x)} \). The data distribution \( p_\theta(x) \) is independent of the latent variable \( z \), thus, can be pulled out of the expectation term,

\[
D_{KL} [q_\phi(z|x) || p_\theta(z|x)] = E_{z \sim q_\phi(z|x)} \left[ \log q_\phi(z|x) - \log p_\theta(x|z) - \log p(z) \right] + \log p_\theta(x). \tag{2.11}
\]

By putting the \( \log p_\theta(x) \) in the right hand side of the above equation, we can write,

\[
D_{KL} [q_\phi(z|x) || p_\theta(z|x)] - \log p_\theta(x) = E_{z \sim q_\phi(z|x)} \left[ \log q_\phi(z|x) - \log p_\theta(x|z) - \log p(z) \right]
\]

\[
\log p_\theta(x) - D_{KL} [q_\phi(z|x) || p_\theta(z|x)] = E_{z \sim q_\phi(z|x)} \left[ \log p_\theta(x|z) - (\log q_\phi(z|x) - \log p(z)) \right]
\]

\[
= E_{z \sim q_\phi(z|x)} \left[ \log p_\theta(x|z) \right] - E_{z \sim q_\phi(z|x)} \left[ \log q_\phi(z|x) - \log p(z) \right]. \tag{2.12}
\]

The second expectation term in above equation, according to definitions, is the KL divergence between the approximate posterior \( q_\phi(z|x) \) and the prior \( \log p(z) \) distributions. Thus, this can be written as

\[
\log p_\theta(x) - D_{KL} [q_\phi(z|x) || p_\theta(z|x)] = E_{z \sim q_\phi(z|x)} \left[ \log p_\theta(x|z) \right] - D_{KL} [q_\phi(z|x) || p(z)]. \tag{2.13}
\]

In above equation, \( \log p_\theta(x) \) is the log-likelihood of the data which we would like to optimize, \( D_{KL} [q_\phi(z|x) || p_\theta(z|x)] \) is the KL divergence between the approximate and the true posterior distributions, which is not tractable to compute, but, according to definitions, we know that is non-negative, \( E_{z \sim q_\phi(z|x)} \left[ \log p_\theta(x|z) \right] \) is the reconstruction loss, and \( D_{KL} [q_\phi(z|x) || p(z)] \) is the KL divergence between the approximate posterior distribution and a prior over the latent variable. The last term can be seen as a regularization term over the latent representation. Therefore, the intractability and non-negativity of \( D_{KL} [q_\phi(z|x) || p_\theta(z|x)] \) only allows us to
Technical Background

Figure 2.4: In a standard VAE, the model learns stochastic mappings between the observed dataset whose empirical distribution is $p_D(x)$, typically a complicated one, and a latent space whose distribution is relatively simpler, e.g., $\mathcal{N}(0, I)$.

Optimize the lower bound on the log-likelihood of the data,

$$\log p_\theta(x) \geq \mathbb{E}_{z \sim q_\phi(z|x)} \left[ \log p_\theta(x|z) \right] - D_{KL} [q_\phi(z|x)||p(z)],$$  \hspace{1cm} (2.14)

which we call variational or evidence lower bound (ELBO).

In practice, VAEs first learn to generate a latent variable $z$ given the data $x$, i.e., approximate the posterior distribution $q_\phi(z|x)$, the encoder, whose goal is to model the variation of the data. From this latent random variable $z$, VAEs then generate a sample $x$ by learning $p_\theta(x|z)$, the decoder, whose goal is to maximize the log likelihood of the data.

These two networks, i.e., the encoder ($q_\phi(z|x)$) and the decoder ($p_\theta(x|z)$), are trained jointly, using a prior over the latent variable, as mentioned above. This prior is usually the standard Normal distribution, $\mathcal{N}(0, I)$. In practice, the posterior distribution is approximated by a Gaussian $\mathcal{N}(\mu, \sigma^2 I)$, whose parameters are output by the encoder. To facilitate optimization, the reparameterization trick [Kingma and Welling, 2013] is used. That is, the latent variable is computed as $z = \mu + \sigma \odot \epsilon$, where $\epsilon$ is a vector sampled from the standard Normal distribution.

As an extension to VAEs, conditional VAEs, or CVAEs [Sohn et al., 2015] in short, use auxiliary information, i.e., the conditioning variable or observation, to generate the data $x$. In the standard setting, both the encoder and the decoder are conditioned on the conditioning variable $c$. That is, the encoder is denoted as $q_\phi(z|x, c)$ and the decoder as $p_\theta(x|z, c)$. Then, the objective of the model becomes

$$\log p_\theta(x|c) \geq \mathbb{E}_{z \sim q_\phi(z|x)} \left[ \log p_\theta(x|z, c) \right] - D_{KL} \left[ q_\phi(z|x, c) || p(z|c) \right],$$  \hspace{1cm} (2.15)

In practice, conditioning is typically done by concatenation; the input of the encoder is the concatenation of the data $x$ and the condition $c$, i.e., $q_\phi(z|x, c)$, and that of the decoder the concatenation of the latent variable $z$ and the condition $c$, i.e., $p_\theta(x|z, c)$. Thus, the prior distribution is still $p(z)$, and the latent variable is sampled independently of the conditioning one. It is then left to the decoder to combine the information from the latent and conditioning...
variables to generate a data sample.
General Action Anticipation

In this chapter, we focus on action anticipation, the task of predicting one discrete representation, i.e., action label, of a deterministic future.

In contrast to the widely studied problem of recognizing an action given a complete sequence, action anticipation aims to identify the action from only partially available videos. As such, it is therefore key to the success of computer vision applications requiring to react as early as possible, such as autonomous navigation. In this chapter, we propose a new action anticipation method that achieves high prediction accuracy even in the presence of a very small percentage of a video sequence. To this end, we develop a multi-stage LSTM architecture that leverages context-aware and action-aware features, and introduce a novel loss function that encourages the model to predict the correct class as early as possible. Our experiments on standard benchmark datasets evidence the benefits of our approach; We outperform the state-of-the-art action anticipation methods for early prediction by a relative increase in accuracy of 22.0% on JHMDB-21, 14.0% on UT-Interaction and 49.9% on UCF-101.

3.1 Introduction

Understanding actions from videos is key to the success of many real-world applications, such as autonomous navigation and sports analysis. While great progress has been made to recognize actions from complete sequences [Donahue et al., 2015b; Wang et al., 2016b; Fernando et al., 2016a; Bilen et al., 2016a; Fernando et al., 2016c; Wang and Schmid, 2013a] in the past decade, action anticipation [Ryoo, 2011; Ma et al., 2016; Soomro et al., 2016b], which aims to predict the observed action as early as possible, has become a popular research problem only recently. Anticipation is crucial in scenarios where one needs to react before the action is finalized, such as to avoid hitting pedestrians with an autonomous car, or to forecast dangerous situations in surveillance scenarios.

The key difference between recognition and anticipation lies in the fact that the methods tackling the latter should predict the correct class as early as possible, given only a few frames from the beginning of the video sequence. To address this, several approaches have introduced new training losses encouraging the score [Soomro et al., 2016a] or the rank [Ma et al., 2016] of the correct action to increase with time, or penalizing increasingly strongly the classification mistakes [Jain et al., 2016b]. In practice, however, the effectiveness of these losses remains limited for very early prediction, such as from 1% of the sequence.
In this chapter, we introduce a novel loss that encourages making correct predictions very early. Specifically, our loss models the intuition that some actions, such as running and high jump, are highly ambiguous after seeing only the first few frames, and false positives should therefore not be penalized too strongly in the early stages. By contrast, we would like to predict a high probability for the correct class as early as possible, and thus penalize false negatives from the beginning of the sequence. Our experiments demonstrate that, for a given model, our new loss yields significantly higher accuracy than existing ones on the task of early prediction.

In particular, in this chapter, we also contribute a novel multi-stage Long Short Term Memory (LSTM) architecture for action anticipation. This model effectively extracts and jointly exploits context- and action-aware features (see Fig. 3.1). This is in contrast to existing methods that typically extract either global representations for the entire image [Diba et al., 2016a; Wang et al., 2016b] or video sequence [Tran et al., 2015a; Karpathy et al., 2014a], thus not focusing on the action itself, or localize the feature extraction process to the action itself via dense trajectories [Wang et al., 2015; Wang and Schmid, 2013b; Fernando et al., 2016b], optical flow [Feichtenhofer et al., 2016b; Wang et al., 2016c; Li et al., 2016] or actionness [Wang et al., 2016a; Chen et al., 2014; Jain et al., 2014; Yu and Yuan, 2015; Van den Bergh et al., 2013; Oneata et al., 2014], thus failing to exploit contextual information.

To the best of our knowledge, only two-stream networks [Simonyan and Zisserman, 2014b; Feichtenhofer et al., 2016b; Chéron et al., 2015; Liu et al., 2016] have attempted to jointly leverage both information types by making use of RGB frames in conjunction with optical flow to localize the action. Exploiting optical flow, however, does not allow these methods to explicitly leverage appearance in the localization process. Furthermore, computing optical flow is typically expensive, thus significantly increasing the runtime of these methods. By not relying on optical flow, our method is significantly more efficient: On a single GPU, our model
analyzes a short video (e.g., 50 frames) 14 times faster than [Simonyan and Zisserman, 2014b] and [Feichtenhofer et al., 2016b].

Our model is depicted in Fig. 3.2. In a first stage, it focuses on the global, context-aware information by extracting features from the entire RGB image. The second stage then combines these context-aware features with action-aware ones obtained by exploiting class-specific activations, typically corresponding to regions where the action occurs. In short, our model first extracts the contextual information, and then merges it with the localized one.

As evidenced by our experiments, our approach significantly outperforms the state-of-the-art action anticipation methods on all the standard benchmark datasets that we evaluated on, including UCF-101 [Soomro et al., 2012a], UT-Interaction [Ryoo et al., 2010], and JH-MDB21 [Jhuang et al., 2013a]. We further show that our combination of context- and action-aware features is also beneficial for the more traditional task of action recognition. Moreover, we evaluate the effect of optical flow features for both action recognition and anticipation.

3.2 Related Work

The focus of this chapter is twofold: Action anticipation, with a new loss that encourages correct prediction as early as possible, and action modeling, with a model that combines context- and action-aware information using multi-stage LSTMs. Below, we discuss the most relevant approaches for these two aspects.

3.2.1 Action Anticipation

The idea of action anticipation was introduced by [Ryoo and Aggarwal, 2009], which models causal relationships to predict human activities. This was followed by several attempts to model the dynamics of the observed actions, such as by introducing integral and dynamic bag-of-words [Ryoo, 2011], using spatial-temporal implicit shape models [Yu et al., 2012], extracting human body movements via skeleton information [Zhao et al., 2013], and accounting for the complete and partial history of observed features [Kong et al., 2014].

More recently, [Soomro et al., 2016b] proposed to make use of binary SVMs to classify video snippets into sub-action categories and obtain the final class label in an online manner using dynamic programming. To overcome the need to train one classifier per sub-action, [Soomro et al., 2016a] extended this approach to using a structural SVM. Importantly, this work further introduced a new objective function to encourage the score of the correct action to increase as time progresses.

While the above-mentioned work made use of handcrafted features, recent advances have naturally led to the development of deep learning approaches to action anticipation. In this context, [Ma et al., 2016] proposed to combine a Convolutional Neural Network (CNN) with an LSTM to model both spatial and temporal information. The authors further introduced new ranking losses whose goal is to enforce either the score of the correct class or the margin between the score of the correct class and that of the best score to be non-decreasing over time. Similarly, in [Jain et al., 2016c], a new loss that penalizes classification mistakes increasingly strongly over time was introduced in an LSTM-based framework that used multiple modalities. While the two above-mentioned methods indeed aim at improving classification accuracy over
We propose to extract context-aware features, encoding global information about the scene, and combine them with action-aware ones, which focus on the action itself. To this end, we introduce a multi-stage LSTM architecture that leverages the two types of features to predict the action or forecast it. Note that, for the sake of visualization, the color maps were obtained from 3D tensors \((512 \times W \times H)\) via an average pooling operation over the 512 channels.

time, they do not explicitly encourage making correct predictions as early as possible. By contrast, while accounting for ambiguities in early stages, our new loss still aims to prevent false negatives from the beginning of the sequence.

Instead of predicting the future class label, in [Vondrick et al., 2016], the authors proposed to predict the future visual representation. However, the main motivation for this was to work with unlabeled videos, and the learned representation is therefore not always related to the action itself.

### 3.2.2 Action Modeling

Most recent action approaches extract global representations for the entire image [Diba et al., 2016b] [Wang et al., 2016d] [Donahue et al., 2015a] or video sequence [Tran et al., 2015b] [Karpathy et al., 2014b]. As such, these methods do not truly focus on the actions of interest, but rather compute a context-aware representation. Unfortunately, context does not always bring reliable information about the action. For example, one can play guitar in a bedroom, a concert hall or a yard. To overcome this, some methods localize the feature extraction process by exploiting dense trajectories [Wang et al., 2015] [Wang and Schmid, 2013b] [Fernando et al., 2016b] or optical flow [Li et al., 2016]. Inspired by objectness, the notion of actionness [Wang et al., 2016a; Chen et al., 2014; Jain et al., 2014; Yu and Yuan, 2015; Van den Bergh et al., 2013; Oneata et al., 2014] has recently also been proposed to localize the regions where a generic action occurs. The resulting methods can then be thought of as extracting action-aware representations. In other words, these methods go to the other extreme and completely discard the notion of context which can be useful for some actions, such as playing football on a grass field.

There is nevertheless a third class of methods that aim to leverage these two types of information [Simonyan and Zisserman, 2014b] [Feichtenhofer et al., 2016b] [Chéron et al., 2015; Liu et al., 2016] [Wang et al., 2016c]. By combining RGB frames and optical flow in two-
stream architectures, these methods truly exploit context and motion, from which the action can be localized by learning to distinguish relevant motion. This localization, however, does not directly exploit appearance. Here, inspired by the success of these methods, we develop a novel multi-stage network that also leverages context- and action-aware information. However, we introduce a new action-aware representation that exploits the RGB data to localize the action. As such, our approach effectively leverages appearance for action-aware modeling, and, by avoiding the expensive optical flow computation, is much more efficient than the above-mentioned two-stream models. In particular, our model is about 14 times faster than the state-of-the-art two-stream network of [Feichtenhofer et al., 2016b] and has less parameters. The reduction in number of parameters is due to the fact that [Feichtenhofer et al., 2016b] and [Simonyan and Zisserman, 2014b] rely on two VGG-like networks (one for each stream) with a few additional layers (including 3D convolutions for [8]). By contrast, our model has, in essence, a single VGG-like architecture, with some additional LSTM layers, which only have few parameters. Moreover, our work constitutes the first attempt at explicitly leveraging context- and action-aware information for action anticipation. Finally, we introduce a novel multi-stage LSTM fusion strategy to integrate action and context aware features.

3.3 Our Approach

Our goal is to predict the class of an action as early as possible, that is, after having seen only a very small portion of the video sequence. To this end, we first introduce a new loss function that encourages making correct predictions very early. We then develop a multi-stage LSTM model that makes use of this loss function while leveraging both context- and action-aware information.

3.3.1 A New Loss for Action Anticipation

As argued above, a loss for action anticipation should encourage having a high score for the correct class as early as possible. However, it should also account for the fact that, early in the sequence, there can often be ambiguities between several actions, such as running and high jump. Below, we introduce a new anticipation loss that follows these two intuitions.

Specifically, let \( y^t(k) \) encode the true activity label at time \( t \), i.e., \( y^t(k) = 1 \) if the sample belongs to class \( k \) and 0 otherwise, and \( \hat{y}^t(k) \) denote the corresponding label predicted by a given model. We define our new loss as

\[
\mathcal{L}(y, \hat{y}) = -\frac{1}{N} \sum_{k=1}^{N} \sum_{t=1}^{T} \left[ y^t(k) \log(\hat{y}^t(k)) + \frac{t(1 - y^t(k))}{T} \log(1 - \hat{y}^t(k)) \right],
\]

where \( N \) is the number of action classes and \( T \) the length (number of frames) of the input sequence.

This loss function consists of two terms. The first one penalizes false negatives with the same strength at any point in time. By contrast, the second one focuses on false positives, and its strength increases linearly over time, to reach the same weight as that on false negatives. Therefore, the relative weight of the first term compared to the second one is larger at the
beginning of the sequence. Altogether, this encourages predicting a high score for the correct
class as early as possible, i.e., preventing false negatives, while accounting for the potential
ambiguities at the beginning of the sequence, which give rise to false positives. As we see
more frames, however, the ambiguities are removed, and these false positives are encouraged
to disappear.

Our new loss matches the desirable properties of an action anticipation loss. In the next
section, we introduce a novel multi-stage architecture that makes use of this loss.

3.3.2 Multi-stage LSTM Architecture

To tackle action anticipation, we develop the novel multi-stage recurrent architecture based
on LSTMs depicted by Fig. 3.2. This architecture consists of a stage-wise combination of
context- and action-aware information. Below, we first discuss our approach to extracting
these two types of information, and then present our complete multi-stage recurrent network.

3.3.2.1 Context- and Action-aware Modeling

To model the context- and action-aware information, we introduce the two-stream architec-
ture shown in Fig. 3.3. The first part of this network is shared by both streams and, up to
conv5-2, corresponds to the VGG-16 network [Simonyan and Zisserman, 2014d], pre-trained
on ImageNet for object recognition. The output of this layer is connected to two sub-models:
One for context-aware features and the other for action-aware ones. We then train these two
sub-models for the same task of action recognition from a single image using a cross-entropy
loss function defined on the output of each stream. In practice, we found that training the en-
tire model in an end-to-end manner did not yield a significant improvement over training only
the two sub-models. In our experiments, we therefore opted for this latter strategy, which is
less expensive computationally and memory-wise. Below, we first discuss the context-aware
sub-network and then turn to the action-aware one.

Context-Aware Feature Extraction. This sub-model is similar to VGG-16 from conv5-3 up to
the last fully-connected layer, with the number of units in the last fully-connected
3.3 Our Approach

layer changed from 1000 (original 1000-way ImageNet classification model) to the number of activities $N$.

In essence, this sub-model focuses on extracting a deep representation of the whole scene for each activity and thus incorporates context. We then take the output of its fc7 layer as our context-aware features.

**Action-Aware Feature Extraction.** As mentioned before, the context of an action does not always correlate with the action itself. Our second sub-model therefore aims at extracting features that focus on the action itself. To this end, we draw inspiration from the object classification work of [Zhou et al., 2016b]. At the core of this work lies the idea of Class Activation Maps (CAMs). In our context, a CAM indicates the regions in the input image that contribute most to predicting each class label. In other words, it provides information about the location of an action. Importantly, this is achieved without requiring any additional annotations.

More specifically, CAMs are extracted from the activations in the last convolutional layer in the following manner. Let $f_l(x, y)$ represent the activation of unit $l$ in the last convolutional layer at spatial location $(x, y)$. A score $S_k$ for each class $k$ can be obtained by performing global average pooling [Lin et al., 2013] to obtain, for each unit $l$, a feature $F_l = \sum_{x,y} f_l(x, y)$, followed by a linear layer with weights $\{w_{kl}\}$. That is, $S_k = \sum_l w_{kl} F_l$. A CAM for class $k$ at location $(x, y)$ can then be computed as

$$ M_k(x, y) = \sum_l w_{kl} f_l(x, y) , $$

which indicates the importance of the activations at location $(x, y)$ in the final score for class $k$.

Here, we propose to make use of the CAMs to extract action-aware features. To this end, we use the CAMs in conjunction with the output of the conv5-3 layer of the model. The intuition behind this is that conv5-3 extracts high-level features that provide a very rich representation of the image [Zeiler and Fergus, 2014] and typically correspond to the most discriminative parts of the object [Bertasius et al., 2015b; Saleh et al., 2018], or, in our case, the action. Therefore, we incorporate a new layer to our sub-model, whose output can be expressed as

$$ A_k(x, y) = \text{conv}_{5-3}(x, y) \times \text{ReLU}(M_k(x, y)) , $$

where $\text{ReLU}(M_k(x, y)) = \max(0, M_k(x, y))$. As shown in Fig. 3.4, this new layer is then followed by fully-connected ones, and we take our action-aware features as the output of the corresponding fc7 layer.

3.3.2.2 Sequence Learning for Action Anticipation

To effectively combine the information contained in the context-aware and action-aware features described above, we design the novel multi-stage LSTM model depicted by Fig. 3.2. This model first focuses on the context-aware features, which encode global information about the entire image. It then combines the output of this first stage with our action-aware features to provide a refined class prediction.
Given the fine-tuned feature extraction network, we introduce a new layer that alters the output of conv5-3. This lets us filter out the conv5-3 features that are irrelevant, to focus on the action itself. Our action-aware features are taken as the output of the last fully-connected layer.

To train this model for action anticipation, we make use of our new loss introduced in Section 3.3.1. Therefore, ultimately, our network models long-range temporal information and yields increasingly accurate predictions as it processes more frames.

Specifically, we write the overall loss of our model as

$$L_o = \frac{1}{V} \sum_{i=1}^{V} (L_{c,i} + L_{a,i}),$$

(3.4)

where $V$ is the total number of training sequences. This loss function combines losses corresponding to the context-aware stage and to the action-aware stage, respectively. Below, we discuss these two stages in more detail.

**Learning Context.** The first stage of our model takes as input our context-aware features, and passes them through a layer of LSTM cells followed by a fully-connected layer that, via a softmax operation, outputs a probability for each action class. Let $\hat{y}_{c,i}$ be the vector of probabilities for all classes and all time steps predicted by the first stage for sample $i$. We then define the loss for a single sample as

$$L_{c,i} = \mathcal{L}(y_i, \hat{y}_{c,i}),$$

(3.5)

where $\mathcal{L}(\cdot)$ is our new loss defined in Eq. 4.1 and $y_i$ is the ground-truth class label for sample $i$.

**Learning Context and Action.** The second stage of our model aims at combining context-aware and action-aware information. Its structure is the same as that of the first stage, i.e., a layer of LSTM cells followed by a fully-connected layer to output class probabilities via a softmax operation. However, its input merges the output of the first stage with our action-aware features. This is achieved by concatenating the hidden activations of the LSTM layer with our action-aware features. We then make use of the same loss function as before, but defined on the final prediction. For sample $i$, this can be expressed as

$$L_{a,i} = \mathcal{L}(y_i, \hat{y}_{a,i}),$$

(3.6)

where $\hat{y}_a$ is the vector of probabilities for all classes predicted by the second stage.
Inference. At inference, the input RGB frames are forward-propagated through our model. We therefore obtain a probability vector for each class at each frame. While one could simply take the probabilities in the current frame $t$ to obtain the class label at time $t$, via $argmax$, we propose to increase robustness by leveraging the predictions of all the frames up to time $t$. To this end, we make use of an average pooling of these predictions over time.

### 3.4 Experiments

In this section, we first compare our method with state-of-the-art techniques on the task of action anticipation, and then analyze various aspects of our model, such as the influence of the loss function and of the different feature types. In the supplementary material, we provide additional experiments to analyze the effectiveness of different LSTM architectures, and the influence of the number of hidden units and of our temporal average pooling strategy. We also report the performance of our method on the task of action recognition from complete videos with and without optical flow, and action anticipation with optical flow.

#### 3.4.1 Datasets

For our experiments, we made use of the standard UCF-101 [Soomro et al., 2012a], UT-Interaction [Ryoo et al., 2010], and JHMDB-21 [Jhuang et al., 2013a] benchmarks, which we briefly describe below.

The UCF-101 dataset consists of 13,320 videos (each contains a single action) of 101 action classes including a broad set of activities such as sports, playing musical instruments and human-object interaction, with an average length of 7.2 seconds. UCF-101 is one of the most challenging datasets due to its large diversity in terms of actions and to the presence of large variations in camera motion, cluttered background and illumination conditions. There are three standard training/test splits for this dataset. In our comparisons to the state-of-the-art for both action anticipation and recognition, we report the average accuracy over the three splits. For the detailed analysis of our model, however, we rely on the first split only.

The JHMDB-21 dataset is another challenging dataset of realistic videos from various sources, such as movies and web videos, containing 928 videos and 21 action classes. Similarly to UCF-101, in our comparison to the state-of-the-art, we report the average accuracy over the three standard splits of data. Similar to UCF-101 dataset, each video contains one action starting from the beginning of the video.

The UT-Interaction dataset contains videos of continuous executions of 6 human-human interaction classes: shake-hands, point, hug, push, kick and punch. The dataset contains 20 video sequences whose length is about 1 minute each. Each video contains at least one execution of each interaction type, providing us with 8 executions of human activities per video on average. Following the recommended experimental setup, we used 10-fold leave-one-out cross validation for each of the standard two sets of 10 videos. That is, within each set, we leave one sequence for testing and use the remaining 9 for training. Following standard practice, we also made use of the annotations provided with the dataset to split each video into sequences containing individual actions.
3.4.2 Implementation Details

**CNN and LSTM Configuration.** The parameters of the CNN were optimized using stochastic gradient descent with a fixed learning rate of 0.001, a momentum of 0.9, a weight decay of 0.0005, and mini-batches of size 32. To train our LSTMs, we similarly used stochastic gradient descent with a fixed learning rate of 0.001, a momentum of 0.9, and mini-batch size of 32. For all LSTMs, we used 2048 hidden units. To implement our method, we used Python and Keras [Chollet, 2015]. We will make our code publicly available.

**Training Procedure.** To fine-tune the network on each dataset, we augment the data, so as to reduce the effect of over-fitting. The input images were randomly flipped horizontally and rotated by a random amount in the range -8 to 8 degrees. We then extracted crops according to the following procedure: (1) Compute the maximum cropping rectangle with a given aspect ratio (320/240) that fits inside the input image. (2) Scale the width and height of the cropping rectangle by a factor randomly selected in the range 0.8-1. (3) Select a random location for the cropping rectangle within the original input image and extract the corresponding subimage. (4) Scale the subimage to 224 × 224.

After these geometric transformations, we further applied RGB channel shifting [Wu et al., 2015], followed by randomly adjusting image brightness, contrast and saturation with a factor $\alpha = 0.3$. The operations are: for brightness, $\alpha \times \text{Image}$, for contrast, $\text{Image} \times \alpha + (1.0 - \alpha) \times \text{mean}(\text{grey}(\text{Image}))$, and for saturation, $\text{Image} \times \alpha + (1.0 - \alpha) \times \text{grey}(\text{Image})$.

3.4.3 Comparison to the State-of-the-Art

We compare our approach to the state-of-the-art action anticipation results reported on each of the three datasets discussed above. We further complement these state-of-the-art results with additional baselines that make use of our context-aware features with the loss of either [Ma et al., 2016] or [Jain et al., 2016c]. Note that a detailed comparison of different losses within our model is provided in Section 3.4.4.1.

Following standard practice, we report the so-called earliest and latest prediction accuracies. Note, however, that there is no real agreement on the proportion of frames that the earliest setting corresponds to. For each dataset, we make use of the proportion that has been employed by the baselines (i.e., either 20% or 50%). Note also that our approach relies on at most $T$ frames (with $T = 50$ in practice). Therefore, in the latest setting, where the baselines rely on the complete sequences, we only exploit the first $T$ frames. We believe that the fact that our method significantly outperforms the state-of-the-art in this setting despite using less information further evidences the effectiveness of our approach.

**JHMDB-21.** The results for the JHMDB-21 dataset are provided in Table 3.1. In this case, following the baselines, earliest prediction corresponds to observing the first 20% of the sequence. Note that we clearly outperform all the baselines by a significant margin in both the earliest and latest settings. Remarkably, we also outperform the methods that rely on additional information as input, such as optical flow [Soomro et al., 2016a,b; Ma et al., 2016] and Fisher vector features based on Improved Dense Trajectories [Soomro et al., 2016a]. This clearly demonstrates the benefits of our approach for anticipation.
Table 3.1: Comparison with state-of-the-art baselines on the task of action anticipation on the JHMDB-21 dataset. Note that our approach outperforms all baselines significantly in both settings.

| Method                               | Earliest | Latest |
|--------------------------------------|----------|--------|
| DP-SVM [Soomro et al., 2016a]        | 5%       | 46%    |
| S-SVM [Soomro et al., 2016a]         | 5%       | 43%    |
| Where/What [Soomro et al., 2016b]    | 10%      | 43%    |
| Ranking Loss [Ma et al., 2016]       | 29%      | 43%    |
| Context-Aware+Loss of [Jain et al., 2016c] | 28%    | 43%    |
| Context-Aware+Loss of [Ma et al., 2016] | 33%    | 39%    |
| Ours                                 | **55%**  | **58%**|

Table 3.2: Comparison with state-of-the-art baselines on the task of action anticipation on the UT-Interaction dataset.

| Method                               | Earliest | Latest |
|--------------------------------------|----------|--------|
| D-BoW [Ryoo, 2011]                   | 70.0%    | 85.0%  |
| I-BoW [Ryoo, 2011]                   | 65.0%    | 81.7%  |
| CuboidSVM [Ryoo et al., 2010]        | 31.7%    | 85.0%  |
| BP-SVM [Laviers et al., 2009]        | 65.0%    | 83.3%  |
| CuboidBayes [Ryoo, 2011]             | 25.0%    | 71.7%  |
| DP-SVM [Soomro et al., 2016a]        | 13.0%    | 14.6%  |
| S-SVM [Soomro et al., 2016a]         | 11.0%    | 13.4%  |
| Context-Aware+Loss of [Jain et al., 2016c] | 45.0%   | 65.0%  |
| Context-Aware+Loss of [Ma et al., 2016] | 48.0%   | 60.0%  |
| Ours                                 | **84.0%**| **90.0%**|

Table 3.3: Action Anticipation on the UCF-101 dataset.

| Method                               | Earliest | Latest |
|--------------------------------------|----------|--------|
| Context-Aware+Loss of [Jain et al., 2016c] | 30.6%   | 71.1%  |
| Context-Aware+Loss of [Ma et al., 2016] | 22.6%   | 73.1%  |
| Ours                                 | **80.5%**| **83.4%**|

**UT-Interaction.** We provide the results for the UT-Interaction dataset in Table 3.2. Here, following standard practice, 50% of the sequence was observed for earliest prediction, and the entire sequence for latest prediction. Recall that our approach uses at most $T = 50$ frames for prediction in both settings, while the average length of a complete sequence is around 120 frames. Therefore, as evidenced by the results, our approach yields significantly higher accuracy despite using considerably less data as input.

**UCF-101.** We finally compare our approach with our two baselines on the UCF-101 dataset. While this is not a standard benchmark for action anticipation, this experiment is motivated by the fact that this dataset is relatively large, has many classes, with similarity across different classes, and contains variations in video capture conditions. Altogether, this makes it a challenging dataset to anticipate actions, especially when only a small amount of data is available. The results on this dataset are provided in Table 3.3. Here, the earliest setting corresponds to
Figure 3.5: **Comparison of different losses for action anticipation on UCF-101.** We evaluate the accuracy of our model trained with different losses as a function of the number of frames observed. This plot clearly shows the superiority of our loss function.

using the first 2 frames of the sequences, which corresponds to around 1% of the data. Again, we clearly outperform the two baselines consisting of exploiting context-aware features with the loss of either [Ma et al., 2016] or [Jain et al., 2016c]. We believe that this further evidences the benefits of our approach, which leverages both context- and action-aware features with our new anticipation loss. A detailed evaluation of the influence of the different feature types and losses is provided in the next section.

### 3.4.4 Analysis

In this section, we provide a more detailed analysis of the influence of our loss function and of the different feature types on anticipation accuracy. Finally, we also provide a visualization of our different feature types, to illustrate their respective contributions.

#### 3.4.4.1 Influence of the Loss Function

Throughout the chapter, we have argued that our novel loss, introduced in Section 3.3.1, is better-suited to action anticipation than existing ones. To evaluate this, we trained several versions of our model with different losses. In particular, as already done in the comparison to the state-of-the-art above, we replaced our loss with the ranking loss of [Ma et al., 2016] (ranking loss on detection score) and the loss of [Jain et al., 2016c], but this time within our complete multi-stage model, with both context- and action-aware features.

Furthermore, we made use of the standard cross-entropy (CE) loss, which only accounts
Figure 3.6: **Influence of our average pooling strategy.** Our simple yet effective average pooling leverages the predictions of all the frames up to time $t$. As shown on the UCF-101 dataset, this increases anticipation performance, especially at very early stages.

for one activity label for each sequence (at time $T$). This loss can be expressed as

$$
\mathcal{L}_{CE} = \sum_{k=1}^{N} [y^T(k) \log(\hat{y}^T(k)) + (1 - y^T(k)) \log(1 - \hat{y}^T(k))].
$$

(3.7)

We then also modified the loss of [Jain et al., 2016c], which consists of an exponentially weighted softmax, with an exponentially weighted cross-entropy loss (ECE), written as

$$
\mathcal{L}_{ECE} = \sum_{t=1}^{T} e^{-(T-t)} \sum_{k=1}^{N} [y^t(k) \log(\hat{y}^t(k)) + (1 - y^t(k)) \log(1 - \hat{y}^t(k))].
$$

(3.8)

The main drawback of this loss comes from the fact that it does not strongly encourage the model to make correct predictions as early as possible. To address this issue, we also introduce a linearly growing loss (LGL), defined as

$$
\mathcal{L}_{LGL} = \sum_{t=1}^{T} -\frac{t}{T} \sum_{k=1}^{N} [y^t(k) \log(y^t(k)) + (1 - y^t(k)) \log(1 - \hat{y}^t(k))].
$$

(3.9)

While our new loss, introduced in Section 3.3.1, also makes use of a linearly-increasing term,
it corresponds to the false positives in our case, as opposed to the false negatives in the LGL. Since some actions are ambiguous in the first few frames, we find it more intuitive not to penalize false positives too strongly at the beginning of the sequence. This intuition is supported by our results below, which show that our loss yields better results than the LGL.

In Fig. 3.5, we report the accuracy of the corresponding models as a function of the number of observed frames on the UCF-101 dataset. Note that our new loss yields much higher accuracies than the other ones, particularly when only a few frames of the sequence are observed; With only 2 frames observed, our loss yields an accuracy similar to the other losses with 30–40 frames. With 30fps, this essentially means that we can predict the action 1 second earlier than other methods. The importance of this result is exemplified by research showing that a large proportion of vehicle accidents are due to mistakes/misinterpretations of the scene in the immediate time leading up to the crash [Jain et al., 2016c; Rueda-Domingo et al., 2004].

Moreover, in Fig. 3.6, we report the performance of the corresponding models as a function of the number of observed frames when using our average pooling strategy. Note that this strategy can generally be applied to any action anticipation method and, as shown by comparing Figs. 3.5 and 3.6, increases accuracy, especially at very early stages, which clearly demonstrates its effectiveness. Note that using it in conjunction with our loss still yields the best results by a significant margin.

3.4.4.2 Influence of the Features

We then evaluate the importance of the different feature types, context-aware and action-aware, on anticipation accuracy. To this end, we compare models trained using each feature type individually with our model that uses them jointly. For the models using a single feature type, we made use of a single LSTM to model temporal information. By contrast, our approach relies on a multi-stage LSTM, which we denote by MS-LSTM. Note that all models were trained using our new anticipation loss. The results of this experiment on the UCF-101 dataset are provided in Table 3.9. These results clearly evidence the importance of using both feature types, which consistently outperforms individual ones.

| Feature Type | Model      | Earliest K=1 | Latest K=50 |
|--------------|------------|---------------|--------------|
| Context-Aware| LSTM       | 62.80%        | 72.71%       |
| Action-Aware | LSTM       | 69.33%        | 77.86%       |
| Context+Action| MS-LSTM    | 80.5%         | 83.37%       |

Since we extract action-aware features, and not motion-aware ones, our approach will not be affected by irrelevant motion. The CNN that extracts these features learns to focus on the discriminative parts of the images, thus discarding the irrelevant information. To confirm this, we conducted an experiment on some classes of UCF-101 that contain irrelevant motions/multiple actors, such as Baseball pitch, Basketball, Cricket Shot and Ice dancing. The results of our action-aware and context-aware frameworks for these classes are: 66.1% vs. 58.2% for Baseball pitch, 83% vs. 76.4% for Basketball, 65.1% vs. 58% for Cricket Shot, and 92.3% vs. 91.7% for Ice dancing. This shows that our action-aware features can effectively discard
irrelevant motion/actors to focus on the relevant one(s).

### 3.4.4.3 Visualization

Finally, we provide a better intuition of the kind of information each of our feature types encode (see Fig. 3.7). This visualization was computed by average pooling over the 512 channels of Conv5-3 (of both the context-aware and action-aware sub-networks). As can be observed in the figure, our context-aware features have high activations on regions corresponding to any relevant object in the scene (context). By contrast, in our action-aware features, high activations correctly correspond to the focus of the action. Therefore, they can reasonably localize the parts of the frame that most strongly participate in the action happening in the video and reduce the noise coming from context.

![Figure 3.7: Visualization of action-aware and context-aware features on UCF-101. These samples are representative of the data.](image)
3.5 Comparison to State-of-the-Art Action Recognition Methods

We first compare the results of our approach to state-of-the-art methods on UCF-101, JHMDB-21 and UT-Interaction in terms of average accuracy over the standard training and testing partitions. In Table 3.5, we provide the results on the UCF-101 dataset. Here, for the comparison to be fair, we only report the results of the baselines that do not use any other information than the RGB image and the activity label (we refer the readers to the baselines’ papers and the survey [Herath et al., 2017] for more detail). In other words, while it has been shown that additional, handcrafted features, such as dense trajectories and optical flow, can help improve accuracy [Wang and Schmid, 2013b; Wang et al., 2015; Li et al., 2016; Simonyan and Zisserman, 2014b; Bilen et al., 2016b], our goal here is to truly evaluate the benefits of our method, not of these features. Note, however, that, as discussed in the next section of this supplementary material, our approach can still benefit from such features. As can be seen from the table, our approach outperforms all these RGB-based baselines. In Tables 3.6 and 3.7, we provide the results for JHMDB-21 and UT-Interaction. Again, we outperform all the baselines, even though, in this case, some of them rely on additional information such as optical flow [Gkioxari and Malik, 2015; Wang et al., 2016a; Soomro et al., 2016a,b; Ma et al., 2016] or IDT Fisher vector features [Soomro et al., 2016a]. We believe that these experiments show the effectiveness of our approach at tackling the action recognition problem.

Table 3.5: Comparison with state-of-the-art methods on UCF-101 (average accuracy over all training/testing splits). For the comparison to be fair, we focus on the baselines that, as us, only use the RGB frames as input.

| Method                      | Accuracy |
|-----------------------------|----------|
| Dynamic Image Network [Bilen et al., 2016b] | 70.0%    |
| Dynamic Image Network + Static RGB [Bilen et al., 2016b] | 76.9%    |
| Rank Pooling [Fernando et al., 2016b] | 72.2%    |
| DHR [Fernando et al., 2016b] | 78.8%    |
| Zhang et al. [Zhang et al., 2016] | 74.4%    |
| LSTM [Srivastava et al., 2015] | 74.5%    |
| LRCN [Donahue et al., 2015a] | 68.8%    |
| C3D [Tran et al., 2015b] | 82.3%    |
| Spatial Stream Net [Simonyan and Zisserman, 2014b] | 73.0%    |
| Deep Network [Karpathy et al., 2014b] | 65.4%    |
| ConvPool (Single frame) [Yue-Hei Ng et al., 2015] | 73.3%    |
| ConvPool (30 frames) [Yue-Hei Ng et al., 2015] | 80.8%    |
| ConvPool (120 frames) [Yue-Hei Ng et al., 2015] | 82.6%    |
| Ours | **83.3%**    |
| Diff. to State-of-the-Art | **0.7%**   |
Table 3.6: Comparison with state-of-the-art methods on JHMDB-21 (average accuracy over all training/testing splits). Note that while the methods of [Gkioxari and Malik, 2015; Wang et al., 2016a; Soomro et al., 2016a] use motion/optical flow information and [Soomro et al., 2016a] uses IDT Fisher vector features, our method yields better performance.

| Method                        | Accuracy |
|-------------------------------|----------|
| Where and What [Soomro et al., 2016b] | 43.8%    |
| DP-SVM [Soomro et al., 2016a] | 44.2%    |
| S-SVM [Soomro et al., 2016a] | 47.3%    |
| Spatial-CNN [Gkioxari and Malik, 2015] | 37.9%    |
| Motion-CNN [Gkioxari and Malik, 2015] | 45.7%    |
| Full Method [Gkioxari and Malik, 2015] | 53.3%    |
| Actionness-Spatial [Wang et al., 2016a] | 42.6%    |
| Actionness-Temporal [Wang et al., 2016a] | 54.8%    |
| Actionness-Full Method [Wang et al., 2016a] | 56.4%    |
| Ours                          | 58.3%    |
| Diff. to State-of-the-Art     | +1.9%    |

Table 3.7: Comparison with state-of-the-art methods on UT-Interaction (average accuracy over all training/testing splits). Note that while the methods of [Soomro et al., 2016a] uses motion/optical flow information and IDT Fisher vector features, our method yields better performance.

| Method                        | Accuracy |
|-------------------------------|----------|
| D-BoW [Ryoo, 2011]           | 85.0%    |
| I-BoW [Ryoo, 2011]           | 81.7%    |
| Cuboid SVM [Ryoo et al., 2010] | 85.0%    |
| BP-SVM [Laviers et al., 2009] | 83.3%    |
| Cuboid/Bayesian [Ryoo, 2011] | 71.7%    |
| DP-SVM [Soomro et al., 2016a] | 14.6%    |
| Yu et al. [Yu et al., 2010]  | 83.3%    |
| Yuan et al. [Yuan et al., 2010] | 78.2%    |
| Waltisberg et al. [Waltisberg et al., 2010] | 88.0%    |
| Ours                          | 90.0%    |
| Diff. to State-of-the-Art     | +2.0%    |

### 3.6 Exploiting Optical Flow

Note that our approach can also be extended into a two-stream architecture to benefit from optical flow information, as state-of-the-art action recognition methods do. In particular, to extract optical flow features, we made use of the pre-trained temporal network of [Simonyan and Zisserman, 2014b]. We then computed the CNN features from a stack of 20 optical flow frames (10 frames in the x-direction and 10 frames in the y-direction), from \( t - 10 \) to \( t \) at each time \( t \). As these features are potentially loosely related to the action (by focusing on motion), we merge them with the input to the second stage of our multi-stage LSTM. In Table 3.8, we compare the results of our modified approach with state-of-the-art methods that also exploit...
optical flow. Note that our two-stream approach yields accuracy comparable to the state-of-the-art.

Table 3.8: Comparison with the state-of-the-art approaches that use optical flow. For the comparison to be fair, we focus on the baselines that, as us, use RGB frames+optical flow as input.

| Method                                      | Accuracy |
|---------------------------------------------|----------|
| Spatio-temporal ConvNet [Karpathy et al., 2014b] | 65.4%    |
| LRCN + Optical Flow [Donahue et al., 2015a]   | 82.9%    |
| LSTM + Optical Flow [Srivastava et al., 2015]  | 84.3%    |
| Two-Stream Fusion [Feichtenhofer et al., 2016b] | 92.5%    |
| CNN features + Optical Flow [Simonyan and Zisserman, 2014b] | 73.9%    |
| ConvPool (30 frames) + OpticalFlow [Yue-Hei Ng et al., 2015] | 87.6%    |
| ConvPool (120 frames) + OpticalFlow [Yue-Hei Ng et al., 2015] | 88.2%    |
| VLAD3 + Optical Flow [Li et al., 2016]       | 84.1%    |
| Two-Stream ConvNet [Simonyan and Zisserman, 2014b] | 88.0%    |
| Two-Stream Conv.Pooling [Yue-Hei Ng et al., 2015] | 88.2%    |
| Two-Stream TSN [Wang et al., 2016c]          | 91.5%    |
| Ours + Optical Flow                          | 91.8%    |

We also conducted an experiment to evaluate the effectiveness of incorporating optical flow in our framework for action anticipation. To handle the case where less than 10 frames are used, we padded the frame stack with gray images (with values 127.5). Our flow-based approach achieved 86.8% for earliest and 91.8% for latest prediction on UCF-101, thus showing that, if runtime is not a concern, optical flow can indeed help increase the accuracy of our approach.

We further compare our approach with the two-stream network [Simonyan and Zisserman, 2014b], designed for action recognition, applied to the task of action anticipation. On UCF-101, this model achieved 83.2% for earliest and 88.6% for latest prediction, which our approach with optical flow clearly outperforms.

### 3.7 Effect of Different Feature Types

Here, we evaluate the importance of the different feature types, context-aware and action-aware, on recognition accuracy. To this end, we compare models trained using each feature type individually with our model that uses them jointly. For all models, we made use of LSTMs with 2048 units. Recall that our approach relies on a multi-stage LSTM, which we denote by MS-LSTM. The results of this experiment for different losses are reported in Table 3.9. These results clearly evidence the importance of using both feature types, which consistently outperforms using individual ones in all settings.

### 3.8 Robustness to the Number of Hidden Units

Based on our experiments, we found that for large datasets such as UCF-101, the 512 hidden units that some baselines use (e.g. [Donahue et al., 2015a; Srivastava et al., 2015]) do not
Table 3.9: Importance of the different feature types using different losses. Note that combining both types of features consistently outperforms using a single one. Note also that, for a given model, our new loss yields higher accuracies than the other ones.

| Feature          | Sequence Learning | Accuracy   |
|------------------|-------------------|------------|
| Context-Aware    | LSTM (CE)         | 72.38%     |
| Action-Aware     | LSTM (CE)         | 74.24%     |
| Context+Action   | MS-LSTM (CE)      | 78.93%     |
| Context-Aware    | LSTM (ECE)        | 72.41%     |
| Action-Aware     | LSTM (ECE)        | 77.20%     |
| Context+Action   | MS-LSTM (ECE)     | 80.38%     |
| Context-Aware    | LSTM (LGL)        | 72.58%     |
| Action-Aware     | LSTM (LGL)        | 77.63%     |
| Context+Action   | MS-LSTM (LGL)     | 81.27%     |
| Context-Aware    | LSTM (Ours)       | 72.71%     |
| Action-Aware     | LSTM (Ours)       | 77.86%     |
| Context+Action   | MS-LSTM (Ours)    | 83.37%     |

suffice to capture the complexity of the data. Therefore, to study the influence of the number of units in the LSTM, we evaluated different versions of our model with 1024 and 2048 hidden units (since 512 yields poor results and higher numbers, e.g., 4096, would require too much memory) and trained the model with 80% training data and validated on the remaining 20%. For a single LSTM, we found that using 2048 hidden units performs best. For our multi-stage LSTM, using 2048 hidden units also yields the best results. We also evaluated the importance of relying on average pooling in the LSTM. The results of these different versions of our MS-LSTM framework are provided in Table 3.10. This shows that, typically, more hidden units and average pooling can improve accuracy slightly.

Effect of the LSTM Architecture

Finally, we study the effectiveness of our multi-stage LSTM architecture at merging our two feature types. To this end, we compare the results of our MS-LSTM with the following baselines: A single-stage LSTM that takes as input the concatenation of our context-aware and action-aware features (Concatenation); The use of two parallel LSTMs whose outputs are merged by concatenation and then fed to a fully-connected layer (Parallel). A multi-stage LSTM where the two different feature-types are processed in the reverse order (Swapped), that is, the model processes the action-aware features first and, in a second stage, combines them with the context-aware ones; The results of this comparison are provided in Table 3.11. Note that both multi-stage LSTMs outperform the single-stage one and the two parallel LSTMs, thus indicating the importance of treating the two types of features sequentially. Interestingly, processing context-aware features first, as we propose, yields higher accuracy than considering...
Table 3.10: Influence of the number of hidden LSTM units and of our average pooling strategy in our multi-stage LSTM model. These experiments were conducted on the first splits of UCF-101 and JHMDB-21.

| Setup       | Average Pooling | Hidden Units | UCF-101   | JHMDB-21   |
|-------------|-----------------|--------------|-----------|------------|
| Ours (CE)   | wo/ 1024        | 77.26%       | 52.80%    |
| Ours (CE)   | wo/ 2048        | 78.09%       | 53.43%    |
| Ours (CE)   | w/ 2048         | 78.93%       | 54.30%    |
| Ours (ECE)  | wo/ 1024        | 79.10%       | 55.33%    |
| Ours (ECE)  | wo/ 2048        | 79.41%       | 56.12%    |
| Ours (ECE)  | w/ 2048         | 80.38%       | 57.05%    |
| Ours (LGL)  | wo/ 1024        | 79.76%       | 55.70%    |
| Ours (LGL)  | wo/ 2048        | 80.10%       | 56.83%    |
| Ours (LGL)  | w/ 2048         | 81.27%       | 57.70%    |
| Ours        | wo/ 1024        | 81.94%       | 56.24%    |
| Ours        | wo/ 2048        | 82.16%       | 57.92%    |
| Ours        | w/ 2048         | 83.37%       | 58.41%    |

The action-aware ones at the beginning. This matches our intuition that context-aware features carry global information about the image and will thus yield noisy results, which can then be refined by exploiting the action-aware features.

Table 3.11: Comparison of our multi-stage LSTM model with diverse fusion strategies. We report the results of simple concatenation of the context-aware and action-aware features, their use in two parallel LSTMs with late fusion, and swapping their order in our multi-stage LSTM, i.e., action-aware first, followed by context-aware. Note that multi-stage architectures yield better results, with the best ones achieved by using context first, followed by action, as proposed in this chapter.

| Feature Sequence | Order | Learning | Accuracy |
|------------------|-------|----------|----------|
| Concatenation LSTM | 77.16% |
| Parallel 2 Parallel LSTMs | 78.63% |
| Swapped MS-LSTM (Ours) | 78.80% |
| Ours MS-LSTM (Ours) | 83.37% |

Furthermore, we evaluate a CNN-only version of our approach, where we removed the LSTM, but kept our average pooling strategy to show the effect of our MS-LSTM architecture on top of the CNN. On UCF-101, this achieved 69.53% for earliest and 73.80% for latest prediction. This shows that, while this CNN-only framework yields reasonable predictions,
our complete approach with our multistage LSTM benefits from explicitly being trained on multiple frames, thus achieving significantly higher accuracy (80.5% and 83.4%, respectively). While the LSTM could in principle learn to perform average pooling, we believe that the lack of data prevents this from happening.

3.9 Conclusion

In this chapter, we have introduced a novel loss function to address very early action anticipation. Our loss encourages the model to make correct predictions as early as possible in the input sequence, thus making it particularly well-suited to action anticipation. Furthermore, we have introduced a new multi-stage LSTM model that effectively combines context-aware and action-aware features. Our experiments have evidenced the benefits of our new loss function over existing ones. Furthermore, they have shown the importance of exploiting both context- and action-aware information. Altogether, our approach significantly outperforms the state-of-the-art in action anticipation on all the datasets we applied it to. However, all of these datasets are mainly designed for action recognition task, predicting the action label given the full extent of the video. We argue that this is not very well-suited for the task of anticipation since most actions start from the beginning of the videos. A proper dataset for action anticipation should contain videos that actions start after a reasonable amount of observation, e.g., after 25% of the video length. In the next chapter, we focus on creating such dataset for the very challenging task of action anticipation in driving scenarios. We also extend the multi-stage LSTM from processing two modalities (action-aware and context-aware) to arbitrary number of modalities.
Chapter 4

Action Anticipation in Driving Scenarios

Following previous chapter, we continue focusing on a discrete and deterministic anticipation task. Motivated by the success of our approach discussed in previous chapter, we move towards a more challenging and crucial application, action anticipation in driving scenarios. In such scenarios, anticipation becomes critical since one, the driver of ego car, other car’s driver, pedestrian, or cyclist, needs to react before the action is finalized, for instance, where a car needs to, e.g., avoid hitting pedestrians and respect traffic lights. While solutions have been proposed to tackle subsets of the driving anticipation tasks, by making use of diverse, task-specific sensors, there is no single dataset or framework that addresses them all in a consistent manner. In this chapter, we therefore introduce a new, large-scale dataset, called VIENA\(^2\), covering 5 generic driving scenarios, with a total of 25 distinct action classes. It contains more than 15K full HD, 5s long videos acquired in various driving conditions, weathers, daytimes and environments, complemented with a common and realistic set of sensor measurements. This amounts to more than 2.25M frames, each annotated with an action label, corresponding to 600 samples per action class. We discuss our data acquisition strategy and the statistics of our dataset, and benchmark state-of-the-art action anticipation techniques, including a new multi-modal LSTM architecture with an effective loss function for action anticipation in driving scenarios.

4.1 Introduction

Understanding actions/events from videos is key to the success of many real-world applications, such as autonomous navigation, surveillance and sports analysis. While great progress has been made to recognize actions from complete sequences [Feichtenhofer et al., 2016a; Donahue et al., 2015a; Wang et al., 2016c; Bilen et al., 2016b], action anticipation, which aims to predict the observed action as early as possible, has only reached a much lesser degree of maturity [Sadegh Aliakbarian et al., 2017; Vondrick et al., 2016; Soomro et al., 2016b]. Nevertheless, anticipation is a crucial component in scenarios where a system needs to react quickly, such as in robotics [Koppula and Saxena, 2016], and automated driving [Jain et al., 2016a; Liebner et al., 2013; Li et al., 2017]. Its benefits have also been demonstrated in surveillance settings [Ramanathan et al., 2016; Wang and Ji, 2017].
In this chapter, we focus on the driving scenario. In this context, when consulting the main actors in the field, may they be from the computer vision community, the intelligent vehicle one or the automotive industry, the consensus is that predicting the intentions of a car’s own driver, for Advanced Driver Assistance Systems (ADAS), remains a challenging task for a computer, despite being relatively easy for a human [Dong et al., 2017; Olabiyi et al., 2017; Jain et al., 2016b; Rasouli et al., 2017]. Anticipation then becomes even more complex when one considers the maneuvers of other vehicles and pedestrians [Klingelschmitt et al., 2016; Zyner et al., 2017; Dong et al., 2017]. However, it is key to avoiding dangerous situations, and thus to the success of autonomous driving.

Over the years, the researchers in the field of anticipation for driving scenarios have focused on specific subproblems of this challenging task, such as lane change detection [Morris et al., 2011; Tawari et al., 2014], a car’s own driver’s intention [Ohn-Bar et al., 2014] or maneuver [Jain et al., 2015; 2016b; Olabiyi et al., 2017] recognition and pedestrian intention prediction [Rasouli et al., 2017; Pool et al., 2017; Li et al., 2017; Schulz and Stiefelhagen, 2015]. Furthermore, these different subproblems are typically addressed by making use of different kinds of sensors, without considering the fact that, in practice, the automotive industry might not be able/willing to incorporate all these different sensors to address all these different tasks.

In this chapter, we study the general problem of anticipation in driving scenarios, encompassing all the subproblems discussed above, and others, such as other drivers’ intention prediction, with a fixed, sensible set of sensors. To this end, we introduce the Virtual ENvironment for Action Analysis (VIENA2) dataset, covering the five different subproblems of predicting driver maneuvers, pedestrian intentions, front car intentions, traffic rule violations, and accidents. Altogether, these subproblems encompass a total of 25 distinct action classes. VIENA2 was acquired using the GTA V video game [Rockstar-Games, 2018a]. It contains more than 15K full HD, 5s long videos, corresponding to more than 600 samples per action class, acquired in various driving conditions, weathers, daytimes, and environments. This amounts to more than 2.25M frames, each annotated with an action label. These videos are complemented by basic vehicle dynamics measurements, and therefore reflect well the type of information that one could have access to in practice.

Below, we describe how VIENA2 was collected and compare its statistics and properties to existing datasets. We then benchmark state-of-the-art action anticipation algorithms on VIENA2, and introduce a new multi-modal, LSTM-based architecture, together with a new
anticipation loss, which outperforms existing approaches in our driving anticipation scenarios. Finally, we investigate the benefits of our synthetic data to address anticipation from real images. In short, our contributions are: (i) a large-scale action anticipation dataset for general driving scenarios; (ii) a multi-modal action anticipation architecture.

VIENA$^2$ is meant as an extensible dataset that will grow over time to include not only more data but also additional scenarios. Note that, for benchmarking purposes, however, we will clearly define training/test partitions. A similar strategy was followed by other datasets such as CityScapes, which contains a standard benchmark set but also a large amount of additional data. VIENA$^2$ is publicly available, together with our benchmark evaluation, our new architecture and our multi-domain training strategy.

4.2 VIENA$^2$

VIENA$^2$ is a large-scale dataset for action anticipation, and more generally action analysis, in driving scenarios. While it is generally acknowledged that anticipation is key to the success of automated driving, to the best of our knowledge, there is currently no dataset that covers a wide range of scenarios with a common, yet sensible set of sensors. Existing datasets focus on specific subproblems, such as driver maneuvers and pedestrian intentions [Rasouli et al., 2017; Pool et al., 2017; Kooij et al., 2014a], and make use of different kinds of sensors. Furthermore, with the exception of [Jain et al., 2016a], none of these datasets provide videos whose first few frames do not already show the action itself or the preparation of the action.

To create VIENA$^2$, we made use of the GTA V video game, whose publisher allows, under some conditions, for the non-commercial use of the footage [Rockstar-Games, 2018b]. Beyond the fact that, as shown in [Richter et al., 2017] via psychophysics experiments, GTA V provides realistic images that can be captured in varying weather and daytime conditions, it has the additional benefit of allowing us to cover crucial anticipation scenarios, such as accidents, for which real-world data would be virtually impossible to collect.

In this section, we first introduce the different scenarios covered by VIENA$^2$ and discuss the data collection process. We then study the statistics of VIENA$^2$ and compare it against existing datasets.

4.2.1 Scenarios and Data Collection

As illustrated in Fig. 4.2, VIENA$^2$ covers five generic driving scenarios. These scenarios are all human-centric, i.e., consider the intentions of humans, but three of them focus on the car’s own driver, while the other two relate to the environment (i.e., pedestrians and other cars). These scenarios are:
1. **Driver Maneuvers (DM).** This scenario covers the 6 most common maneuvers a driver performs while driving: Moving forward (FF), stopping (SS), turning (left (LL) and right (RR)) and changing lane (left (CL) and right (CR)). Anticipation of such maneuvers as early as possible is critical in an ADAS context to avoid dangerous situations.

2. **Traffic Rules (TR).** This scenario contains sequences depicting the car’s own driver either violating or respecting traffic rules, e.g., stopping at (SR) and passing (PR) a red light, driving in the (in)correct direction (WD,CD), and driving off-road (DO). Forecasting these actions is also crucial for ADAS.

3. **Accidents (AC).** In this scenario, we capture the most common real-world accident cases [Volvo 2017]: Accidents with other cars (AC), with pedestrians (AP), and with assets (AA), such as buildings, traffic signs, light poles and benches, as well as no accident (NA). Acquiring such data in the real world is virtually infeasible. Nevertheless, these actions are crucial to anticipate for ADAS and autonomous driving.

4. **Pedestrian Intentions (PI).** This scenario addresses the question of whether a pedestrian is going to cross the road (CR), or has stopped (SS) but does not want to cross, or is walking along the road (AS) (on the sidewalk). We also consider the case where no pedestrian is in the scene (NP). As acknowledged in the literature [Pool et al. 2017; Schulz and Stiefelhagen 2015; Rasouli et al. 2017], early understanding of pedestrians’ intentions is critical for automated driving.

5. **Front Car Intentions (FCI).** The last generic scenario of VIENA$^2$ aims at anticipating the maneuvers of the front car. This knowledge has a strong influence on the behavior to adopt to guarantee safety. The classes are same as the ones in Driver Maneuver scenario, but for the driver of the front car.

We also consider an additional scenario consisting of the same driver maneuvers as above but for heavy vehicles, i.e., trucks and buses. In all these scenarios, for the data to resemble a real driving experience, we made use of the equipment depicted in Fig. 4.1 consisting of a steering wheel with a set of buttons and a gear stick, as well as a set of pedals. We then captured images at 30 fps with a single virtual camera mounted on the vehicle and facing the road forward. Since the speed of the vehicle is displayed at a specific location in these images, we also extracted it using an OCR module [Smith 2007] (see supplementary material for more detail on data collection). Furthermore, we developed an application that records measurements from the steering wheel. In particular, it gives us access to the steering angle every 1 microsecond, which allowed us to obtain a value of the angle synchronized with each image. Our application also lets us obtain the ground-truth label of each video sequence by recording the driver input from the steering wheel buttons. This greatly facilitated our labeling task, compared to [Richter et al. 2017; 2016], which had to use a middleware to access the rendering commands from which the ground-truth labels could be extracted. Ultimately, VIENA$^2$ consists of video sequences with synchronized measurements of steering angles and speed, and corresponding action labels.

Altogether, VIENA$^2$ contains more than 15K full HD videos (with frame size of 1920 $\times$ 1280), corresponding to a total of more than 2.25M annotated frames. The detailed number
of videos for each class and the proportions of different weather and daytime conditions of VIENA$^2$ are provided in Fig. 4.3. Each video contains 150 frames captured at 30 frames-per-second depicting a single action from one scenario. The action occurs in the second half of the video (mostly around the 4 second mark), which makes VIENA$^2$ well-suited to research on action anticipation, where one typically needs to see what happens before the action starts.

Our goal is for VIENA$^2$ to be an extensible dataset. Therefore, by making our source code and toolbox for data collection and annotation publicly available, we aim to encourage the community to participate and grow VIENA$^2$. Furthermore, while VIENA$^2$ was mainly collected for the task of action anticipation in driving scenarios, as it contains full length videos, i.e., videos of a single drive of 30 minutes on average depicting multiple actions, it can also be used for the tasks of action recognition and temporal action localization.

4.2.2 Comparison to Other Datasets

The different scenarios and action classes of VIENA$^2$ make it compatible with existing datasets, thus potentially allowing one to use our synthetic data in conjunction with real images. For instance, the action labels in the Driver Maneuver scenario correspond to the ones in Brain4Cars [Jain et al., 2016a] and in the Toyota Action Dataset [Olabiyi et al., 2017]. Similarly, our last two scenarios dealing with heavy vehicles contain the same labels as in Brain4Cars [Jain et al., 2016a]. Moreover, the actions in the Pedestrian Intention scenario corresponds to those in [Kooij et al., 2014b]. Note, however, that, to the best of our knowledge, there is no other dataset covering our Traffic Rules and Front Car Intention scenarios, or containing data involving heavy vehicles. Similarly, there is no dataset that covers accidents involving a driver’s own car. In this respect, the most closely related dataset is DashCam [Chan et al., 2016], which depicts accidents of other cars. Furthermore, VIENA$^2$ covers a much larger diversity of environmental conditions, such as daytime variations (morning, noon, afternoon, night, midnight), weather variations (clear, sunny, cloudy, foggy, hazy, rainy, snowy), and location variations (city, suburbs, high-
Table 4.1: Statistics comparison with action recognition and anticipation datasets. A * indicates a dataset specialized to one scenario, e.g., driving, as opposed to generic.

| Recognition Dataset          | Samples/Class | classes | videos  |
|------------------------------|---------------|---------|---------|
| UCF-101 [Soomro et al., 2012b] | 150           | 101     | 13.3K   |
| HMDB/JHMDB [Kuehne et al., 2011] | 120           | 51/21   | 5.1K/928|
| UCF-Sport* [Rodriguez et al., 2008] | 30            | 10      | 150     |
| Charades [Sigurdsson et al., 2016] | 100           | 157     | 9.8K    |
| ActivityNe [Fabian Caba Heilbron and Niebles, 2015] | 144           | 200     | 15K     |
| Kinetics [Kay et al., 2017]  | 400           | 400     | 306K    |

| Anticipation Dataset         | Samples/Class | classes | videos  |
|------------------------------|---------------|---------|---------|
| UT-Interaction [Ryoo and Aggarwal, 2009] | 20            | 6       | 60      |
| Brain4Cars* [Jain et al., 2016a] | 140           | 6       | 700     |
| JAAD* [Rasouli et al., 2017]  | 86            | 4       | 346     |
| VIENA^2*                     | 600           | 25      | 15K     |

ways, industrial, woods), than existing public datasets. In the supplementary material, we provide examples of each of these different environmental conditions.

In addition to covering more scenarios and conditions than other driving anticipation datasets, VIENA^2 also contains more samples per class than existing action analysis datasets, both for recognition and anticipation. As shown in Table 4.1, with 600 samples per class, VIENA^2 outsizes (at least class-wise) the datasets that are considered *large* by the community. This is also the case for other synthetic datasets, such as VIPER [Richter et al., 2017], GTA5 [Richter et al., 2016], Virtual KITTI [Gaidon et al., 2016], and SYNTHIA [Ros et al., 2017], which, by targeting different problems, such as semantic segmentation for which annotations are more costly to obtain, remain limited in size. We acknowledge, however, that, since we target driving scenarios, our dataset cannot match in absolute size more general recognition datasets, such as Kinetics.

### 4.3 Benchmark Algorithms

In this section, we first discuss the state-of-the-art action analysis and anticipation methods that we used to benchmark our dataset. We then introduce a new multi-modal LSTM-based approach to action anticipation, and finally discuss how we model actions from our images and additional sensors.

#### 4.3.1 Baseline Methods

The idea of anticipation was introduced in the computer vision community almost a decade ago by [Ryoo and Aggarwal, 2009]. While the early methods [Ryoo, 2011; Soomro et al., 2016b] relied on handcrafted-features, they have now been superseded by end-to-end learning...
Benchmark Algorithms

methods [Ma et al., 2016; Jain et al., 2016a; Sadegh Aliakbarian et al., 2017], focusing on designing new losses better-suited to anticipation. In particular, the loss of our approach in Chapter 3 has proven highly effective, achieving state-of-the-art results on several standard benchmarks.

Despite the growing interest of the community in anticipation, action recognition still remains more thoroughly investigated. Since recognition algorithms can be converted to performing anticipation by making them predict a class label at every frame, we include the state-of-the-art recognition methods in our benchmark. Specifically, we evaluate the following baselines:

**Baseline 1: CNN+LSTMs.** The high performance of CNNs in image classification makes them a natural choice for video analysis, via some modifications. This was achieved in [Donahe et al., 2015a] by feeding the frame-wise features of a CNN to an LSTM model, and taking the output of the last time-step LSTM cell as prediction. For anticipation, we can then simply consider the prediction at each frame. We then use the temporal average pooling strategy introduced in Chapter 3, which has proven effective to increase the robustness of the predictor for action anticipation.

**Baseline 2: Two-Stream Networks.** Baseline 1 only relies on appearance, ignoring motion inherent to video (by motion, we mean explicit motion information as input, such as optical flow). Two-stream architectures, such as the one of [Feichtenhofer et al., 2016a], have achieved state-of-the-art performance by explicitly accounting for motion. In particular, this is achieved by taking a stack of 10 externally computed optical flow frames as input to the second stream. A prediction for each frame can be obtained by considering the 10 previous frames in the sequence for optical flow. We also make use of temporal average pooling of the predictions.

**Baseline 3: Multi-Stage LSTMs.** The Multi-Stage LSTM (MS-LSTM) discussed in Chapter 3 constitutes the state of the art in action anticipation. This model jointly exploits context- and action-aware features that are used in two successive LSTM stages. As mentioned above, the key to the success of MS-LSTM is its training loss function. This loss function can be expressed as

\[
\mathcal{L}(y, \hat{y}) = -\frac{1}{N} \sum_{k=1}^{N} \sum_{t=1}^{T} \left[ y'(k) \log(\hat{y}'(k)) + w(t)(1 - y'(k)) \log(1 - \hat{y}'(k)) \right],
\]

where \(y'(k)\) is the ground-truth label of sample \(k\) at frame \(t\), \(\hat{y}'(k)\) the corresponding prediction, and \(w(t) = \frac{t}{T}\). The first term encourages the model to predict the correct action at any time, while the second term accounts for ambiguities between different classes in the earlier part of the video.
Action Anticipation in Driving Scenarios

Our MM-LSTM architecture

Figure 4.4: (Left) Our Multi-Stage LSTM architecture. (Right) Visualization of our weighting function for the anticipation loss of Eq. 4.1.

4.3.2 A New Multi-Modal LSTM

While effective, MS-LSTM suffers from the fact that it was specifically designed to take two modalities as input, the order of which needs to be manually defined. As such, it does not naturally apply to our more general scenario, and must be actively modified, in what might be a sub-optimal manner, to evaluate it with our action descriptors. To overcome this, we therefore introduce a new multi-modal LSTM (MM-LSTM) architecture that generalizes the multi-stage architecture, introduced in Chapter 3 of this thesis, to an arbitrary number of modalities. Furthermore, our MM-LSTM also aims to learn the importance of each modality for the prediction.

Specifically, as illustrated in Fig. 4.4 for \( M = 4 \) modalities, at each time \( t \), the representations of the \( M \) input modalities are first passed individually into an LSTM with a single hidden layer. The activations of these \( M \) hidden layers are then concatenated into an \( M \times 1024 \) matrix \( D_t \), which acts as input to a time-distributed fully-connected layer (FC-Pool). This layer then combines the \( M \) modalities to form a single vector \( O_t \in \mathbb{R}^{1024} \). This representation is then passed through another LSTM whose output is concatenated with the original \( D_t \) via a skip connection. The resulting \( (M + 1) \times 1024 \) matrix is then compacted into a 1024D vector via another FC-Pool layer. The output of this FC-Pool layer constitutes the final representation and acts as input to the classification layer.

The reasoning behind this architecture is the following. The first FC-Pool layer can learn the importance of each modality. While its parameters are shared across time, the individual, modality-specific LSTMs can produce time-varying outputs, thus, together with the FC-Pool layer, providing the model with the flexibility to change the importance of each modality over time. In essence, this allows the model to learn the importance of the modalities dynamically. The second LSTM layer then models the temporal variations of the combined modalities. The skip connection and the second FC-Pool layer produce a final representation that can leverage both the individual, modality-specific representations and the learned combination of these features.

Learning. To train our model, we make use of the loss of Eq. 4.1. However, we modify the weights as \( w(t) = \frac{e^{(\alpha t - \beta)}}{1 + e^{(\alpha t - \beta)}} \), allowing the influence of the second term to vary nonlinearly. In practice, we set \( \alpha = 3 \) and \( \beta = 6 \), yielding the weight function of Fig. 4.4. These values were motivated by the study of [Pentland and Liu 1999], which shows that driving actions typically undergo the following progression: In a first stage, the driver is not aware of an
action or decides to take an action. In the next stage, the driver becomes aware of an action or decides to take one. In the last portion of the video, the action has started. In this portion of the video, we do not want to make a wrong prediction, thus penalizing false positives strongly. Generally speaking, our sigmoid-based strategy to define the weight reflects the fact that, in practice and in contrast with many academic datasets, such as UCF-101 [Soomro et al., 2012b] and JHMDB-21 [Huang et al., 2013b], actions do not start right at the beginning of a video sequence, but at any point in time, the goal being detecting them as early as possible.

During training, we rely on stage-wise supervision, by introducing an additional classification layer after the second LSTM block, as illustrated in Fig. 4.3. At test time, however, we remove this intermediate classifier to only keep the final one. We then make use of the temporal average pooling strategy introduced in Chapter 3 to accumulate the predictions over time.

4.3.3 Action Modeling

Our MM-LSTM can take as input multiple modalities that provide diverse and complementary information about the observed data. Here, we briefly describe the different descriptors that we use in practice.

- **Appearance-based Descriptors.** Given a frame at time $t$, the most natural source of information to predict the action is the appearance depicted in the image. To encode this information, we make use of a slightly modified DenseNet [Huang et al., 2016], pre-trained on ImageNet. See Section 4.3.4 for more detail. Note that we also use this DenseNet as appearance-based CNN for Baselines 1 and 2.

- **Motion-based Descriptors.** Motion has proven a useful cue for action recognition [Fichtenhofer et al., 2017, 2016a]. To encode this, we make use of a similar architecture as for our appearance-based descriptors, but modify it to take as input a stack of optical flows. Specifically, we extract optical flow between $L$ consecutive pairs of frames, in the range $[t - L, t]$, and form a $2L$ flow stack encoding horizontal and vertical flows. We fine-tune the model pre-trained on ImageNet for the task of action recognition, and take the output of the additional fully-connected layer as our motion-aware descriptor. Note that we also use this DenseNet for the motion-based stream of Baseline 2.

- **Vehicle Dynamics.** In our driving context, we have access to additional vehicle dynamics measurements. For each such measurement, at each time $t$, we compute a vector from its value $s_t$, its velocity $(s_t - s_{t-\delta})$ and its acceleration $(s_t - 2s_{t-\delta} + s_{t-2\delta})$. To map these vectors to a descriptor of size comparable to the appearance- and motion-based ones, inspired by [Fernando et al., 2017], we train an LSTM with a single hidden layer modeling the correspondence between vehicle dynamics and action label. In our dataset, we have two types of dynamics measurements, steering angle and speed, which results in two additional descriptors.

When evaluating the baselines, we report results of both their standard version, relying on the descriptors used in the respective papers, and of modified versions that incorporate the
four descriptor types discussed above. Specifically, for CNN-LSTM, we simply concatenate the vehicle dynamics descriptors and the motion-based descriptors to the appearance-based ones. For the Two-Stream baseline, we add a second two-stream sub-network for the vehicle dynamics and merge it with the appearance and motion streams by adding a fully-connected layer that takes as input the concatenation of the representation from the original two-stream sub-network and from vehicle dynamics two-stream sub-network. Finally, for MS-LSTM, we add a third stage that takes as input the concatenation of the second-stage representation with the vehicle dynamics descriptors.

4.3.4 Implementation Details

We make use of the DenseNet-121 [Huang et al., 2016], pre-trained on ImageNet, to extract our appearance- and motion-based descriptors. Specifically, we replace the classifier with a fully-connected layer with 1024 neurons followed by a classifier with $N$ outputs, where $N$ is the number of classes. We fine-tune the resulting model using stochastic gradient descent for 10 epochs with a fixed learning rate of 0.001 and mini-batches of size 16. Recall that, for the motion-based descriptors, the corresponding DenseNet relies on $2L$ flow stacks as input, which requires us to also replace the first layer of the network. To initialize the parameters of this layer, we average the weights over the three channels corresponding to the original RGB channels, and replicate these average weights $2L$ times [Wang et al., 2016c]. We found this scheme to perform better than random initialization.

4.4 Benchmark Evaluation and Analysis

We now report and analyze the results of our benchmarking experiments. For these experiments to be as extensive as possible given the available time, we performed them on a representative subset of VIENA $^2$ containing about 6.5K videos acquired in a large variety of environmental conditions and covering all 25 classes. This subset contains 277 samples per class, and thus still outsizes most action analysis datasets, as can be verified from Table 4.1. The detailed statistics of this subset are provided in the supplementary material.

To evaluate the behavior of the algorithms in different conditions, we defined three different partitions of the data. The first one, which we refer to as Random in our experiments, consists of randomly assigning 70% of the samples to the training set and the remaining 30% to the test set. The second partition considers the daytime of the sequences, and is therefore referred to as Daytime. In this case, the training set is formed by the day images and the test set by the night ones. The last partition, Weather, follows the same strategy but based on the information about weather conditions, i.e., a training set of clear weather and a test set of rainy/snowy/... weather.

Below, we first present the results of our benchmarking on the Random partition, and then analyze the challenges related to our new dataset. We finally evaluate the benefits of our synthetic data for anticipation from real images, and analyze the bias of VIENA $^2$. Note that additional results including benchmarking on the other partitions and ablation studies of our MM-LSTM model are provided in the supplementary material. Note also that the scenarios and classes acronyms are defined in Section 4.2.1.
4.4 Benchmark Evaluation and Analysis

Table 4.2: Results on the Random split of VIENA$^2$ for the original versions of our three baselines: CNN+LSTM [Donahue et al., 2015a] with only appearance, Two-Stream [Feichtenhofer et al., 2016a] with appearance and motion, and MS-LSTM (introduced in Chapter 3) with action-aware and context-aware features.

|                  | 1" | 2" | 3" | 4" | 5" | 1" | 2" | 3" | 4" | 5" | 1" | 2" | 3" | 4" | 5" |
|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| CNN+LSTM         |    |    |    |    |    | 23.3| 24.0 | 30.6 | 37.5 | 41.5 | 22.4| 28.1 | 37.5 | 42.6 | 44.0 |
| Two-Stream       |    |    |    |    |    | 26.3| 28.0 | 30.1 | 32.1 |    | 28.3| 35.6 | 44.5 | 51.5 | 53.1 |
| MS-LSTM          |    |    |    |    |    |    |    |    |    |    | 36.8| 37.5 | 40.0 | 40.0 | 41.2 |
| Ours MM-LSTM     |    |    |    |    |    | 37.1| 38.0 | 35.5 | 39.3 | 39.3 | 44.0| 45.3 | 51.3 | 60.2 | 63.1 |

Table 4.3: Results on the Random split of VIENA$^2$ for our three baselines with our action descriptors and for our approach.

|                  | 1" | 2" | 3" | 4" | 5" | 1" | 2" | 3" | 4" | 5" | 1" | 2" | 3" | 4" | 5" |
|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| CNN+LSTM         | 24.6| 25.6| 28.0| 30.0| 30.3| 26.8| 30.5| 40.4| 53.4| 62.6| 28.5| 35.8| 57.8| 68.1| 78.7|
| Two-Stream       | 56.7| 58.3| 59.0| 61.6| 61.7| 70.0| 72.0| 74.0| 77.1| 79.7| 69.6| 75.3| 80.6| 83.3| 83.6|
| MS-LSTM          | 28.0| 28.7| 30.6| 32.2| 32.8| 30.6| 38.7| 48.0| 49.6| 54.1| 33.3| 39.4| 48.3| 57.1| 61.0|
| Ours MM-LSTM     | 39.6| 39.6| 40.4| 42.0| 42.4| 42.0| 42.8| 44.4| 46.0| 48.0| 55.8| 57.6| 62.6| 69.0| 70.8|

4.4.1 Action Anticipation on VIENA$^2$

We report the results of our benchmark evaluation on the different scenarios of VIENA$^2$ in Table 4.2 for the original versions of the baselines, relying on the descriptors used in their respective paper, and in Table 4.3 for their modified versions that incorporate all descriptor types. Specifically, we report the recognition accuracies for all scenarios after every second of the sequences. Note that, in general, incorporating all descriptor types improves the results. Furthermore, while the action recognition baselines perform quite well in some scenarios, such as Accidents and Traffic Rules for the two-stream model, they are clearly outperformed by the anticipation methods in the other cases. Altogether, our new MM-LSTM consistently outperforms the baselines, thus showing the benefits of learning the dynamic importance of the modalities.

A comparison of the baselines with our approach on the Daytime and Weather partitions of VIENA$^2$ is provided in the supplementary material. In essence, the conclusions of these experiments are the same as those drawn above.

4.4.2 Challenges of VIENA$^2$

Based on the results above, we now study what challenges our dataset brings, such as which classes are the most difficult to predict and which classes cause the most confusion. We base this analysis on the per-class accuracies of our MM-LSTM model, which achieved the best performance in our benchmark. This, we believe, can suggest new directions to investigate in the future.

Our MM-LSTM per-class accuracies are provided in Table 4.4 and the corresponding confusion matrices at the earliest (after seeing 1 second) and latest (after seeing 5 seconds)
Table 4.4: Per-class accuracy of our approach on all scenarios of VIENA\(^2\) (Random).

|      | FF  | SS  | LL  | RR  | CL  | CR  | NA  | AP  | AC  | AA  | CD  | WD  | PK  | SR  | DO  | NF  | CR  | SS  | AS  | FF  | SS  | LL  | RR  | CL  | CR  |
|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 1\(^{st}\) | 50.7 | 43.8 | 38.7 | 35.0 | 18.7 | 26.1 | 96.9 | 65.7 | 63.4 | 42.9 | 45.4 | 100  | 75.4 | 76.1 | 75.2 | 100  | 81.6 | 79.4 | 84.3 | 100  | 80.5 | 86.1 | 80.2 | 85.7 | 75.0 | 40.0 | 95.1 | 48.6 |
| 2\(^{nd}\) | 60.1 | 46.8 | 83.3 | 35.0 | 18.7 | 26.1 | 97.7 | 70.7 | 71.6 | 75.0 | 96.9 | 49.3 | 29.7 | 52.3 | 37.9 | 79.6 | 51.2 | 71.4 | 53.4 | 63.0 | 51.2 | 71.4 | 53.4 | 76.9 | 48.6 | 37.1 | 45.9 | 49.6 | 52.0 |
| 3\(^{rd}\) | 81.3 | 54.6 | 43.4 | 63.4 | 42.9 | 45.4 | 100  | 75.4 | 76.1 | 75.2 | 83.7 | 60.0 | 35.1 | 69.5 | 45.8 | 70.4 | 67.6 | 70.2 | 58.6 | 79.6 | 49.3 | 29.7 | 52.3 | 37.9 | 57.3 | 63.9 | 100  | 50.7 | 52.7 |
| 4\(^{th}\) | 81.2 | 63.8 | 72.9 | 77.3 | 53.4 | 55.6 | 100  | 81.6 | 79.4 | 84.3 | 96.7 | 65.3 | 37.9 | 78.7 | 50.8 | 72.2 | 75.9 | 80.1 | 61.5 | 80.5 | 86.1 | 80.2 | 85.7 | 75.0 | 40.0 | 95.1 | 48.6 | 89.1 | 83.5 | 84.6 | 81.4 | 59.4 | 66.8 |
| 5\(^{th}\) | 88.0 | 97.2 | 95.8 | 90.4 | 64.9 | 65.4 | 100  | 81.6 | 79.4 | 84.3 | 85.7 | 75.0 | 40.0 | 95.1 | 48.6 | 74.1 | 78.2 | 76.6 | 63.6 | 91.2 | 85.5 | 84.6 | 81.4 | 59.4 | 66.8 | 89.1 | 83.5 | 84.6 | 81.4 | 59.4 | 66.8 |

Figure 4.5: **Confusion Matrices.** Confusion matrices of all five scenarios after observing 1 second (top) and 5 seconds (bottom) of each video sample.

Below, we discuss the challenges of the various scenarios.

1. **Driver maneuver:** After 1s, most actions are mistaken for *Moving Forward*, which is not surprising since the action has not started yet. After 5s, most of the confusion has disappeared, except for *Changing Lane* (left and right), for which the appearance, motion and vehicle dynamics are subject to small changes only, thus making this action look similar to *Moving Forward*.

2. **Accident:** Our model is able to distinguish *No Accident* from the different accident types early in the sequence. Some confusion between the different types of accident remains until after 5s, but this would have less impact in practice, as long as an accident is predicted.

3. **Traffic rule:** As in the maneuver case, there is initially a high confusion with *Correct Direction*, due to the fact that the action has not started yet. The confusion is then much reduced as we see more information, but *Passing a Red Light* remains relatively poorly predicted.

4. **Pedestrian intention:** The most challenging class for early prediction in this scenario is *Pedestrian Walking along the Road*. The prediction is nevertheless much improved after 5s.
Table 4.5: **Anticipating actions on real data.** Pre-training our MM-LSTM with our VIENA² dataset yields higher accuracy than training from scratch on real data.

| Model / Training          | After 1" | After 2" | After 3" | After 4" | After 5" |
|---------------------------|----------|----------|----------|----------|----------|
| MM-LSTM / From Scratch   | 41.01%   | 45.84%   | 51.38%   | 54.94%   | 56.12%   |
| MM-LSTM / Fine-Tuned      | 45.06%   | 54.15%   | 58.10%   | 65.61%   | 66.0%    |

5. **Front car intention:** Once again, at the beginning of the sequence, there is much confusion with the *Forward* class. After 5s, the confusion is significantly reduced, with, as in the maneuver case, some confusion remaining between the *Change lane* classes and the *Forward* class, illustrating the subtle differences between these actions.

### 4.4.3 Benefits of VIENA² for Anticipation from Real Images

To evaluate the benefits of our synthetic dataset for anticipation on real videos, we make use of the JAAD dataset [Rasouli et al., 2017] for pedestrian intention recognition, which is better suited to deep networks than other datasets, such as [Kooij et al., 2014b], because of its larger size (58 videos vs. 346). This dataset is, however, not annotated with the same classes as we have in VIENA², as its purpose is to study pedestrian and driver behaviors at pedestrian crossings. To make JAAD suitable for our task, we re-annotated its videos according to the four classes of our *Pedestrian Intention* scenario, and prepared a corresponding train/test split.

JAAD is also heavily dominated by the *Crossing* label, requiring augmentation of both training and test sets to have a more balanced number of samples per class.

To demonstrate the benefits of VIENA² in real-world applications, we conduct two sets of experiments: 1) Training on JAAD from scratch, and 2) Pre-training on VIENA² followed by fine-tuning on JAAD. For all experiments, we use appearance-based and motion-based features, which can easily be obtained for JAAD. The results are shown in Table 4.5. This experiment clearly demonstrates the effectiveness of using our synthetic dataset that contains photo-realistic samples simulating real-world scenarios.

Another potential benefit of using synthetic data is that it can reduce the amount of real data required to train a model. To evaluate this, we fine-tuned an MM-LSTM trained on VIENA² using a random subset of JAAD ranging from 20% to 100% of the entire dataset. The accuracies at every second of the sequence and for different percentages of JAAD data are shown in Fig. 4.6. Note that with 60% of real data, our MM-LSTM pre-trained on VIENA² already outperforms a model trained from scratch on 100% of the JAAD data. This shows that our synthetic data can save a considerable amount of labeling effort on real images.

### 4.4.4 Bias Analysis

For a dataset to be unbiased, it needs to be representative of the entire application domain it covers, thus being helpful in the presence of other data from the same application domain. This is what we aimed to achieve when capturing data in a large diversity of environmental conditions. Nevertheless, every dataset is subject to some bias. For example, since our data is synthetic, its appearance differs to some degree from real images, and the environments we cover are limited by those of the GTA V video game. However, below, we show empirically...
that the bias in VIENA\(^2\) remains manageable, making it useful beyond evaluation on VIENA\(^2\) itself. In fact, the experiments of Section 4.4.3 on real data already showed that performance on other datasets, such as JAAD, can be improved by making use of VIENA\(^2\). To further evaluate the bias of the visual appearance of our dataset, we relied on the idea of domain adversarial training introduced in [Ganin and Lempitsky, 2015]. In short, given data from two different domains, synthetic and real in our case, domain adversarial training aims to learn a feature extractor, such as a DenseNet, so as to fool a classifier whose goal is to determine from which domain a sample comes. If the visual appearance of both domains is similar, such a classifier should perform poorly. We therefore trained a DenseNet to perform action classification from a single image using both VIENA\(^2\) and JAAD data, while learning a domain classifier to discriminate real samples from synthetic ones. The performance of the domain classifier quickly dropped down to chance, i.e., 50%. To make sure that this was not simply due to failure to effectively train the domain classifier, we then froze the parameters of the DenseNet while continuing to train the domain classifier. Its accuracy remained close to chance, thus showing that the features extracted from both domains were virtually indistinguishable. Note that the accuracy of action classification improved from 18% to 43% during the training, thus showing that, while the features are indistinguishable to the discriminator, they are useful for action classification.

In our context of synthetic data, another source of bias could arise from the specific users who captured the data. To analyze this, we trained an MM-LSTM model from the data acquired by a single user, covering all classes and all environmental conditions, and tested it on the data acquired by another user. In Table 4.6, we compare the average accuracies of this experiment...
Table 4.6: Accuracy of MM-LSTM on the Driver Maneuver scenario when the training and test sets were captured by different users.

| Train Set Captured by | Test Set Captured by | After 1" | After 2" | After 3" | After 4" | After 5" |
|-----------------------|----------------------|---------|---------|---------|---------|---------|
| User 1                | User 1               | 32.0%   | 38.5%   | 60.5%   | 71.5%   | 83.6%   |
| User 1                | User 2               | 32.8%   | 37.3%   | 60.7%   | 70.9%   | 82.8%   |

Table 4.7: Results on the Weather split of VIENA2 for all scenarios and for our three baselines and our approach.

|          | CNN+LSTM | Two-Stream | MS-LSTM | Ours MM-LSTM |
|----------|----------|------------|---------|--------------|
| 1"       | 1"       | 1"         | 1"      | 1"           |
| 2"       | 2"       | 2"         | 2"      | 2"           |
| 3"       | 3"       | 3"         | 3"      | 3"           |
| 4"       | 4"       | 4"         | 4"      | 4"           |
| 5"       | 5"       | 5"         | 5"      | 5"           |

4.5 Benchmark Evaluation on All Splits of VIENA2

We now report our results on the different scenarios of VIENA2. In Table 4.7 and Table 4.8, we report the recognition accuracies for all scenarios after every second of the sequences. Note that some scenarios are easier to anticipate than others, e.g., accidents vs. driver maneuvers. Altogether, the trend remains the same as in the Random split.

4.6 Ablation Study for our MM-LSTM

In this section, we evaluate different aspects of our new MM-LSTM model.

Effect of Loss Function  We first evaluate the influence of our new loss function on our results. To this end, we replaced it with the standard cross-entropy loss, and with the loss of [Sadegh Aliakbarian et al., 2017], which has proven effective at action anticipation and corresponds to a linear weight instead of our nonlinear one. In Table 4.9, we compare the results of our MM-LSTM model trained with these losses and with ours on the Accidents scenario of VIENA2. Note that our loss yields more accurate predictions, particularly in the early stages of the videos. This confirms the effectiveness of our false positive weighting strategy, well-suited for action anticipation.

Effect of Different Features  We then analyze the impact of the different features on our results. To this end, we trained separate MM-LSTM models with a single modality as input. We then evaluate these models on the Accidents and Front car intention scenarios of VIENA2. The results are provided in Table 4.10. Note that, in the former scenario, speed and appearance...
Table 4.8: Results on the Daytime split of VIENA$^2$ for all scenarios and for our three baselines and our approach.

|        | CNN+LSTM | Two-Stream | MS-LSTM | Ours MM-LSTM |
|--------|-----------|------------|---------|--------------|
|        | 1" 2" 3" 4" 5" | 1" 2" 3" 4" 5" | 1" 2" 3" 4" 5" | 1" 2" 3" 4" 5" |
| DM     | 23.4 24.1 27.2 27.8 30.0 | 26.3 28.4 33.5 40.2 42.4 | 27.0 34.2 55.3 62.1 75.9 | 33.3 40.4 57.2 77.6 84.3 |
| AC     | 48.4 52.3 52.8 55.9 57.4 | 47.2 49.4 51.0 55.9 60.0 | 48.0 50.4 53.2 57.8 61.6 | 50.7 54.5 59.7 64.7 66.2 |
| TR     | 26.3 26.9 27.3 29.8 31.1 | 29.1 34.6 36.8 38.0 44.4 | 33.1 36.8 42.5 49.3 54.7 | 35.0 39.5 49.6 61.1 68.2 |
| PI     | 38.0 38.2 39.3 40.1 41.3 | 39.3 40.9 44.1 50.2 55.5 | 46.5 53.3 60.0 66.1 68.7 | 49.0 61.8 69.1 70.1 70.1 |
| FCI    | 33.5 36.0 37.5 38.8 39.0 | 34.5 35.2 38.8 44.3 49.6 | 41.1 45.5 53.7 56.7 62.2 | 41.8 45.1 57.1 69.4 77.1 |

Table 4.9: **Influence of the loss function.** We compare the use of the standard cross-entropy, the loss introduced in Chapter 3, and our loss within our MM-LSTM model using the Accidents of VIENA$^2$. Note that our loss consistently yields higher accuracy, particularly in the early stages of the videos.

| Loss Function                  | After 1 sec | After 2 sec | After 3 sec | After 4 sec | After 5 sec |
|-------------------------------|-------------|-------------|-------------|-------------|-------------|
| wo/ AvgPool                   |             |             |             |             |             |
| Cross Entropy                 | 48.3%       | 60.3%       | 72.0%       | 76.8%       | 76.9%       |
| Linear weight (as in Chapter 3)| 45.2%       | 58.0%       | 70.2%       | 77.7%       | 77.7%       |
| Ours                          | **56.8%**   | **67.0%**   | **76.7%**   | **80.4%**   | **80.3%**   |
| w/ AvgPool                    |             |             |             |             |             |
| Cross Entropy                 | 69.3%       | 74.3%       | 78.0%       | 82.0%       | 83.3%       |
| Linear weight (as in Chapter 3)| 73.3%       | 75.7%       | 79.0%       | 82.7%       | 83.3%       |
| Ours                          | **76.3%**   | **79.0%**   | **81.7%**   | **86.3%**   | **86.7%**   |

Table 4.10: **Influence of different features.** We evaluate the accuracy obtained by using our different features individually on the Accidents and Front car intention scenarios for the Random partition. See the discussion in the text for more detail.

| Features | After 1 sec | After 2 sec | After 3 sec | After 4 sec | After 5 sec |
|----------|-------------|-------------|-------------|-------------|-------------|
| Accident |             |             |             |             |             |
| Steering | 26.0%       | 27.5%       | 27.7%       | 28.2%       | 28.7%       |
| Speed    | 46.3%       | 46.3%       | 48.8%       | 49.3%       | 50.7%       |
| Appearance | 48.1%       | 48.7%       | 51.0%       | 51.3%       | 53.7%       |
| Motion   | 21.7%       | 23.6%       | 24.3%       | 24.3%       | 25.5%       |
| All      | **76.3%**   | **79.0%**   | **81.7%**   | **86.3%**   | **86.7%**   |

| Front car intention          |             |             |             |             |             |
| Steering | 34.2%       | 33.3%       | 34.7%       | 34.7%       | 35.0%       |
| Speed    | 17.3%       | 15.0%       | 16.2%       | 16.5%       | 17.3%       |
| Appearance | 29.8%       | 30.3%       | 29.9%       | 29.6%       | 30.3%       |
| Motion   | 34.7%       | 36.9%       | 42.0%       | 47.0%       | 55.3%       |
| All      | **49.9%**   | **51.7%**   | **60.4%**   | **71.5%**   | **77.8%**   |

play an important role, whereas in the latter one, the steering angle and motion are the dominant
features, closely followed by appearance. This shows that different scenarios need to rely more strongly on different features. Nevertheless, using all descriptors jointly always improves over using them individually, which evidences that our MM-LSTM model can effectively discover the importance of the different modalities.

**Effect of the Number of Hidden Units** Furthermore, we study the effect of varying the number of hidden units in the LSTM blocks of our MM-LSTM model. To this end, we simultaneously varied this number between 256 and 4096 across all LSTM units in our model. As can be seen in Table 4.11 where we provide the detailed numbers for this analysis for the Random partition, the results are stable and only marginally improving with more hidden units. However, the number of parameters of the MM-LSTM model increases drastically, in this case from 7.6M to 436.3M. In our experiments in this chapter, we therefore used 1024 hidden units for all LSTMs, which is a good balance between performance and number of parameters (46.2M).

**Table 4.11: Effect of number of LSTM hidden units on anticipation performance.** Using 1024 units represents a good trade-off between memory consumption and accuracy.

| Units  | Params | After 1 sec | After 2 sec | After 3 sec | After 4 sec | After 5 sec |
|--------|--------|-------------|-------------|-------------|-------------|-------------|
| 256    | 7.6M   | 68.77%      | 71.33%      | 77.50%      | 82.11%      | 82.32%      |
| 512    | 17.8M  | 75.01%      | 77.78%      | 79.90%      | 84.63%      | 86.10%      |
| 1024   | 46.2M  | 76.31%      | 79.00%      | 81.73%      | 86.29%      | 86.68%      |
| 2048   | 134.3M | 76.45%      | 79.22%      | 82.01%      | 86.43%      | 87.01%      |
| 4096   | 436.3M | 76.66%      | 78.12%      | 83.33%      | 86.37%      | 86.60%      |

**Effect of Input Block** As mentioned earlier, the FC-Pool layer of our MM-LSTM model encodes the importance of the modalities and is shared across time. While, on its own, this cannot change the importance of the modalities over time, the modality-specific LSTMs can adapt their outputs over time so as to serve this purpose. To further evidence the importance of these two modules in learning a good combination of the modalities, we performed an experiment where we deactivated them in turn, and trained models on all 24 permutations of the 4 input modalities. The results in Table 4.12 where we provide the average accuracy and standard deviation over the 24 permutations, show that using both modules not only yields higher accuracy, but also makes our network virtually unaffected by the order of the modalities, i.e., very small standard deviation.

**Comparison to the State of the Art** To evaluate the effectiveness of our model on another dataset, and compare it to the state-of-the-art anticipation techniques, we make use of the standard JHMDB-21 benchmark. For this experiment, we followed the experimental setup of [Sadegh Aliakbarian et al., 2017] as we did in Chapter 3 and used their context-aware and action-aware features instead of our features, to make the comparison more fair. The results of several state-of-the-art methods on this dataset are shown in Table 4.13. Note that we outperform all these methods. Importantly, we outperform the multi-stage LSTM introduced in
Table 4.12: Importance of the Input Block to learn a good combination of the modalities. We report the mean accuracy and standard deviation over the 24 permutations of the 4 input modalities for the Accident scenario. Note that using both modality-specific LSTMs (L) and the FC-Pool (P) layer yields not only the highest accuracies, but also very low std, thus indicating that the modality order has virtually no impact on our results.

| LSTM | FC-Pool | After 1" | After 2" | After 3" | After 4" | After 5" |
|------|---------|----------|----------|----------|----------|----------|
| w/   | w/      | 76.38 ± 0.16 | 79.08 ± 0.18 | 81.52 ± 0.16 | 86.30 ± 0.25 | 86.56 ± 0.15 |
| w/   | wo/     | 70.04 ± 2.37 | 73.31 ± 3.11 | 76.41 ± 3.05 | 80.78 ± 2.37 | 82.25 ± 2.78 |
| wo/  | w/      | 50.58 ± 4.02 | 51.48 ± 4.01 | 53.21 ± 3.63 | 54.47 ± 3.10 | 56.03 ± 3.24 |
| wo/  | wo/     | 48.48 ± 4.71 | 49.27 ± 4.36 | 50.76 ± 3.73 | 52.58 ± 3.06 | 53.57 ± 2.25 |

Chapter[3] which manually defines the order in which the two feature types should be considered. This, we believe, truly evidences the benefits of our approach, which can automatically learn the dynamic importance of the input modalities for the final prediction. Moreover, we evaluated our model when trained with the loss function introduced in Chapter[3] Note that, in the JHMDB-21 dataset, all actions start from the beginning of the videos and continue to the end. This, however, is not the case in VIENA^2, where the actions begin to occur only during the second half of the video. Therefore, the loss function introduced in Chapter[3] which linearly grows over time, performs better on JHMDB-21, as it is better suited for such cases, but performs worse on VIENA^2, as shown in Table[4.9] In other words, the design of the loss function should consider the nature of the target dataset. JHMDB depicts non-driving activities, where the actions occur over the entire duration of the video. This is in fact a case where linearly growing weights better model the way the action occurs, i.e., linearly over the entire sequence, and thus perform better. Note, however, that the best accuracy in Table[4.13] is still achieved by our approach, but by using linear weights, and that our approach with nonlinear weights follows very closely, thus showing good robustness to the precise definition of the weights.

Table 4.13: Comparison to the state-of-the-art anticipation techniques on JHMDB-21. Note that our MM-LSTM architecture outperforms all the baselines, thus achieving state-of-the-art performance on this dataset.

| Architecture / Method | Earliest (20%) | Latest (100%) |
|-----------------------|----------------|---------------|
| DP-SVM [Soomro et al., 2016a] | 5.0% | 46.0% |
| S-SVM [Soomro et al., 2016a] | 5.0% | 43.0% |
| Where/What [Soomro et al., 2016b] | 10.0% | 43.0% |
| Ranking Loss [Ma et al., 2016] | 29.0% | 43.0% |
| MS-LSTM (Our approach in Chapter[3]) | 55.0% | 58.0% |
| MM-LSTM w/ Loss of [Sadegh Aliakbarian et al., 2017] | 59.3% | 61.0% |
| MM-LSTM w/ our Loss | 58.1% | 62.7% |
4.7 VIENA\textsuperscript{2} Visualization: Environmental Diversity

To collect data for the different scenarios of VIENA\textsuperscript{2}, we made use of various vehicles, including heavy vehicles for Driver Maneuver of heavy vehicles’ driver. To provide diversity in terms of vehicle conditions, type, etc., we made use of different kinds of cars, such as old cars that are harder to control during turns or at high speed, and sport cars that are better-suited when driving at high speed (see Fig. 4.7). For heavy vehicles, we collected data with two types of vehicles, trucks and buses. However, again for the sake of diversity, we made use of heavy vehicles with different specifications, top speed, and length (see Fig. 4.8).

Note that, for all of our videos, the camera was mounted on the car so as to provide us with a front view. For heavy vehicles, the camera was mounted in a similar manner, but providing a higher viewpoint, since heavy vehicles are taller. In Figs. 4.9, 4.10 and 4.11 we provide example images acquired at different daytimes, i.e., from pre-dawn to midnight, in different weather conditions and at different locations, respectively. We believe that the combination of these conditions, i.e., different vehicles, daytimes, weathers, and locations, provide a very diverse set of data which will reduce the effect of dataset bias.
In this chapter, we have introduced a new large-scale dataset for general action anticipation in driving scenarios, which covers a broad range of situations with a common set of sensors. Furthermore, we have proposed a new MM-LSTM architecture allowing us to learn the importance of multiple input modalities for action anticipation. Our experimental evaluation has shown the benefits of our new dataset and of our new model. This chapter also concludes our contributions on the task of anticipating a discrete representation of a deterministic future. In the next chapter, we start studying the problem of anticipating a continuous representation of a stochastic future, with the focus on diverse human motion prediction.
Figure 4.11: Examples of different locations, including suburbs, downtown, off-road, nature, highway, and industrial locations.

Driver Maneuver: Turn Left

Accident: Accident with Pedestrian

Traffic Rules: Passing a Red Light

Pedestrian Intention: Crossing

Front Car Intention: Change Lane to Left

Figure 4.12: Examples of our action classes in different scenarios. Each row represents an action example from one scenario.
A Stochastic Conditioning Scheme for Diverse Human Motion Prediction

In this chapter, we start studying the problem of anticipating continuous future representations of a stochastic process. That is, for instance, the case of human motion prediction. Human motion prediction, the task of predicting future 3D human poses given a sequence of observed ones, has been mostly treated as a deterministic problem. However, human motion is a stochastic process: Given an observed sequence of poses, multiple future motions are plausible. Existing approaches to modeling this stochasticity typically combine a random noise vector with information about the previous poses. This combination, however, is done in a deterministic manner, which gives the network the flexibility to learn to ignore the random noise. Alternatively, in this chapter, we propose to stochastically combine the root of variations with previous pose information, so as to force the model to take the noise into account. We exploit this idea for motion prediction by incorporating it into a recurrent encoder-decoder network with a conditional variational autoencoder block that learns to exploit the perturbations. Our experiments on two large-scale motion prediction datasets demonstrate that our model yields high-quality pose sequences that are much more diverse than those from state-of-the-art stochastic motion prediction techniques.

5.1 Introduction

Human motion prediction aims to forecast the sequence of future poses of a person given past observations of such poses. To achieve this, existing methods typically rely on recurrent neural networks (RNNs) that encode the person’s motion [Martinez et al., 2017; Gui et al., 2018a; Walker et al., 2017; Kundu et al., 2018; Barsoum et al., 2018; Pavllo et al., 2019, 2018]. While they predict reasonable motions, RNNs are deterministic models and thus cannot account for the highly stochastic nature of human motion; given the beginning of a sequence, multiple, diverse futures are plausible. To correctly model this, it is therefore critical to develop algorithms that can learn the multiple modes of human motion, even when presented with only deterministic training samples.

Recently, several attempts have been made at modeling the stochastic nature of human motion [Yan et al., 2018; Barsoum et al., 2018; Walker et al., 2017; Kundu et al., 2018; Lin and Amer, 2018]. These methods rely on sampling a random vector that is then combined
with an encoding of the observed pose sequence. In essence, this combination is similar to the conditioning of generative networks; the resulting models aim to generate an output from a random vector while taking into account additional information about the content.

While standard conditioning strategies, i.e., concatenating the condition to the latent variable, may be effective for many tasks, as in [Yan et al., 2016; Kulkarni et al., 2015; Esser et al., 2018; Engel et al., 2017; Bao et al., 2017; Larsen et al., 2015], they are ill-suited for motion prediction. The reason is the following: In other tasks, the conditioning variable only provides auxiliary information about the output to produce, such as the fact that a generated face should be smiling. By contrast, in motion prediction, it typically contains the core signal to produce the output, i.e., the information about the previous poses. We empirically observed that, since the prediction model is trained using deterministic samples (i.e., one condition per sample), it can then simply learn to ignore the random vector and still produce a meaningful output based on the conditioning variable only. In other words, the model can ignore the root of variations, and thus essentially become deterministic. This problem was discussed in [Bowman et al., 2015] in the context of unconditional text generation, and we identified it in our own motion prediction experiments.

We introduce a simple yet effective approach to counteracting this loss of diversity and thus to generating truly diverse future pose sequences. At the heart of our approach lies the idea of Mix-and-Match perturbations: Instead of combining a noise vector with the conditioning variables in a deterministic manner, we randomly select and perturb a subset of these variables. By randomly changing this subset at every iteration, our strategy prevents training from identifying the root of variations and forces the model to take it into account in the generation process. Consequently, as supported by our experiments, our approach produces not only high-quality predictions but also truly diverse ones.

In short, our contributions in this chapter are (i) a novel way of imposing diversity into conditional VAEs, called Mix-and-Match perturbations; (ii) a new motion prediction model capable of generating multiple likely future pose sequences from an observed motion; (iii) a new set of evaluation metrics for quantitatively measuring the quality and the diversity of generated motions, thus facilitating the comparison of different stochastic approaches; and (iv) a curriculum learning paradigm for training generative models that use Mix-and-Match perturbation as the stochastic conditioning scheme. Despite its simplicity, curriculum learning of variation is essential to achieve optimal performance in case of imposing large variations.

5.2 Related Work

**Deterministic Motion Prediction.** Most motion prediction approaches are based on deterministic models [Pavllo et al., 2018; 2019; Gui et al., 2018a; Jain et al., 2016d; Martinez et al., 2017; Gui et al., 2018b; Fragkiadaki et al., 2015; Ghosh et al., 2017; Mao et al., 2019a], casting motion prediction as a regression task where only one outcome is possible given the observations. Due to the success of RNN-based methods at modeling sequence-to-sequence learning problems, many attempts have been made to address motion prediction within a recurrent framework [Martinez et al., 2017; Gui et al., 2018a; Walker et al., 2017; Kundu et al., 2018; Barsoum et al., 2018; Pavllo et al., 2019]. Typically, these approaches try to learn a mapping from the observed sequence of poses to the future sequence. Another group of
study addresses this problem within feed-forward models [Mao et al., 2019a; Li et al., 2018a; Butepage et al., 2017], either with fully-connected [Butepage et al., 2017], convolutional [Li et al., 2018a], or more recently, graph neural networks [Mao et al., 2019a]. While a deterministic approach may produce accurate predictions, it fails to reflect the stochastic nature of human motion, where multiple plausible outcomes can be highly likely for a single given series of observations. Modeling this diversity is the topic of this chapter, and we therefore focus the discussion below on the other methods that have attempted to do so.

**Stochastic Motion Prediction.** The general trend to incorporate variations in the predicted motions consists of combining information about the observed pose sequence with a random vector. In this context, two types of approaches have been studied: The techniques that directly incorporate the random vector into the RNN decoder, e.g., as in GANs, and those that make use of an additional Conditional Variational Autoencoder (CVAE) [Sohn et al., 2015] to learn a latent variable that acts as the root of variation.

In the first class of methods, [Lin and Amer, 2018] sample a random vector $z_t \sim \mathcal{N}(0, I)$ at each time step and add it to the pose input to the RNN decoder. By relying on different random vectors at each time step, however, this strategy is prone to generating discontinuous motions. To overcome this, [Kundu et al., 2018] make use of a single random vector to generate the entire sequence. This vector is both employed to alter the initialization of the decoder and concatenated with a pose embedding at each iteration of the RNN. By relying on concatenation as a mean to fuse the condition and the random vector, these two methods contain parameters that are specific to the random vector, and thus give the model the flexibility to ignore this information. In [Barsoum et al., 2018], instead of using concatenation, the random vector is added to the hidden state produced by the RNN encoder. While addition prevents having parameters that are specific to the random vector, this vector is first transformed by multiplication with a parameter matrix, and thus can again be zeroed out so as to remove the source of diversity, as we observe empirically in Section 5.4.2.

The second category of stochastic methods introduce an additional CVAE between the RNN encoder and decoder. This allows them to learn a more meaningful transformation of the noise, combined with the conditioning variables, before passing the resulting information to the RNN decoder. In this context, [Walker et al., 2017] propose to directly use the pose as conditioning variable. As will be shown in our experiments, while this approach is able to maintain some degree of diversity, albeit less than ours, it yields motions of lower quality because of its use of independent random vectors at each time step. In [Butepage et al., 2018], an approach similar to that of [Walker et al., 2017] is proposed, but with one CVAE per limb. As such, this method suffers from the same discontinuity problem as [Walker et al., 2017; Lin and Amer, 2018]. Finally, instead of perturbing the pose, the recent work of [Yan et al., 2018] uses the RNN decoder hidden state as conditioning variable in the CVAE, concatenating it with the random vector. While this approach generates high-quality motions, it suffers from the fact that the CVAE decoder gives the model the flexibility to ignore the random vector.

Ultimately, both classes of methods suffer from the fact that they allow the model to ignore the random vector, thus relying entirely on the conditioning information to generate future poses. Here, we introduce an effective way to maintain the root of diversity by randomizing the combination of the random vector with the conditioning variable.
5.3 Proposed Method

In this section, we first introduce our Mix-and-Match approach to introducing diversity in CVAE-based motion prediction. We then describe the motion prediction architecture we used in our experiments and propose a novel evaluation metric to quantitatively measure the diversity and quality of generated motions.

Figure 5.1: Mix-and-Match perturbation. (Top) Illustration of the Sampling operation (left) and of the Resampling one (right). Given a sampling rate $\alpha$ and a vector length $L$, the Sampling operation samples $\lceil \alpha L \rceil$ indices, say $I$. The complementary, unsampled indices are denoted by $\bar{I}$. Then, given two $L$-dimensional vectors and the corresponding $\lceil \alpha L \rceil$ and $\lfloor (1 - \alpha)L \rfloor$ indices, the Resampling operation mixes the two vectors to form a new $L$-dimensional one. (Middle) Example of Mix-and-Match perturbation. (Bottom) Example of perturbation by concatenation, as in [Yan et al., 2018]. Note that, in Mix-and-Match perturbations, sampling is stochastic; the indices are sampled uniformly randomly for each mini-batch. By contrast, in [Yan et al., 2018], sampling is deterministic, and the indices in $I$ are fixed and correspond to $I = \{1, \ldots, \frac{L}{2}\}$.

5.3.1 Mix-and-Match Perturbation

The main limitation of prior work in the area of stochastic motion modeling, such as [Walker et al., 2017; Barsoum et al., 2018; Yan et al., 2018], lies in the way they fuse the random vector with the conditioning variable, i.e., RNN hidden state or pose, which causes the model to learn to ignore the randomness and solely exploit the deterministic conditioning information to generate motion. To overcome this, we propose to make it harder for the model to decouple the random variable from the deterministic information. Specifically, we observe that the way the random variable and the conditioning one are combined in existing methods is deterministic. We therefore propose to make this process stochastic.

Similarly to [Yan et al., 2018], we propose to make use of the hidden state as the conditioning variable and generate a perturbed hidden state by combining a part of the original hidden
5.3 Proposed Method

Figure 5.2: Overview of our approach. (Top): Overview of the model during inference. During inference, given past information and a random vector sampled from a Normal distribution, the model generates new motions. (Bottom): Overview of the model during training. During training, we use a future pose autoencoder with a CVAE between the encoder and the decoder. The RNN encoder-decoder network mapping the past to the future then aims to generate good conditioning variables for the CVAE.

More specifically, as depicted in Fig. 5.1 a mix-and-match perturbation takes two vectors of size $L$ as input, say $h_t$ and $z$, and combines them in a stochastic manner. To this end, it relies on two operations. The first one, called Sampling, chooses $\lfloor \alpha L \rfloor$ indices uniformly at random among the $L$ possible values, given a sampling rate $0 \leq \alpha \leq 1$. Let us denote by $\mathcal{I} \subseteq \{1, \ldots, L\}$, the resulting set of indices and by $\mathcal{I}^c$ the complementary set. The second operation, called Resampling, then creates a new $L$-dimensional vector whose values at indices in $\mathcal{I}$ are taken as those at corresponding indices in the first input vector and the others at the complementary indices, of dimension $\lfloor (1 - \alpha) L \rfloor$, in the second input vector.

5.3.2 M&M Perturbation for Motion Prediction

Let us now describe the way we use our mix-and-match perturbation strategy for motion prediction. To this end, we first discuss the network we rely on during inference, and then explain our training strategy.

**Inference.** The high-level architecture we use at inference time is depicted by Fig. 5.2 (Top). It consists of an RNN encoder that takes $t$ poses $x_{1:t}$ as input and outputs an $L$-dimensional hidden vector $h_t$. A random $\lfloor \alpha L \rfloor$-dimensional portion of this hidden vector, $h_{\mathcal{I}}$, is then combined with an $\lfloor (1 - \alpha) L \rfloor$-dimensional random vector $z \sim \mathcal{N}(0, I)$ via our mix-and-match perturbation strategy. The resulting $L$-dimensional output is passed through a small neural network (i.e., ResBlock2 in Fig. 5.2) that reduces its size to $\lfloor \alpha L \rfloor$, and then fused with the remaining $\lfloor (1 - \alpha) L \rfloor$-dimensional portion of the hidden state, $h_{\mathcal{I}}$. This, in turn, is passed through the VAE decoder to produce the final hidden state $h_z$, from which the future poses $x_{t+1:T}$ are obtained via the RNN decoder.

**Training.** During training, we aim to learn both the RNN parameters and the CVAE ones.
Because the CVAE is an autoencoder, it needs to take as input information about future poses. To this end, we complement our inference architecture with an additional RNN future encoder, yielding the training architecture depicted in Fig. 5.2 (Bottom). Note that, in this architecture, we incorporate an additional mix-and-match perturbation that fuses the hidden state of the RNN past encoder \( h_t \) with that of the RNN future encoder \( h_T \) and forms \( h_{Tt} \). This allows us to condition the VAE encoder in a manner similar to the decoder. Note that, for each mini batch, we use the same set of sampled indices for all mix-and-match perturbation steps throughout the network. Furthermore, following the standard CVAE strategy, during training, the random vector \( z_p \) is sampled from the approximate posterior distribution \( \mathcal{N}(\mu_\theta(x), \Sigma_\theta(x)) \), whose mean \( \mu_\theta(x) \) and covariance matrix \( \Sigma_\theta(x) \) are produced by the CVAE encoder with parameters \( \theta \). This, in practice, is done by the reparameterization technique [Kingma and Welling, 2013].

To learn the parameters of our model, we rely on the availability of a dataset \( D = \{ X_1, X_2, ..., X_N \} \) containing \( N \) videos \( X_i \) depicting a human performing an action. Each video consists of a sequence of \( T \) poses, \( X_i = \{ x_1^i, x_2^i, ..., x_T^i \} \), and each pose comprises \( J \) joints forming a skeleton, \( x_t^i = \{ x_{t1}^i, x_{t2}^i, ..., x_{tJ}^i \} \). The pose of each joint is represented as a 4D quaternion. Given this data, we train our model by minimizing a loss function of the form

\[
\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \left( \mathcal{L}_{rot}(X_i) + \mathcal{L}_{skl}(X_i) \right) + \lambda \mathcal{L}_{prior} .
\]

The first term in this loss compares the output of the network with the ground-truth motion using the squared loss. That is,

\[
\mathcal{L}_{rot}(X_i) = - \sum_{k=t+1}^{T} \sum_{j=1}^{J} \left\| \hat{x}_{k}^{i,j} - x_{k}^{i,j} \right\|^2 ,
\]

where \( \hat{x}_{k}^{i,j} \) is the predicted 4D quaternion for the \( j^{th} \) joint at time \( k \) in sample \( i \), and \( x_{k}^{i,j} \) the corresponding ground-truth one. The main weakness of this loss is that it treats all joints equally. However, when working with angles, some joints have a much larger influence on the pose than others. For example, because of the kinematic chain, the pose of the shoulder affects that of the rest of the arm, whereas the pose of the wrists has only a minor effect. To take this into account, we define our second loss term as the error in 3D space. That is,

\[
\mathcal{L}_{skl}(X_i) = - \sum_{k=t+1}^{T} \sum_{j=1}^{J} \left\| \hat{p}_{k}^{i,j} - p_{k}^{i,j} \right\|^2 ,
\]

where \( \hat{p}_{k}^{i,j} \) is the predicted 3D position of joint \( j \) at time \( k \) in sample \( i \) and \( p_{k}^{i,j} \) the corresponding ground-truth one. These 3D positions can be computed using forward kinematics, as in [Pavllo et al., 2018, 2019]. Note that, to compute this loss, we first perform a global alignment of the predicted pose and the ground-truth one by rotating the root joint to face \( [0, 0, 0] \).
§5.3 Proposed Method

Known State

Completely Random

Training Progress

Figure 5.3: Example of curriculum perturbation of the hidden state.

Finally, following standard practice in training VAEs, we define our third loss term as the KL divergence

\[
L_{\text{prior}} = -KL\left(\mathcal{N}(\mu_\theta(x), \Sigma_\theta(x)) \parallel \mathcal{N}(0, I)\right)
\]

\[
= -\frac{1}{2} \sum_{j=1}^{d} \left(1 + \log(\sigma_\theta(x)_j^2) - \mu_\theta(x)_j^2 - \sigma_\theta(x)_j^2\right).
\] (5.4)

where \(\Sigma_\theta(x) = \text{diag}(\sigma_\theta(x)^2)\) and \(d\) is the length of the diagonal of the covariance matrix. In practice, since our VAE appears within a recurrent model, we weigh \(L_{\text{prior}}\) by a function \(\lambda\) corresponding to the KL annealing weight of [Bowman et al., 2015]. We start from \(\lambda = 0\), forcing the model to encode as much information in \(z\) as possible, and gradually increase it to \(\lambda = 1\), following a logistic curve.

5.3.3 Curriculum Learning of Variation

The parameter \(\alpha\) in our mix-and-match perturbation scheme determines a trade-off between stochasticity and motion quality. The larger \(\alpha\), the larger the portion of the original hidden state that will be perturbed. Thus, the model incorporates more randomness and less information from the original hidden state. As such, given a large \(\alpha\), it becomes harder for the model to deliver motion information from the observation to the future representation since a large portion of the hidden state is changing randomly. In particular, we observed that training becomes unstable if we use a large \(\alpha\) from the beginning, with the motion-related loss terms fluctuating while the prior loss \(L_{\text{prior}}\) quickly converges to zero. To overcome this while still enabling the use of sufficiently large values of \(\alpha\) to achieve high diversity, we introduce the curriculum learning strategy depicted by Fig. 5.3. In essence, we initially select \(\lceil \alpha L \rceil\) indices in a deterministic manner and gradually increase the randomness of these indices as training progresses. More specifically, given a set of \(\lceil \alpha L \rceil\) indices, we replace \(c\) indices from the sampled ones with the corresponding ones from the remaining \(\lfloor (1 - \alpha) L \rfloor\) indices. Starting from \(c = 0\), we gradually increase \(c\) to the point where all \(\lceil \alpha L \rceil\) indices are sampled uniformly.
randomly. More details, including the pseudo-code of this approach, are provided in the supplementary material. This strategy helps the motion decoder to initially learn and incorporate information about the observations (as in [Yan et al., 2018]), yet, in the long run, still prevents it from ignoring the random vector.

### 5.3.4 Quality and Diversity Metrics

When dealing with multiple plausible motions, or in general diverse solutions to a problem, evaluation is a challenge. The standard metrics used for deterministic motion prediction models are ill-suited to this task, because they typically compare the predictions to the ground truth, thus inherently penalizing diversity. For multiple motions, two aspects are important: the diversity and the quality, or realism, of each individual motion. Prior work typically evaluates these aspects via human judgement. While human evaluation is highly valuable, and we will also report human results, it is very costly and time-consuming. Here, we therefore introduce two metrics that facilitate the quantitative evaluation of both quality and diversity of generated human motions. We additionally extend the Inception-Score [Salimans et al., 2016] to our task.

To measure the quality of generated motions, we propose to rely on a binary classifier trained to discriminate real (ground-truth) samples from fake (generated) ones. The accuracy of this classifier on the test set is thus inversely proportional to the quality of the generated motions. In other words, high-quality motions are those that are not distinguishable from real ones. Note that we do not rely on adversarial training, i.e., we do not define a loss based on this classifier when training our model. To measure the diversity of the generated motions, a naive approach would consist of relying on the distance between the generated motion and a reference one. However, generating identical motions that are all far from the reference one would therefore yield a high value, while not reflecting diversity. To prevent this, we propose to make use of the average distance between all pairs of generated motions. A similar idea has been investigated to measure the diversity of solutions in other domains [Yuan and Kitani, 2019; Yang et al., 2019].

The quality and diversity metrics can reliably evaluate a stochastic motion prediction model. While providing valuable information, drawing conclusion about the performance of a model is always easier with a single measure. To this end, we extend the Inception-Score (IS) [Salimans et al., 2016] used to measure the quality of images produced by a generative model. Our extension to IS is twofold: (1) Inspired by [Huang et al., 2018], we extend IS to the conditional case, where the condition provides the core signal to generate the sample; (2) Our extended IS measures the quality and diversity of sequential solutions. To this end, we first train a strong skeleton-based action classifier [Li et al., 2018b] on ground-truth motions. With then compute the IS of each of the multiple motions generated for a given condition (observed motion), and report the mean IS and its standard deviation over all conditions. The reason behind reporting the mean IS over all conditions is to evaluating the diversity of generated motions given each observation. Note that studying IS only makes it hard to evaluate quality and diversity separately, and thus we still believe that all three metrics are required. Importantly, we show empirically that our proposed metrics are in line with human judgement, at considerably lower cost.
§5.4 Experiments

We now evaluate the effectiveness of our approach at generating multiple plausible motions. To this end, we use Human3.6M [Ionescu et al., 2014] and the CMU Mocap dataset\(^1\), two large publicly available motion capture datasets. In this section, we introduce the baselines and give information about the implementation details and evaluation metrics. We then provide all the experimental results.

**Baselines.** We compare our Mix-and-Match approach with the different means of imposing variation in motion prediction discussed in Section 5.2, i.e., concatenating the hidden state to a learned latent variable, Yan et al., [Yan et al., 2018], concatenating the pose to a learned latent variable at each time-step, Walker et al., [Walker et al., 2017], and adding a (transformed)  

\(^1\)Available at [http://mocap.cs.cmu.edu/](http://mocap.cs.cmu.edu/).
A Stochastic Conditioning Scheme for Diverse Human Motion Prediction

Figure 5.5: Diversity of $K$ RNN decoder inputs, generated with $K = 50$ different random vectors. We report the mean diversity over $N = 50$ samples and the corresponding standard deviation.

random noise to the hidden state, Barsoum et al. [Barsoum et al., 2018]. For the comparison to be fair, we use 16 frames (i.e., 640ms) as observation to generate the next 60 frames (i.e., 2.4sec) for all baselines. All models are trained with the same motion representation, annealing strategy, backbone network, and losses, except for Barsoum et al. [Barsoum et al., 2018] which cannot make use of $L_{\text{prior}}$.

Implementation Details. The motion encoders and decoders in our model are single layer GRU [Cho et al., 2014] networks, comprising 1024 hidden units each. For the decoders, we use a teacher forcing technique [Williams and Zipser, 1989] to decode motion. At each time-step, the network chooses with probability $P_{tf}$ whether to use its own output at the previous time-step or the ground-truth pose as input. We initialize $P_{tf} = 1$, and decrease it linearly at each training epoch such that, after a certain number of epochs, the model becomes completely autoregressive, i.e., uses only its own output as input to the next time-step. We train our model on a single GPU with the Adam optimizer [Kingma and Ba, 2014] for 100K iterations. We use a learning rate of 0.001 and a mini-batch size of 64. To avoid exploding gradients, we use the gradient-clipping technique of [Pascanu et al., 2013] for all layers in the network. We implemented our model using the Pytorch framework of [Paszke et al., 2017].

Evaluation Metrics. In addition to the metrics discussed in Section 5.3.4, we also report the standard ELBO metric (approximated by the reconstruction loss and the KL on the test set) and the sampling loss (S-MSE) of our approach and the state-of-the-art stochastic motion prediction techniques. However, evaluating only against one ground-truth motion (i.e., one sample from multi-modal distribution), as in MSE or S-MSE, can lead to a high score for one sample while penalizing other plausible modes. This behavior is undesirable since it cannot differentiate a multi-modal solution from a good, but uni-modal one. Similarly, the metrics in [Yan et al., 2018] or the approximate ELBO only evaluate quality given one single ground truth. While the ground truth has high quality, there exist multiple high quality continuations of an observation, which our proposed metric accounts for. As discussed in Section 5.3.4.
§5.4 Experiments

Figure 5.6: (Top) Diversity of our approach and the stochastic baselines. (Middle) Quality of our approach and the stochastic baselines. (Bottom) Comparing classifier-based and human evaluation of quality for our approach and the baselines, where the statistics correspond to evaluation after the models are fully trained. The numbers are provided in the supplementary material to facilitate future comparisons.

we evaluate the quality and diversity of the predicted motions. Note, these metrics should be considered together, since each one taken separately does not provide a complete picture of how well a model can predict multiple plausible future motions. For example, a model can generate diverse but unnatural motions, or, conversely, realistic but identical motions. To evaluate quality, as discussed in Section 5.3.4, we use a recurrent binary classifier whose task...
is to determine whether a sample comes from the ground-truth data or was generated by the model. We train such a classifier for each method, using 25K samples generated at different training steps together with 25K real samples, forming a binary dataset of 50K motions for each method. To evaluate diversity, as discussed in Section 5.3.4 we compute the mean Euclidean distance from each motion to all other $K-1$ motions when generating $K = 50$ motions. To compute IS, we trained an action classifier [Li et al., 2018b] with 50K real motions. We then compute the IS for $K = 50$ samples per condition for 50 different conditions. We followed Section 5.3.4 to report IS. Furthermore, we also performed a human evaluation to measure the quality of the motions generated by each method. To this end, we asked eight users to rate the quality of 50 motions generated by each method, for a total of 200 motions. The ratings were defined on a scale of 1-5, 1 representing a low-quality motion and 5 a high-quality, realistic one. We then scaled the values to the range 0-50 to make them comparable with those of the binary classifier.

### 5.4.1 Comparison to the State-of-the-Art

In this section, we quantitatively compare our approach to the state-of-the-art stochastic motion prediction techniques in terms of approximate ELBO, Diversity, Quality, and IS on a held-out test set, as well as the training KL term at convergence. Table 5.1 shows the results on the Human3.6M and CMU Mocap datasets.

These results show that Mix-and-Match is highly capable of learning the variation in human motion while maintaining a good motion quality. This is shown by IS, Diversity, and Quality metrics, which should be considered together. It is also evidenced by the low reconstruction loss and higher KL term on the test set. The training KL term at convergence also shows that, in Mix-and-Match, the posterior does not collapse to the prior distribution, i.e., the model does not ignore the latent variable. While the MSE of our approach is slightly higher than that of Yan et al., [Yan et al., 2018] on Human3.6M and Barsoum et al., [Barsoum et al., 2018] on the CMU Mocap dataset, we effectively exploit the latent variables, as demonstrated by the KL term on the test set, the IS and diversity metric and the qualitative results provided in Fig. 5.4 and in the supplementary material. As evidenced by the examples of diverse motions generated by our model in Fig. 5.4, given a single observation, Mix-and-Match is able to generate diverse, but natural motions.

### 5.4.2 Analysis on Diversity and Quality

To provide a deeper understanding of our approach, we evaluate different aspects of Mix-and-Match. All these experiments were done on Human3.6M. In the following, we first analyze the diversity in the hidden state space, i.e., the first part of the model where variation is imposed. We then evaluate the quality and diversity of prediction when tested at different stages of the training. We also perform a human evaluation on the quality of the generated motions, comparing it with our inexpensive, automatic quality metric. Finally, we compare Mix-and-Match with other stochastic techniques in terms of sampling error (S-MSE), i.e., by computing

---

2See the video of our results in the supplementary material.
the error of the best of \( K \) generated motions given the ground-truth one. More experiments and visualizations are provided in the supplementary material.

**Diversity in Hidden State Space.** In Fig. 5.5, we plot the diversity of the representations used as input to the RNN decoders of [Yan et al., 2018] and [Barsoum et al., 2018], two state-of-the-art methods that are closest in spirit to our approach. Here, diversity is measured as the average pairwise distance across the \( K = 50 \) representations produced for a single series of observations. We report the mean diversity over 50 samples and the corresponding standard deviation. As can be seen from the figure, the diversity of [Yan et al., 2018] and [Barsoum et al., 2018] decreases as training progresses, thus supporting our observation that these models learn to ignore the perturbations. As evidenced by the black curve, which shows an increasing diversity as training progresses, our approach produces not only high-quality predictions but also truly diverse ones. The gradual but steady increase in diversity of our approach is due to our curriculum learning strategy described in Section 5.3.3. Without it, training is less stable, with large diversity variations.

**Table 5.2:** Quantitative comparison of the S-MSE against stochastic baselines for four actions of the Human3.6M dataset.
Table 5.3: Comparison against deterministic motion prediction techniques for four actions of the Human3.6M dataset.
Diversity and Quality in Motion Space. Now, we thoroughly compare our approach with state-of-the-art stochastic motion prediction models in terms of quality and diversity. The results of the metrics of Section 5.3.4 are provided in Fig. 5.6 (Left and Middle) and those of the human evaluation in Fig. 5.6 (Right). Below, we analyze the results of the different models.

As can be seen from Fig. 5.6, [Yan et al., 2018] tends to ignore the random variable $z$, thus ignoring the root of variation. As a consequence, it achieves a low diversity, much lower than ours, but produces samples of high quality, albeit almost identical, which is also shown in qualitatively in Fig. 3 of the supplementary material. We empirically observed that the magnitude of the weights acting on $z$ to be orders of magnitude smaller than that of acting on the condition, 0.008 versus 232.85 respectively. Note that this decrease in diversity occurs after 16K iterations, indicating that the model takes time to identify the part of the hidden state that contains the randomness. Nevertheless, at iteration 16K, prediction quality is low, and thus one could not simply stop training at this stage. Note that the lack of diversity of [Yan et al., 2018] is also evidenced by Fig. 5.5. As can be verified in Fig. 5.6 (Right), where [Yan et al., 2018] appears in a region of high quality but low diversity, the results of human evaluation match those of our classifier-based quality metric.

Fig. 5.6 also evidences the limited diversity of the motions produced by [Barsoum et al., 2018] despite its use of random noise during inference. Note that the authors of [Barsoum et al., 2018] mentioned in their chapter that the random noise was added to the hidden state. Only by studying their publicly available code did we understand the precise way this combination was done. In fact, the addition relies on a parametric, linear transformation of the noise vector. That is, the perturbed hidden state is obtained as $h_{\text{perturbed}} = h_{\text{original}} + W_{z \rightarrow h} z$. Because the parameters $W_{z \rightarrow h}$ are learned, the model has the flexibility to ignore $z$ (the magnitude of $W_{z \rightarrow h}$ is in the order of $O(1e^{-3})$), which causes the behavior observed in Figs. 5.6 and 5.5. Note that the authors of [Barsoum et al., 2018] acknowledged that, despite their best efforts, they noticed very little variations between predictions obtained with different $z$ values. By depicting [Barsoum et al., 2018] in a region of high quality but low diversity, the human evaluation results in Fig. 5.6 (Right) again match those of our classifier-based quality metric.

As can be seen in Fig. 5.6 (Left and Middle), [Walker et al., 2017] produces motions with higher diversity than [Barsoum et al., 2018; Yan et al., 2018], but of much lower quality. The main reason behind this is that the random vectors that are concatenated to the poses at each time-step are sampled independently of each other, which translates to discontinuities in the generated motions. Human evaluation in Fig. 5.6 (Right) further confirms that [Walker et al., 2017]'s results lie in a low-quality, medium-diversity region.

The success of our approach is confirmed by Fig. 5.6 (Left and Middle). Our model generates diverse motions, even after a long training time, and the quality of these motions is high. While this quality is slightly lower than that of [Barsoum et al., 2018; Yan et al., 2018] when looking at our classifier-based metric, it is rated higher by IS and humans, as can be verified from Fig. 5.6 (Right) and Table 5.1. Altogether, these results confirm the ability of our approach to generate highly diverse yet realistic motions.

---

3 https://github.com/ebarsoum/hpgan
Evaluating the Sampling Error. We now quantitatively compare our approach with other stochastic baselines in terms of sampling error (aka S-MSE). To this end, we follow the evaluation setting of deterministic motion prediction (as in [Pragkiadaki et al., 2015; Pavllo et al., 2019; 2018; Martinez et al., 2017; Gui et al., 2018a]) which allows further comparisons to deterministic baselines. We report the standard metric, i.e., the Euclidean distance between the generated and ground-truth Euler angles (aka MAE). To evaluate this metric for our method and the stochastic motion prediction models, which generate multiple, diverse predictions, we make use of the best sample among the $K$ generated ones with $K = 50$ for the stochastic baselines and for our approach. This evaluation procedure aims to show that, among the $K$ generated motions, at least one is close to the ground truth. As shown in Table 5.2, by providing higher diversity, our approach outperforms the baselines. Similarly, in Table 5.3, we compare the best of $K = 50$ sampled motions for our approach with the deterministic motion prediction techniques. Note that the goal of this experiment is not to provide a fair comparison to deterministic models, but to show that, among the diverse set of motions generated by our model, there exists at least one motion that is very close to the ground-truth one. The point of bringing the MAE of other deterministic methods, is to show how good deterministic models, with sophisticated architectures and complicated loss functions, perform on this task.

5.5 Conclusion

In this chapter, we have proposed an effective way of perturbing the hidden state of an RNN such that it becomes capable of learning the multiple modes of human motions. Our evaluation of quality and diversity, based on both new quantitative metrics and human judgment, have evidenced that our approach outperforms existing stochastic methods. Generating diverse plausible motions given limited observations has many applications, especially when the motions are generated in an action-agnostic manner, as done here. For instance, our model can be used for human action forecasting [Rodriguez et al., 2018; Aliakbarian et al., 2016; Shi et al., 2018], as well as the methods introduced in Chapter 3 and Chapter 4, where one seeks to anticipate the action as early as possible, or for motion inpainting, where, given partial observations, one aims to generate multiple in-between solutions. Although Mix-and-Match can successfully encourage diversity in the generate motions, it does not account for the fact that all generated motions should convey the semantic information exist in the past observation. This could be particularly important in certain applications, such as automatic animation generation. In the next chapter, we introduce another VAE-based framework that not only encourages diversity, but also encourages generating semantically plausible motions.
Better Motion Generation via Variational Autoencoders with Learned Conditional Priors

Following previous chapter, we continue studying the problem of anticipating continuous representations of a stochastic future, with the focus on diverse human motion prediction. Similar to previous chapter, in this context, a popular approach consists of using Conditional Variational Autoencoders (CVAEs). In this chapter, we identify a major weakness of the CVAE frameworks used in existing techniques: During training, they rely on a global prior on the latent variable, thus making it independent of the conditioning signal. At inference, this translates to sampling latent variables that do not match the given CVAE condition. In this chapter, we directly address this by conditioning the sampling of the latent variable on the CVAE condition, thus encouraging it to carry relevant information. Our experiments demonstrate that our approach not only yields samples of higher quality while retaining the semantic information contained in the observed 3D pose sequence, but also helps the model to avoid the posterior collapse, a known problem of VAEs with expressive decoders. We additionally show the generality of our approach by using it for diverse image captioning.

6.1 Introduction

Human motion prediction is the task of forecasting plausible 3D human motion continuation(s) given a sequence of past 3D human poses. To address this problem, prior work mostly relies on recurrent encoder-decoder architectures, where the encoder processes the observed motion, and the decoder generates a single estimated future trajectory given the encoded representation of the past [Martinez et al., 2017; Gui et al., 2018a; Walker et al., 2017; Kundu et al., 2018; Barsoum et al., 2018; Pavllo et al., 2019; 2018; Mao et al., 2019b]. While this approach yields valid future motion, it tends to ignore the fact that human motion is stochastic in nature; given one single observation, multiple diverse continuations of the motion are likely and plausible. The lack of stochasticity of these encoder-decoder methods ensues from the fact that both the network operations and the sequences in the training dataset are deterministic. In this chapter, we introduce an approach to modeling this stochasticity by learning multiple modes of human motion.
Figure 6.1: Training and inference for a standard CVAE. (Left) During the training phase, the encoder takes as input the combination of the data and the corresponding condition and compresses it into the latent space. The decoder then samples a latent variable from the approximate posterior computed by the encoder, combines the latent variable with the conditioning signal, and reconstructs the data. The approximate posterior distribution of all training samples, i.e., for all conditioning signals, is encouraged to match the prior distribution. (Right) During inference, the decoder samples different latent variables from the prior distribution, combines them with the condition, and generates samples that all respect the conditioning signal. However, there is no guarantee that the latent variable is sampled from the region of the prior that corresponds to the given condition.

Recent attempts that account for human motion stochasticity rely on combining a random vector with an encoding of the observed pose sequence [Bütepage et al., 2018; Yan et al., 2018; Barsoum et al., 2018; Walker et al., 2017; Kundu et al., 2018; Lin and Amer, 2018; Aliakbarian et al., 2020b]. In particular, the state-of-the-art approaches to diverse human motion prediction [Aliakbarian et al., 2020b; Yan et al., 2018] make use of conditional variational auto-encoders (CVAEs). In this chapter, we argue that standard CVAEs are ill-suited to this task for the following reason.

In essence, VAEs utilize neural networks to learn the distribution of the data. To this end, VAEs first learn to generate a latent variable $z$ given the data $x$, i.e., approximate the posterior distribution $q_\phi(z|x)$, where $\phi$ are the parameters of a neural network, the encoder, whose goal is to model the variation of the data. From this latent random variable $z$, VAEs then generate a new sample $x$ by learning $p_\theta(x|z)$, where $\theta$ denotes the parameters of another neural network, the decoder, whose goal is to maximize the log likelihood of the data. These two networks, i.e., the encoder and the decoder, are trained jointly, using a prior over the latent variable. By using a variational approximation of the posterior, training translates to maximizing the variational lower bound of the log likelihood with respect to the parameters $\phi$ and $\theta$, given by

$$\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[ \log p_\theta(x|z) \right] - KL \left( q_\phi(z|x) \parallel p(z) \right), \quad (6.1)$$

where the second term on the right hand side encodes the KL divergence between the posterior $q_\phi(z|x)$ and a chosen prior distribution $p(z)$. As an extension to VAEs, CVAEs use auxiliary information, i.e., the conditioning variable or observation, to generate the data $x$. In the stan-
Related Work

In the standard setting, both the encoder and the decoder are conditioned on the conditioning variable \( c \). That is, the encoder becomes \( q_\phi(z|x,c) \) and the decoder \( p_\theta(x|z,c) \). Then, in theory, the objective of the model should become

\[
\log p_\theta(x|c) \geq \mathbb{E}_{q_\phi(z|x,c)} \left[ \log p_\theta(x,z,c) \right] - \text{KL} \left( q_\phi(z|x,c) \parallel p(z|c) \right).
\] (6.2)

In practice, however, the prior distribution of the latent variable is still assumed to be independent of \( c \), i.e., \( p(z \mid c) = p(z) \). As illustrated by Fig. 6.1 at test time, this translates to sampling a latent variable from a region of the prior that is unlikely to be highly correlated with the (observed) condition.

In this chapter, we overcome this limitation by explicitly making the sampling of the latent variable depend on the condition. In other words, instead of using \( p(z) \) as prior distribution, we truly use \( p(z \mid c) \). This not only respects the theory behind the design of CVAEs, but, as we empirically demonstrate, leads to generating motions of higher quality, that preserve the context of the conditioning signal, i.e., the observed past motion. To achieve this, we develop a CVAE architecture that learns a distribution not only of the latent variable but also of the conditioning one. We then use this distribution as a prior over the latent variable, making its sampling explicitly dependent on the condition. As such, we name our method \textbf{LCP–VAE}, for \textit{Learned Conditional Prior}.

Our experiments show that not only does \textbf{LCP–VAE} yield a much wider variety of plausible samples than state-of-the-art stochastic motion prediction methods, but it also preserves the semantic information of the condition, such as the type of action performed by the person, without explicitly exploiting this information. We also show that, by unifying latent variable sampling and conditioning, we can mitigate the posterior collapse problem, a well-known issue for VAEs with expressive decoders [Bowman et al., 2015; Yang et al., 2017; Kim et al., 2018; Gulrajani et al., 2016; Liu et al., 2019; Semeniuta et al., 2017; Zhao et al., 2017b; Tolstikhin et al., 2017; Chen et al., 2016; Alemi et al., 2017; He et al., 2019; Li et al., 2019; Goyal et al., 2017; Lucas and Verbeek, 2018; Dieng et al., 2018; van den Oord et al., 2017; Guu et al., 2018; Xu and Durrett, 2018; Davidson et al., 2018; Razavi et al., 2019], but unexplored for CVAEs. For more detail on this problem, we refer the reader to Appendix A. Finally, we show that our approach generalizes to the task of diverse image captioning.

6.2 Related Work

Because of space limitation, in this section, we focus on the human motion prediction literature. We nonetheless review of the literature on image captioning work and posterior collapse in Appendix A.

Most motion prediction methods are based on \textit{deterministic} models [Pavllo et al., 2018; Mao et al., 2019b; Pavllo et al., 2019; Gu et al., 2018a; Jain et al., 2016d; Martinez et al., 2017; Gu et al., 2018b; Fragkiadaki et al., 2015; Ghosh et al., 2017], casting motion prediction as a regression task where only one outcome is possible given the observation. While this may produce accurate predictions, it fails to reflect the stochastic nature of human motion, where multiple futures can be highly likely for a single given series of observations. Modeling stochasticity is the topic of this chapter, and we therefore focus the discussion below on the
other methods that have attempted to do so.

The general trend to incorporate variations in the predicted motions consists of combining information about the observed pose sequence with a random vector. In this context, two types of approaches have been studied: The techniques that directly incorporate the random vector into the RNN decoder and those that make use of an additional CVAE. In the first class of methods, [Lin and Amer, 2018] samples a random vector \( z_t \sim \mathcal{N}(0, I) \) at each time step and adds it to the pose input of the RNN decoder. By relying on different random vectors at each time step, however, this strategy is prone to generating discontinuous motions. To overcome this, [Kundu et al., 2018] makes use of a single random vector to generate the entire sequence. As we will show in our experiments, by relying on concatenation, these two methods contain parameters that are specific to the random vector, and thus give the model the flexibility to ignore this information. In [Barsoum et al., 2018], instead of using concatenation, the random vector is added to the hidden state produced by the RNN encoder. While addition prevents having parameters that are specific to the random vector, this vector is first transformed by multiplication with a learnable parameter matrix, and thus can again be zeroed out so as to remove the source of diversity, as observed in our experiments.

The second category of stochastic methods introduce an additional CVAE between the RNN encoder and decoder. In this context, [Walker et al., 2017] proposes to directly use the pose as conditioning variable. As will be shown in our experiments, while this approach is able to maintain some degree of diversity, albeit less than ours, it yields motions of lower quality because of its use of independent random vectors at each time step. Instead of perturbing the pose, [Yan et al., 2018] uses the RNN decoder hidden state as conditioning variable in the CVAE, concatenating it with the random vector. While this approach generates high-quality motions, it suffers from the fact that the CVAE decoder gives the model the flexibility to ignore the random vector, which therefore yields low-diversity outputs. To overcome this, [Aliakbarian et al., 2020b] perturbs the hidden states via a stochastic Mix-and-Match operation instead of concatenation. Through such a perturbation, the decoder is not able to decouple the noise and the condition. However, since the perturbation is not learned and is a non-parametric operation, the quality and the context of the generated motion are inferior to those obtained with our approach. More importantly, all of the above-mentioned CVAE-based approaches use priors that are independent of the condition. We will show in our experiments that such designs are ill-suited for human motion prediction. By contrast, our approach uses a conditional prior and is thus able to generate diverse motions of higher quality, carrying the contextual information of the conditioning signal.

### 6.3 Unifying Sampling Latent Variable and Conditioning

In this section, we introduce our approach as a general framework with a new conditioning scheme for CVAEs that is capable of generating diverse and plausible samples, where the latent variables are sampled from an appropriate region of the prior distribution. In essence, our framework consists of two autoencoders, one acting on the conditioning signal and the other on the samples we wish to model. The latent representation of the condition then serves as a conditioning variable to generate samples from a learned distribution.

As discussed above, we are interested in problems that are stochastic in nature; given
§6.3 Unifying Sampling Latent Variable and Conditioning

Figure 6.2: Illustration of a CVAE architecture and an LCP–VAE architecture. **(Top)** In a standard CVAE architecture, the latent variable $z$ is sampled from a standard Normal distribution, independently of the condition. This leads to the reparameterization trick illustrated on the right. In this case, the approximate posterior is normally distributed. **(Bottom)** In an LCP–VAE, the sampling of $z$ is conditioned on the CVAE condition via the latent variable $z_c$. Specifically, the posterior distribution of the condition acts as prior on the data posterior. This corresponds to the extended reparameterization trick illustrated on the right. Note that the approximate posterior is not normally distributed anymore.

one condition, multiple plausible and natural samples are likely. However, for complicated tasks such as motion prediction, the training data is typically insufficiently sampled, in that, for any given condition, the dataset contains only a single sample, in effect making the data appear deterministic. For instance, in motion prediction, we never observed twice the same past motion with two different future ones. Moreover, for such problems, the condition provides the core signal to generate a good sample, even in a deterministic model. Therefore, it is highly likely that a CVAE trained for this task learns to ignore the latent variable and rely only on the condition to produce its output [Aliakbarian et al., 2020b; Yan et al., 2018]. This relates to the posterior collapse problem in strongly conditioned VAEs [Aliakbarian et al., 2020b], as discussed in more detail in Appendix A. Below, we address this by forcing the sampling of the random latent variable to depend on the conditioning one. By making this dependency explicit, we (1) sample an informative latent variable given the condition, and thus generate a sample of higher quality, and (2) prevent the network from ignoring the latent variable in the presence of a strong condition, thus enabling it to generate diverse outputs.

Note that conditioning the VAE encoder via standard strategies, e.g., concatenation, is perfectly fine, since the two inputs to the encoder, i.e., the data and the condition, are deterministic and useful to compress the sample into the latent space. However, conditioning the VAE decoder requires special care, which is what we focus on below.
6.3.1 Stochastically Conditioning the Decoder

We propose to make the sampling of the latent variable from the prior/posterior distribution explicitly depend on the condition instead of treating these two variables as independent. To this end, we first learn the distribution of the condition via a simple VAE, which we refer to as CS-VAE because this VAE acts on the conditioning signal. The goal of CS-VAE is to reconstruct the condition, e.g., the observed past motion, given its latent representation. We take the prior of CS-VAE as a standard Normal distribution \( N(0, I) \). Following Kingma and Welling [Kingma and Welling, 2013], this allows us to approximate the CS-VAE posterior with another sample from a Normal distribution \( \epsilon \sim N(0, I) \) via the reparametrization trick 

\[
  z_c = \mu_c + \sigma_c \odot \epsilon ,
\]

where \( \mu_c \) and \( \sigma_c \) are the parameter vectors of the posterior distribution generated by the VAE encoder, and thus \( z_c \sim N(\mu_c, \text{diag}(\sigma_c)^2) \).

Following the same strategy for the data VAE translates to treating the conditioning and the data latent variables independently, which we seek to avoid. Therefore, as illustrated in Fig. 6.2 (Bottom), we instead define the LCP-VAE posterior not as directly normally distributed but conditioned on the posterior of CS-VAE. To this end, we extend the standard reparameterization trick as 

\[
  z = \mu + \sigma \odot z_c \\
  = \mu + \sigma \odot (\mu_c + \sigma_c \odot \epsilon) \\
  = (\mu + \sigma \odot \mu_c) + (\sigma \odot \sigma_c) \odot \epsilon ,
\]

where \( z_c \) comes from Eq. 6.3 and \( \mu \) and \( \sigma \) are the parameter vectors generated by the LCP-VAE encoder. In fact, \( z_c \) in Eq. 6.3 is a sample from the scaled and translated version of \( N(0, I) \) given \( \mu_c \) and \( \sigma_c \), and \( z \) in Eq. 6.4 is a sample from the scaled and translated version of \( N(\mu_c, \text{diag}(\sigma_c)^2) \) given \( \mu \) and \( \sigma \). Since we have access to the observations during both training and testing, we always sample \( z_c \) from the condition posterior. As \( z \) is sampled given \( z_c \), one expects the latent variable \( z \) to carry information about the strong condition, and thus a sample generated from \( z \) to correspond to a plausible sample given the condition. This extended reparameterization trick lets us sample one single informative latent variable that contains information about both the data and the conditioning signal. This further allows us to avoid conditioning the LCP-VAE decoder by concatenating the latent variable with a deterministic representation of the condition. As will be shown in our experiments, and in more detail in Appendix A, our sampling strategy not only yields higher-quality motions that retain the semantic information contained in the conditioning signal, but also helps the model to avoid posterior collapse. However, it changes the variational family of the LCP-VAE posterior. In fact, the posterior is no longer \( N(\mu, \text{diag}(\sigma)^2) \), but a Gaussian distribution with mean \( \mu + \sigma \odot \mu_c \) and covariance matrix \( \text{diag}(\sigma \odot \sigma_c)^2 \). This will be accounted for when designing the KL divergence loss discussed below.
Figure 6.3: Inference procedure in an LCP-VAE and a CVAE. (Left) In a standard CVAE, at inference time, since the prior distribution is independent of the conditioning signal, we are unlikely to sample a latent variable that corresponds to the region of the latent space that is related to the observed condition. (Right) In an LCP-VAE, at inference time, since we have access to the conditioning signal, we use the CS-VAE encoder to approximate the posterior of each condition. To generate a sample given a condition, the LCP-VAE decoder then samples a latent variable from the posterior of its condition, instead of using a general prior distribution as in CVAEs, and generate a sample. For instance, \( \hat{x}_j \) and \( \hat{x}_l \) are conditioned on \( C_4 \) and their corresponding latent variables are sampled from a very similar region. By contrast, \( \hat{x}_k \) is generated by a latent variable sampled from a completely different region, i.e., the region corresponding to the approximate posterior of \( C_5 \).

6.3.2 Learning

To learn the parameters of our model, we rely on the availability of a dataset \( D = \{X_1, X_2, ..., X_N\} \) containing \( N \) training samples \( X_i \). Each training sample is a pair of condition and data sample. For CS-VAE, which learns the distribution of the condition, we define the loss as the KL divergence between its posterior and the standard Gaussian prior, that is,

\[
\mathcal{L}_{\text{prior}}^{\text{CS-VAE}} = KL \left( \mathcal{N}(\mu_c, \text{diag}(\sigma_c)^2) \middle| \mathcal{N}(0, I) \right) = -\frac{1}{2} \sum_{j=1}^{d} \left( 1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2 \right), \tag{6.5}
\]

where \( d \) is the dimension of the latent variable \( z_c \). By contrast, for LCP-VAE, we define the loss as the KL divergence between the posterior of LCP-VAE and the posterior of CS-VAE, i.e., of the condition. To this end, we freeze the weights of CS-VAE before computing the KL divergence, since we do not want to move the posterior of the condition but that of the data. The KL divergence is then computed as the divergence between two multivariate Normal distributions, encoded by their mean vectors and covariance matrices as

\[
\mathcal{L}_{\text{prior}}^{\text{LCP-VAE}} = KL \left( \mathcal{N}(\mu + \sigma \odot \mu_c, \text{diag}(\sigma \odot \sigma_c)^2) \middle| \mathcal{N}(\mu_c, \text{diag}(\sigma_c)^2) \right). \tag{6.6}
\]
Let $\Sigma = \text{diag}(\sigma^2)$, $\Sigma_c = \text{diag}(\sigma_c^2)$, $d$ be the dimensionality of the latent space and $\text{tr}\{\cdot\}$ the trace of a square matrix. The loss in Eq. 6.6 can be written as\footnote{See Appendix A for more detail on the KL divergence between two multivariate Gaussians and the derivation of Eq. 6.7.}

$$L_{\text{prior}}^{\text{LCP-VAE}} = -\frac{1}{2} \left[ \log \frac{1}{|\Sigma|} - d + \text{tr}\{\Sigma\} + (\mu_c - (\mu + \Sigma \mu_c))^T \Sigma_c^{-1} (\mu_c - (\mu + \Sigma \mu_c)) \right].$$

(6.7)

After computing the loss in Eq. 6.7, we unfreeze CS-VAE and update it with its previous gradient. Trying to match the posterior of LCP-VAE to that of CS-VAE allows us to effectively use our extended reparameterization trick in Eq. 6.4. Furthermore, we use the standard reconstruction loss for both CS-VAE and LCP-VAE, thus minimizing the mean squared error (MSE) in the case of human motion prediction and the negative log-likelihood (NLL) in the case of image captioning.

We refer to the reconstruction losses as $L_{\text{rec}}^{\text{CS-VAE}}$ and $L_{\text{rec}}^{\text{LCP-VAE}}$ for CS-VAE and LCP-VAE, respectively. Thus, our complete loss is

$$L = \lambda (L_{\text{prior}}^{\text{CS-VAE}} + L_{\text{prior}}^{\text{LCP-VAE}}) + L_{\text{rec}}^{\text{CS-VAE}} + L_{\text{rec}}^{\text{LCP-VAE}}.$$ 

(6.8)

In practice, since the nature of our data is sequential (e.g., sequence of human poses in human motion prediction and sequence of words in image captioning), the LCP-VAE encoder is a recurrent model. Thus, we weigh the KL divergence terms by a function $\lambda$ corresponding to the KL annealing weight of [Bowman et al., 2015]. We start from $\lambda = 0$, forcing the model to encode as much information in $z$ as possible, and gradually increase it to $\lambda = 1$ during training, following a logistic curve. We then continue training with $\lambda = 1$.

In short, our method can be interpreted as a simple yet effective framework (designed for CVAEs) for altering the variational family of the posterior such that (1) a latent variable from this posterior distribution is explicitly sampled given the condition, both during training and inference (as illustrated in Fig. 6.3), and (2) the model is much less likely to suffer from posterior collapse because the mismatch between the posterior and prior distributions makes it harder for learning to drive the KL divergence of Eq. 6.7 towards zero.

### 6.4 Experiments

In this chapter, we mainly focus on stochastic human motion prediction, where the goal is to generate diverse and plausible continuations of given past observations. Additionally, to show that our LCP-VAE generalizes to other domains, we tackle the problem of stochastic image captioning, where, given an image representation, the task is to generate diverse yet related captions.
§6.4 Experiments

Figure 6.4: Qualitative evaluation of the diversity in human motion. The first row depicts the ground-truth motion. The first six poses of each row correspond to the observation (the condition) and the rest are sampled from our model. Each row is a randomly sampled motion (not cherry picked). Note that all sampled motions are natural, with a smooth transition from the observed poses to the generated ones. Diversity increases with the length of the generated sequence.

6.4.1 Diverse Human Motion Prediction

Datasets. To evaluate the effectiveness of our approach on the task of stochastic human motion prediction, we use the Human3.6M [Ionescu et al., 2014] and CMU MoCap datasets, two large publicly-available motion capture (mocap) datasets. Human3.6M comprises more than 800 long indoor motion sequences performed by 11 subjects, leading to 3.6M frames. Each frame contains a person annotated with 3D joint positions and rotation matrices for all 32 joints. In our experiments, for our approach and the replicated VAE-based baselines, we represent each joint in 4D quaternion space. We follow the standard preprocessing and evaluation settings used in [Martinez et al., 2017; Gui et al., 2018a; Pavllo et al., 2018; Jain et al., 2016d].

The CMU MoCap dataset is another large-scale motion capture dataset covering diverse human activities, such as jumping, running, walking, and playing basketball. Each frame contains a person annotated with 3D joint rotation matrices for all 38 joints. As for Human3.6M, and following standard practice [Mao et al., 2019b; Li et al., 2018a], we represent each joint in 4D quaternion space. We also evaluate our approach on the real-world Penn Action dataset [Zhang et al., 2013], which contains 2326 sequences of 15 different actions, where for each person, 13 joints are annotated in 2D space. The results on Penn Action are provided in Appendix A.

Available at http://mocap.cs.cmu.edu/.
Table 6.1: Comparison of LCP-VAE with the stochastic motion prediction baselines on Human3.6M (top) and CMU MoCap (bottom).

| Method            | ELBO (KL) (Reconstructed) | Diversity (Sampled) | Quality (Sampled) | Context (Sampled) | Training KL (Reconstructed) |
|-------------------|---------------------------|---------------------|-------------------|-------------------|-----------------------------|
| MT-VAE [Yan et al., 2018] | 0.51 (0.06)               | 0.26                | 0.45              | 0.42              | 0.08                        |
| Pose-Knows [Walker et al., 2017] | 2.08 (N/A)                | 1.70                | 0.13              | 0.08              | N/A                         |
| HP-GAN [Barsoum et al., 2018] | 0.61 (N/A)                | 0.48                | 0.47              | 0.35              | N/A                         |
| Mix-and-Match [Aliakbarian et al., 2020b] | 0.55 (2.03)               | 3.52                | 0.42              | 0.37              | 1.98                        |
| LCP-VAE           | 0.41 (8.07)               | 3.12                | 0.48              | 0.54              | 6.93                        |

Table 6.2: Comparison of the generated motions with the ground-truth future motions in terms of context. The gap between the performance of the state-of-the-art pose-based action classifier [Li et al., 2018b] with and without true future motions is 0.22 / 0.54. Using our predictions, this gap decreases to 0.06 / 0.08, showing that our predictions reflect the class label. H3.6M / CMU represent Human3.6M and CMU MoCap, respectively.

| Setting | Obs.       | Future Motion | Context (H3.6M / CMU) |
|---------|------------|---------------|-----------------------|
| Lower bound | GT         | Zero velocity | 0.38 / 0.42           |
| Upper bound (GT poses as future motion) | GT         | GT            | 0.60 / 0.96           |
| Ours (sampled motions as future motion) | GT         | Sampled from LCP-VAE | 0.54 / 0.88           |

Evaluation Metrics. To quantitatively evaluate our approach and other stochastic motion prediction baselines [Yan et al., 2018, Barsoum et al., 2018, Walker et al., 2017, Aliakbarian et al., 2020b], we report the reconstruction error, commonly referred to as ELBO, along with the KL-divergence on the held-out test set. Additionally, we report quality [Aliakbarian et al., 2020b] and diversity [Yang et al., 2019, Aliakbarian et al., 2020b, Yuan and Kitani, 2019] metrics, which should be considered together. Specifically, to measure the diversity of the motions generated by a stochastic model, we make use of the average distance between all pairs of the K motions generated from the same observation. To measure quality, we train a binary classifier [Aliakbarian et al., 2020b] to discriminate real (ground-truth) samples from fake (generated) ones. The accuracy of this classifier on the test set is inversely proportional to the quality of the generated motions. Furthermore, we report a context metric measured as the performance of a strong action classifier [Li et al., 2018b] trained on ground-truth motions. Specifically, the classifier is tested on each of the K motions generated from each observation. For N observations and K continuations per observation, the accuracy is measured by computing the argmax over each prediction’s probability vector, and we report context as the mean class accuracy on the K × N motions. Finally, we report the training KL at convergence to show that no posterior collapse occurred. For all metrics, we use K = 50 motions per test observation. For all experiments, we use 16 frames (i.e., 640ms) as observation to generate the
§6.4 Experiments

Figure 6.5: t-SNE plots of the posterior mean for 3750 test motions. With MT-VAE [Yan et al., 2018], all classes are mixed, suggesting that the latent variable carries little information about the motions. By contrast, our condition-dependent sampling allows LCP-VAE to better preserve context. Note that some actions, such as discussion and directions, are very hard to identify and are thus spread over other actions. Others, such as walking, walking with dog, and walking together, or sitting and sitting down overlap due to their similarity.

Evaluating Stochasticity. In Table 6.1, we compare our approach (whose detailed architecture is described in Appendix A) with the state-of-the-art stochastic motion prediction models [Yan et al., 2018; Aliakbarian et al., 2020b; Walker et al., 2017; Barsoum et al., 2018]. Note that one should consider the reported metrics jointly to truly evaluate a stochastic model. For instance, while MT-VAE [Yan et al., 2018] and HP-GAN [Barsoum et al., 2018] generate high-quality motions, they are not diverse. Conversely, while Pose-Knows [Walker et al., 2017] generates diverse motions, they are of low quality. By contrast, our approach generates both high quality and diverse motions. This is also the case of Mix-and-Match [Aliakbarian et al., 2020b], which, however, preserves much less context. In fact, none of the baselines effectively conveys the context of the observation to the generated motions. As shown in Table 6.2, the upper bound for context on Human3.6M is 0.60 (i.e., the classifier [Li et al., 2018b] performance given the ground-truth motions). Our approach yields a context of 0.54 when given only about 20% of the data. We observe a similar behavior on the CMU MoCap dataset, shown in Table 6.2. Altogether, as also supported by the qualitative results of Fig. 6.4 and in Appendix A, our approach yields diverse, high-quality and context-preserving predictions. This is further evidenced by the t-SNE [Maaten and Hinton, 2008] plots of Fig. 6.5 where different samples of various actions are better separated for our approach than for, e.g., MT-VAE [Yan et al., 2018]. For further discussion of the baselines, a deeper insight of their behavior under different evaluation metrics, and a discussion of the relation of each method’s performance to posterior collapse, we refer the reader to Appendix A.

Evaluating Sampling Quality. To further evaluate the sampling quality, we evaluate stochastic baselines using the standard mean angle error (MAE) metric in Euler space. To this end, we use the best of the $K = 50$ generated motions for each observation (referred to as S-MSE in [Yan et al., 2018]). A model that generates more diverse motions has higher chances of producing a motion close to the ground-truth one. As shown in Table 6.3, this is the case with our
Table 6.3: Comparison with the state-of-the-art stochastic motion prediction models for 4 actions of Human3.6M (all methods use the best of $K = 50$ sampled motions).

| Method          | Walking 80 | Walking 160 | Walking 320 | Walking 400 | Walking 560 | Walking 1000 | Eating 80 | Eating 160 | Eating 320 | Eating 400 | Eating 560 | Eating 1000 |
|-----------------|------------|-------------|-------------|-------------|-------------|-------------|-----------|-----------|-------------|-------------|-------------|-------------|
| MT-VAE [Yan et al., 2018] | 0.73 0.79 0.90 0.93 0.95 1.05 0.68 0.74 0.95 1.00 1.03 1.38 | | | | | | 0.68 0.74 0.95 1.00 1.03 1.38 |
| HP-GAN [Wارsom et al., 2018] | 0.61 0.62 0.71 0.79 0.83 1.07 0.53 0.67 0.79 0.88 0.97 1.12 | | | | | | 0.53 0.67 0.79 0.88 0.97 1.12 |
| Pose-Knows [Walker et al., 2017] | 0.56 0.66 0.98 1.05 1.28 1.60 0.44 0.60 0.71 0.84 1.05 1.54 | | | | | | 0.44 0.60 0.71 0.84 1.05 1.54 |
| Mix&Match [Aliakbarian et al., 2020b] | 0.33 0.48 0.56 0.58 0.64 0.68 0.23 0.34 0.41 0.50 0.61 0.91 | | | | | | 0.23 0.34 0.41 0.50 0.61 0.91 |
| LCP-VAE | 0.22 0.36 0.47 0.52 0.58 0.69 0.19 0.28 0.40 0.51 0.58 0.90 | | | | | | 0.19 0.28 0.40 0.51 0.58 0.90 |

Table 6.4: Comparison with the state-of-the-art deterministic models for 4 actions of Human3.6M.

| Method          | Walking 80 | Walking 160 | Walking 320 | Walking 400 | Walking 560 | Walking 1000 | Eating 80 | Eating 160 | Eating 320 | Eating 400 | Eating 560 | Eating 1000 |
|-----------------|------------|-------------|-------------|-------------|-------------|-------------|-----------|-----------|-------------|-------------|-------------|-------------|
| Zero Velocity   | 0.39 0.86 0.99 1.15 1.35 1.32 0.27 0.48 0.73 0.86 1.04 1.38 | | | | | | 0.48 0.73 0.86 1.04 1.38 |
| LSTM-3LR [Fragkiadaki et al., 2015] | 1.18 1.50 1.67 1.76 1.81 2.20 1.36 1.79 2.29 2.42 2.49 2.82 | | | | | | 1.36 1.79 2.29 2.42 2.49 2.82 |
| SRNN [Jain et al., 2016d] | 1.08 1.34 1.60 1.80 1.90 2.13 1.35 1.71 2.12 2.21 2.28 2.58 | | | | | | 1.35 1.71 2.12 2.21 2.28 2.58 |
| DAE-LSTM [Jain et al., 2017] | 1.00 1.11 1.39 1.48 1.55 1.39 1.31 1.49 1.86 1.89 1.97 2.01 | | | | | | 1.39 1.49 1.86 1.89 1.97 2.01 |
| GRU [Martinez et al., 2017] | 0.28 0.49 0.72 0.81 0.93 1.03 0.23 0.39 0.62 0.76 0.95 1.08 | | | | | | 0.39 0.62 0.76 0.95 1.08 |
| AGED [Gui et al., 2018a] | 0.22 0.36 0.55 0.67 0.78 0.91 0.17 0.28 0.51 0.64 0.86 0.93 | | | | | | 0.28 0.51 0.64 0.86 0.93 |
| DCT-GCN [Mao et al., 2019b] | 0.18 0.31 0.49 0.56 0.65 0.67 0.16 0.29 0.50 0.62 0.76 1.12 | | | | | | 0.29 0.50 0.62 0.76 1.12 |
| LCP-VAE ($z = \mu_C$) | 0.20 0.34 0.48 0.53 0.57 0.71 0.20 0.26 0.44 0.52 0.61 0.92 | | | | | | 0.20 0.44 0.52 0.61 0.92 |

approach and Mix-and-Match [Aliakbarian et al., 2020b], which both yield higher diversity. However, our approach performs better thanks to its context-preserving latent representation and its higher quality of the generated motions.

In Table 6.4 we compare our approach with the state-of-the-art deterministic motion prediction models [Martinez et al., 2017; Jain et al., 2016d; Gui et al., 2018b; Fragkiadaki et al., 2015; Gui et al., 2018a] using the MAE metric in Euler space. To have a fair comparison, we generate one motion per observation by setting the latent variable to the distribution mode, i.e., $z = \mu_C$. This allows us to generate a plausible motion without having access to the ground truth. To compare against the deterministic baselines, we follow the standard setting, and thus use 50 frames (i.e., 2sec) as observation to generate the next 25 frames (i.e., 1sec). Surprisingly, despite having a very simple motion decoder architecture (one-layer GRU network) with a very simple reconstruction loss function (MSE), this motion-from-mode strategy yields re-
Image Groundtruth and Baselines

Ground-truth Captions:
1. A big blue two level bus at a bus stop with people.
2. A blue and white double decker bus parked at a bus stop.
3. A blue bus parked on a curb where people are standing.
4. A blue double decker bus parked on the side of a road.
5. A bus pulls up to a station where people wait.

Caption from Autoregressive:
a blue bus is parked on the side of the road.

Caption from Conditional VAE:
a bus is parked on the side of the road.

Captions from LCP-VAE:
1. A large bus is going down the street next to a huge crowd of people.
2. A double decker bus is driving down the street.
3. A blue and white bus on a street next to a building.
4. A bus is driving down the street with a car on the opposite side of the road.
5. A bus is driving down the street with a group of people on the side of it.
6. A street with a lot of people on it and a street sign on it.
7. A group of people standing on a street near a bus.
8. A street sign on a pole with a bus in the background.

Caption from the LCP-VAE mode:
a double decker bus is driving down the street in pursuit of passengers.

Figure 6.6: Qualitative evaluation of the diversity in generated captions. While captions generated by our approach are diverse, they all describe the image properly. The caption from the mode also typically provides an accurate description.

| Model               | ELBO (KL) (Reconstructed) | Perplexity (Reconstructed) | Quality (Sampled) | Diversity (Sampled) | Context (Sampled) | Training KL (Reconstructed) |
|---------------------|---------------------------|---------------------------|-------------------|---------------------|-------------------|-----------------------------|
| Autoregressive      | 3.01 (N/A)                | 20.29                     | 0.40              | N/A                 | 0.46              | N/A                         |
| Conditional VAE     | 2.86 (0.00)               | 17.46                     | 0.39              | 0.00                | 0.44              | 0.00                         |
| LCP-VAE             | 0.21 (3.28)               | 1.23                      | 0.40              | 0.53                | 0.43              | 3.11                         |

6.4.2 Diverse Image Captioning

For the task of conditional text generation, we focus on stochastic image captioning. To demonstrate the effectiveness of our approach, we report results on the MSCOCO [Lin et al., 2014] captioning task, with the original train/test splits of 83K and 41K images, respectively. The MSCOCO dataset has five captions per image. However, we make it deterministic by removing four captions per image, yielding a Deterministic-MSCOCO captioning dataset. Note that the goal of this experiment is not to advance the state of the art in image captioning, but rather to explore the effectiveness of our approach on a different task where we have strong conditioning signal and an expressive decoder in the presence of a deterministic dataset. A brief review of the recent work on diverse text generation is given in Appendix A.

We compare LCP-VAE (with the architecture described in Appendix A) with a standard CVAE and with its autoregressive, non-variational counterpart. For quantitative evaluation, we report the ELBO (the negative log-likelihood), along with the KL-divergence and the Perplexity of the reconstructed captions on the held-out test set. We also quantitatively measure the diversity, the quality, and the context of sampled captions. To measure the context, we rely on the BLEU1 score, making sure that the sampled captions represent elements that appear in

Note that LCP-VAE is agnostic to the choice of data encoder/decoder architecture. We leave the use of more sophisticated architectures for future research.
the image. For CVAE and LCP-VAE, we compute the average BLEU1 score for $K = 50$ captions sampled per image and report the mean over the images. To measure diversity, we compute the BLEU4 score between every pair of $K = 50$ sampled captions per image. The smaller the BLEU4 is, the more diverse the captions are. The diversity metric is then 1-BLEU4, i.e., the higher the better. To measure quality, we use a metric similar to that in our human motion prediction experiments, obtained by training a binary classifier to discriminate real (ground-truth) captions from fake (generated) ones. The accuracy of this classifier on the test set is inversely proportional to the quality of the generated captions. We provide qualitative examples for all the methods in Fig. 6.6 and in Appendix A. As shown in Table 6.5, a CVAE learns to ignore the latent variable as it can minimize the caption reconstruction loss given solely the image representation. By doing so, all the generated captions at test time are identical, despite sampling multiple latent variables. This can be further seen in the ELBO and Perplexity of the reconstructed captions. We expect a model that takes as input the captions and the image to have a much lower reconstruction loss compared to the autoregressive baseline, which takes only the image as input. However, this is not the case with CVAE, indicating that the connection between the encoder and the decoder, i.e., the latent variable, does not carry essential information about the input caption. However, the quality of the generated sample is reasonably good. This is also evidenced by the qualitative evaluations in Appendix A. By contrast, LCP-VAE effectively handles this situation by unifying the sampling of the latent variable and the conditioning, leading to diverse but high quality captions, as reflected by the ELBO of our approach in Table 6.5 and the qualitative results in Appendix A. Additional quantitative evaluations and ablation studies for image captioning are provided in Appendix A.

6.5 Conclusion

In this chapter, we have studied the problem of conditionally generating diverse sequences with a focus on scenarios where the conditioning signal is strong enough such that an expressive decoder can generate plausible samples from it only. We have addressed this problem by forcing the sampling of the latent variable to depend on the conditioning one, which contrasts with standard CVAEs. By making this dependency explicit, the model receives a latent variable that carries information about the condition during both training and test time. This further prevents the network from ignoring the latent variable in the presence of a strong condition, thus enabling it to generate diverse outputs. To demonstrate the effectiveness of our approach, we have investigated two application domains: Stochastic human motion prediction and diverse image captioning. In both cases, our LCP-VAE model was able to generate diverse and plausible samples, as well as to retain contextual information, leading to semantically-meaningful predictions. We believe our approach will have great impact on practical applications such as pedestrian intention forecasting [Aliakbarian et al. 2018b], trajectory forecasting and human tracking [Saleh et al. 2020; Kosaraju et al. 2019]. In the future, we will also apply our approach to other problems that rely on strong conditions, such as image inpainting and super-resolution, for which only deterministic datasets are available.
Conclusion

In this thesis we have tackled the problem video anticipation by learning sequential representations. We have started from anticipating a discrete representation of a deterministic future, e.g., as in action anticipation and then move toward a more complex task of anticipating multiple plausible continuous representations of a stochastic task, e.g., human motion prediction.

The major contributions of this thesis are outlined below:

1. We propose a novel action anticipation framework (that can be also seen as an early recognition model). In particular, we introduce a novel loss that encourages making correct predictions very early. Our loss models the intuition that some actions, such as running and high jump, are highly ambiguous after seeing only the first few frames, and false positives should therefore not be penalized too strongly in the early stages. By contrast, we would like to predict a high probability for the correct class as early as possible, and thus penalize false negatives from the beginning of the sequence. Our experiments demonstrate that, for a given model, our new loss yields significantly higher accuracy than existing ones on the task of early prediction. We also propose a novel multi-stage Long Short Term Memory (MS-LSTM) architecture for action anticipation. This model effectively extracts and jointly exploits context- and action-aware features. This is in contrast to existing methods that typically extract either global representations for the entire image or video sequence, thus not focusing on the action itself, or localize the feature extraction process to the action itself via dense trajectories, optical flow or actionness, thus failing to exploit contextual information. This work has been published in ICCV 2017, Venice, Italy.

2. We improve and extend our previous contribution by focusing on driving scenarios, encompassing common the subproblems of anticipating ego car’s driver maneuvers, front car’s driver maneuver, accidents, violating or respecting traffic rules, and pedestrian intention, with a fixed, sensible set of sensors. To this end, we introduce the VIrtual ENvironment for Action Analysis (VIENA\textsuperscript{2}) dataset. Altogether, these subproblems encompass a total of 25 distinct action classes. VIENA\textsuperscript{2} is acquired using the GTA V video game. It contains more than 15K full HD, 5s long videos, corresponding to more than 600 samples per action class, acquired in various driving conditions, weathers, daytime, and environments. This amounts to more than 2.25M frames, each annotated with an action label. These videos are complemented by basic vehicle dynamics measure-
ments, reflecting well the type of information that one could have access to in practice. We then benchmark state-of-the-art action anticipation algorithms on VIENA2, and as another contribution, introduce a new multi-modal, LSTM-based architecture that generalizes out previous contribution to an arbitrary number of modalities, together with a new anticipation loss, which outperforms existing approaches in our driving anticipation scenarios. This work has been published in ACCV 2018, Perth, Australia.

3. For continuous, stochastic anticipation task, we address the problem of stochastic human motion prediction. As introduced earlier in this thesis, human motion prediction aims to forecast the sequence of future poses of a person given past observations of such poses. To achieve this, existing methods typically rely on recurrent neural networks (RNNs) that encode the person’s motion. While they predict reasonable motions, RNNs are deterministic models and thus cannot account for the highly stochastic nature of human motion; given the beginning of a sequence, multiple, diverse futures are plausible. To correctly model this, it is therefore critical to develop algorithms that can learn the multiple modes of human motion, even when presented with only deterministic training samples. We introduce an approach to effectively learn the stochasticity in human motion. At the heart of our approach lies the idea of Mix-and-Match perturbations: Instead of combining a noise vector with the conditioning variables in a deterministic manner (as usually done in standard practices), we randomly select and perturb a subset of these variables. By randomly changing this subset at every iteration, our strategy prevents training from identifying the root of variations and forces the model to take it into account in the generation process. This is a highly effective conditioning scheme in scenarios when (1) we are dealing with a deterministic dataset, i.e., one sample per condition, (2) the conditioning signal is very strong and representative, e.g., the sequence of past observations, and (3) the model has an expressive decoder that can generate a plausible sample given only the condition. We utilize Mix-and-Match by incorporating it into a recurrent encoder-decoder network with a conditional variational autoencoder (CVAE) block that learns to exploit the perturbations. Mix-and-Match then acts as the stochastic conditioning scheme instead of concatenation that usually appears in standard CVAEs. This work has been published in CVPR 2020, Seattle, Washington, USA.

4. In our previous contribution, we identified one limitation of a standard CVAEs when dealing deterministic datasets and strong conditioning signals. In this work, we further investigates this problem from a more theoretical point of view. Specifically, in this contribution we tackle the task of diverse sequence generation in which all the diversely generated sequences carry the same semantic as in the conditioning signal. We observe that in standard CVAE, conditioning and sampling the latent variable are two independent processes, leading to generating samples that are not necessarily carry all the contextual information about the condition. To address this, we propose to explicitly make the latent variables depend on the observations (the conditioning signal). To achieve this, we develop a CVAE architecture that learns a distribution not only of the latent variables, but also of the observations, the latter acting as prior on the former. By

---

\(^1\)Our dataset is publicly available at [https://sites.google.com/view/viena2-project/](https://sites.google.com/view/viena2-project/)
doing so, we change the variational family of the posterior distribution of the CVAE, thus, as a side effect, our approach can mitigate posterior collapse to some extent. This work will be submitted to CVPR 2021.

7.1 Future Work

In this thesis we focused on the task of video anticipation. Although there has been great progress in recent years to address the task of action anticipation, diverse human motion prediction remained a relatively less studied problem. Below, we list some potential future direction based on our research, mostly focusing on diverse sequence generation:

- Incorporating sequential latent variables: In our proposed frameworks, we use a single latent variable to represent a sequence, e.g., human motion or an image caption. However, a more expressive alternative could be a sequential latent variable, considering a latent variable for the representations at each time-step. Specifically, instead of having \( z \) sampled from the approximate posterior \( \mathcal{N}(\mu_x, \text{diag}(\sigma^2_x)) \), we consider a sequence of latent variables \( z_{1:T} \) where \( z_i \) is sampled from \( \mathcal{N}(\mu_i, \text{diag}(\sigma^2_i)) \) and given \( z_{<i} \). We believe that this not only provides a more expressive latent space, it allows us to have more control on the variations at different time-steps.

- Incorporating Auxiliary information: Our proposed methods for diverse human motion prediction are completely unsupervised; no extra annotation/feedback are required to train our models. However, one can certainly benefit from additional information to guide the latent space to learn better (more semantically meaningful) representations. For instance, in the case of human motion prediction, the motion capture datasets are often come with additional action label; the action that the subject is performing while his/her motion is being captured. One can utilize this information with a shallow action classifier on top of the latent space, forcing the latent space to learn not only the information that is useful for motion reconstruction, but also the ones that are discriminative enough to distinguish multiple actions.

- Going beyond poses: Our motion prediction approaches works on abstract pose representation of the human at each time-step. That is a representation of e.g., 32 body joints represented in either 3D position or 4D Quaternion representation. One potential future direction is to evaluate how our methods perform when dealing with very high dimensional data, e.g., video frames or fine-grained human shapes.
Appendix A

A.1 Technical Background on Evidence Lower Bound

Our goal is to solve a maximum likelihood problem. To this end, as discussed in the main paper, we rely on Variational Inference, which approximates the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$ with another distribution $q_\phi(\mathbf{z}|\mathbf{x})$. This distribution is computed via another neural network parameterized by $\phi$ (called variational parameters), such that $q_\phi(\mathbf{z}|\mathbf{x}) \approx p_\theta(\mathbf{z}|\mathbf{x})$. Using such an approximation, Variational Autoencoders [Kingma and Welling, 2013], or VAEs in short, are able to optimize the marginal likelihood in a tractable way. The optimization objective of the VAEs is a variational lower bound, also known as evidence lower bound, or ELBO in short.

Specifically, to find an approximation of the posterior that represents the true one, variational inference minimizes the Kullback-Leibler (KL) divergence between the approximate and true posteriors. This divergence can be written as

$$D_{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})] = \sum_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}.$$  

(A.1)

This can further be seen as an expectation, yielding

$$D_{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})] = \mathbf{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[ \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} \right] = \mathbf{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[ \log q_\phi(\mathbf{z}|\mathbf{x}) - \log p_\theta(\mathbf{z}|\mathbf{x}) \right] + \log p_\theta(\mathbf{x}).$$  

(A.2)

According to Bayes’ theorem, the second term above, i.e., the true posterior, can be written as $p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p_\theta(\mathbf{x} | \mathbf{z})p(\mathbf{z})}{p_\theta(\mathbf{x})}$. The data distribution $p_\theta(\mathbf{x})$ is independent of the latent variable $\mathbf{z}$, and can thus be pulled out of the expectation term, giving

$$D_{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})] = \mathbf{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[ \log q_\phi(\mathbf{z}|\mathbf{x}) - \log p_\theta(\mathbf{x} | \mathbf{z}) - \log p(\mathbf{z}) \right] + \log p_\theta(\mathbf{x}).$$  

(A.3)
By moving the $\log p_\theta(x)$ term to the right-hand side of the above equation, we can write

$$
D_{KL}[q_\phi(z|x)||p_\theta(z|x)] - \log p_\theta(x) = \mathbb{E}_{z \sim q_\phi(z|x)} \left[ \log q_\phi(z|x) - \log p_\theta(x|z) - \log p(z) \right]
$$

$$
\log p_\theta(x) - D_{KL}[q_\phi(z|x)||p_\theta(z|x)] = \mathbb{E}_{z \sim q_\phi(z|x)} \left[ \log p_\theta(x|z) - (\log q_\phi(z|x) - \log p(z)) \right]
$$

$$
= \mathbb{E}_{z \sim q_\phi(z|x)} \left[ \log p_\theta(x|z) \right] - \mathbb{E}_{z \sim q_\phi(z|x)} \left[ \log q_\phi(z|x) - \log p(z) \right].
$$

(A.4)

The second expectation term in the resulting equation is, by definition, the KL divergence between the approximate posterior $q_\phi(z|x)$ and the prior $p(z)$. This lets us write

$$
\log p_\theta(x) - D_{KL}[q_\phi(z|x)||p_\theta(z|x)] = \mathbb{E}_{z \sim q_\phi(z|x)} \left[ \log p_\theta(x|z) \right] - D_{KL}[q_\phi(z|x)||p(z)].
$$

(A.5)

In this equation, $\log p_\theta(x)$ is the log-likelihood of the data, which we would like to maximize; $D_{KL}[q_\phi(z|x)||p_\theta(z|x)]$ is the KL divergence between the approximate and the true posterior distributions, which, while not computable, is by definition non-negative; $\mathbb{E}_{z \sim q_\phi(z|x)} \left[ \log p_\theta(x|z) \right]$ is the reconstruction loss; and $D_{KL}[q_\phi(z|x)||p(z)]$ is the KL divergence between the approximate posterior distribution and a prior over the latent variable. This last term can be seen as a regularizer of the latent representation. Altogether, the intractability and non-negativity of $D_{KL}[q_\phi(z|x)||p_\theta(z|x)]$ only allows us to optimize a lower bound of the log-likelihood of the data, given by

$$
\log p_\theta(x) \geq \mathbb{E}_{z \sim q_\phi(z|x)} \left[ \log p_\theta(x|z) \right] - D_{KL}[q_\phi(z|x)||p(z)].
$$

(A.6)

This is referred to as variational or evidence lower bound (ELBO).

### A.2 Derivation of LCP-VAE’s KL Divergence Loss

In our approach, the model encourages the posterior of LCP-VAE to be close to the one of CS-VAE. In general, the KL divergence between two distributions $P_1$ and $P_2$ is defined as

$$
D_{KL}(P_1||P_2) = \mathbb{E}_{P_1} \left[ \log \frac{P_1}{P_2} \right].
$$

(A.7)

Let us now consider the case where the distributions are multivariate Gaussians $\mathcal{N}(\mu, \Sigma)$ in $\mathbb{R}^d$, where $\Sigma = \text{diag}(\sigma^2)$, with $\sigma$ and $\mu$ are $d$-dimensional vectors predicted by the encoder network of the VAE. The density function of such a distribution is

$$
p(x) = \frac{1}{(2\pi)^{\frac{d}{2}} \det(\Sigma)^{\frac{1}{2}}} \exp \left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right).
$$

(A.8)
Thus, the KL divergence between two multivariate Gaussians is computed as

$$D_{KL}(P_1 || P_2) = \frac{1}{2} \mathbb{E}_{P_1} \left[ -\log \det \Sigma_1 - (x - \mu_1)^T \Sigma_1^{-1} (x - \mu_1) + \log \det \Sigma_2 + (x - \mu_2)^T \Sigma_2^{-1} (x - \mu_2) \right]$$

$$= \frac{1}{2} \log \det \Sigma_2 \frac{\text{det} \Sigma_1}{\text{det} \Sigma_1} + \frac{1}{2} \mathbb{E}_{P_1} \left[ - (x - \mu_1)^T \Sigma_1^{-1} (x - \mu_1) + (x - \mu_2)^T \Sigma_2^{-1} (x - \mu_2) \right]$$

$$= \frac{1}{2} \log \det \Sigma_2 \frac{\text{det} \Sigma_1}{\text{det} \Sigma_1} + \frac{1}{2} \mathbb{E}_{P_1} \left[ - \text{tr} \{ \Sigma_1^{-1} (x - \mu_1) (x - \mu_1)^T \} + \text{tr} \{ \Sigma_2^{-1} (x - \mu_2) (x - \mu_2)^T \} \right]$$

$$= \frac{1}{2} \log \det \Sigma_2 \frac{\text{det} \Sigma_1}{\text{det} \Sigma_1} + \frac{1}{2} \mathbb{E}_{P_1} \left[ - \text{tr} \{ \Sigma_1^{-1} \Sigma_1 \} + \text{tr} \{ \Sigma_2^{-1} (xx^T - 2x\mu_1^T + \mu_1\mu_2^T) \} \right]$$

$$= \frac{1}{2} \log \frac{\text{det} \Sigma_2}{\text{det} \Sigma_1} - d + \text{tr} \{ \Sigma_2^{-1} \Sigma_1 \} + \text{tr} \{ \mu_1^T \Sigma_2^{-1} \mu_1 - 2\mu_1^T \Sigma_2^{-1} \mu_2 + \mu_2^T \Sigma_2^{-1} \mu_2 \}$$

$$= \frac{1}{2} \left[ \log \frac{\text{det} \Sigma_2}{\text{det} \Sigma_1} - d + \text{tr} \{ \Sigma_2^{-1} \Sigma_1 \} + (\mu_2 - \mu_1)^T \Sigma_2^{-1} (\mu_2 - \mu_1) \right].$$

(A.9)

where $\text{tr} \{ \cdot \}$ denotes the trace operator. In Eq. A.9, the covariance matrix $\Sigma_1$ and mean $\mu_1$ correspond to distribution $P_1$ and the covariance matrix $\Sigma_2$ and mean $\mu_2$ correspond to distribution $P_2$. Given this result, we can then compute the KL divergence of the LCP–VAE and the posterior distribution with mean $\mu + \sigma \circ \mu_c$ and covariance matrix $\text{diag}((\sigma \circ \sigma_c)^2)$. Let $\Sigma = \text{diag}(\sigma^2)$, $\Sigma_c = \text{diag}(\sigma_c^2)$, and $d$ be the dimensionality of the latent space. The loss in Eq. 6 of the main paper can then be written as

$$L_{\text{prior}}^{\text{LCP–VAE}} = -\frac{1}{2} \left[ \log \frac{|\Sigma_c|}{|\Sigma|} - d + \text{tr} \{ \Sigma_c^{-1} \Sigma \} + (\mu_c - (\mu + \Sigma \mu_c))^T \Sigma_c^{-1} (\mu_c - (\mu + \Sigma \mu_c)) \right].$$

(A.10)

Since $\Sigma_c^{-1} \Sigma = 1$, $|\Sigma_c|$ will be cancelled out in the log term, which yields

$$L_{\text{prior}}^{\text{LCP–VAE}} = -\frac{1}{2} \left[ \log \frac{1}{|\Sigma|} - d + \text{tr} \{ \Sigma \} + (\mu_c - (\mu + \Sigma \mu_c))^T \Sigma_c^{-1} (\mu_c - (\mu + \Sigma \mu_c)) \right].$$

(A.11)

A.3 Mitigating Posterior Collapse: Related Work

Deep generative models offer promising results in generating diverse, realistic samples, such as images, text, motion, and sound, from purely unlabeled data. One example of such successful generative models are variational autoencoders [Kingma and Welling [2013] (VAEs), the stochastic variant of autoencoders, which, thanks to strong and expressive decoders, can generate high-quality samples. Training such models, however, may often result in posterior collapse: the posterior distribution $q(z|x)$ of the latent variable $z$ given the input $x$ becomes
equal to the prior distribution, resulting in a latent variable carrying no information about the input. In other words, the model learns to ignore the latent variable.

The most common approaches to tackling posterior collapse consist of weighing the KL divergence between the posterior and prior during training by an annealing function [Bowman et al., 2015; Yang et al., 2017; Kim et al., 2018; Gulrajani et al., 2016; Liu et al., 2019], weakening the decoder [Semeniuta et al., 2017; Zhao et al., 2017b], or changing the training objective [Zhao et al., 2017b; Tolstikhin et al., 2017]. All of them are based on the perspective that the solution to posterior collapse can be found in a good local optimum in terms of evidence lower bound [Chen et al., 2016; Alemi et al., 2017]. However, they each suffer from drawbacks: Any annealing weight that does not become and remain equal to one at some point during training yields an improper statistical model; weakening the decoder tends to degrade the quality of the generated samples; changing the objective does not optimize the true variational lower bound. As an alternative, some methods modify the training strategy to more strongly encourage the inference network to approximate the model’s true posterior [He et al., 2019; Li et al., 2019]. Other methods add auxiliary tasks either with non-autoregressive models [Lucas and Verbeek, 2018] or that exploit the latent variable [Goyal et al., 2017; Lucas and Verbeek, 2018; Dieng et al., 2018]. While this encourages the latent variable to carry some information, it may not be directly useful for the main task. Alternatively, several techniques incorporate constraints in VAEs. In this context, VQ-VAE [van den Oord et al., 2017] introduces a discrete latent variable obtained by vector quantization of the latent one that, given a uniform prior over the outcome, yields a fixed KL divergence equal to \( \log K \), with \( K \) the size of the codebook; several recent works use the von Mises-Fisher distribution to obtain a fixed KL divergence, thus mitigating the posterior collapse problem [Guu et al., 2018; Xu and Durrett, 2018; Davidson et al., 2018]; more recently, delta-VAE [Razavi et al., 2019] modifies the posterior such that it maintains a minimum distance between the prior and the posterior.

Although successful at handling the posterior collapse in the presence of expressive decoders, e.g., LSTM, GRUs, PixelCNN, all of these approaches were designed for standard VAEs, not conditional VAEs (CVAEs). As such, they do not address the problem of mitigating the influence of a strong conditioning signal in ignoring the latent variable, which is our focus here. In this context, we observed that a strong condition provides enough information for an expressive decoder to reconstruct the data, thus allowing the decoder to ignore the latent variable at no loss in reconstruction quality.

### A.4 Further Discussion on the Performance of Stochastic Base-lines

The MT-VAE model [Yan et al., 2018] tends to ignore the random variable \( z \), thus ignoring the root of variation. As a consequence, it achieves a low diversity, much lower than ours, but produces samples of high quality, albeit almost identical (see the diversity of the generated motions in quantitative evaluations). We empirically observed this by analysing the weights acting on the latent variable \( z \) and the ones acting on the conditioning signal (i.e., the hidden state of the past motion). We observed the magnitude of the weights acting on \( z \) to be orders of magnitude smaller than that of acting on the condition, 0.008 versus 232.85, respectively.
To further confirm that the MT-VAE ignores the latent variable, we performed an additional experiment where, at test time, we sampled each element of the random vector independently from \( \mathcal{N}(50, 50) \) instead of from the prior \( \mathcal{N}(0, I) \). This led to neither loss of quality nor increase of diversity of the generated motions.

Our experiments with the HP-GAN model [Barsoum et al., 2018] evidence the limited diversity of the sampled motions despite its use of random noise during inference. Note that the authors of [Barsoum et al., 2018] mentioned in their paper that the random noise was added to the hidden state. Only by studying their publicly available code\(^1\) did we understand the precise way this combination was done. In fact, the addition relies on a parametric, linear transformation of the noise vector. That is, the perturbed hidden state is obtained as

\[
h_{\text{perturbed}} = h_{\text{original}} + W^{z \rightarrow h} z.
\]

Because the parameters \( W^{z \rightarrow h} \) are learned, the model has the flexibility to ignore \( z \), which leads to the low diversity of sampled motions. Note that the authors of [Barsoum et al., 2018] acknowledged that, despite their best efforts, they noticed very little variation between predictions obtained with different \( z \) values. We further analysed this phenomenon and observed that the magnitude of \( W^{z \rightarrow h} \) is in the order of \( O(1 \times 10^{-3}) \), confirming the fact that it provides the model with the flexibility to ignore \( z \). Since the perturbation is ignored, however, the quality of the generated motions is high.

Pose-Knows [Walker et al., 2017], produces motions with higher diversity than the aforementioned two baselines, but of much lower quality. The main reason behind this is that the random vectors that are concatenated to the poses at each time-step are sampled independently of each other, which translates to discontinuities in the generated motions. This problem might be mitigated by sampling the noise in a time-dependent, autoregressive manner, as in [Kumar et al., 2019] for video generation. Doing so, however, goes beyond the scope of our analysis.

The Mix-and-Match approach [Aliakbarian et al., 2020b] yields sampled motions with higher diversity and reasonable quality. The architecture of Mix-and-Match is very close to that of MT-VAE, but replaces the deterministic concatenation operation with a stochastic perturbation of the hidden state with the noise. Through such a perturbation, the decoder is not able decouple the noise and the condition, by contrast with concatenation. However, since the perturbation is not learned and is a non-parametric operation, the quality of the generated motion is lower than ours and of other baselines (except for Pose-Knows). We see the Mix-and-Match perturbation as a workaround to the posterior collapse problem, which nonetheless sacrifices the quality and the context of the sampled motions.

A.5 Ablation Study on Different Means of Conditioning

In addition to the experiments in the main paper, we also study various designs to condition the VAE encoder and decoder. As discussed before, conditioning the VAE encoder can be safely done via concatenating two deterministic sources of information, i.e., the representations of the past and the future, since both sources are useful to compress the future motion into the latent

\(^1\)https://github.com/ebarsoum/hpgan
Appendix A

Table A.1: Evaluation of various architecture designs for a CVAE. A smaller KL value, indicating posterior collapse, leads to less diversity.

| Encoder Conditioning | Decoder Conditioning | LCP–VAE’s Training KL |
|----------------------|----------------------|-----------------------|
| Concatenation ($z_c$) | Reparameterization ($z_c$) | 6.92 |
| Concatenation ($h_t$) | Concatenation ($h_t$) | 0.04 |
| Concatenation ($z_c$) | Concatenation ($z_c$) | 0.61 |
| Concatenation ($h_t$) | Reparameterization ($z_c$) | 8.07 |

Table A.2: Quantitative evaluation on the Penn Action dataset. Note that a diversity of 1.21 is reasonably high for normalized 2D joint positions, i.e., values between 0 and 1, normalized with the width and the height of the image.

| Method                         | ELBO (KL) (Reconstructed) | Diversity (Sampled) | Quality (Sampled) | Context (Sampled) | Training KL (Reconstructed) |
|-------------------------------|---------------------------|---------------------|------------------|------------------|-----------------------------|
| LCP–VAE                       | 0.034 (6.07)              | 1.21                | 0.46             | 0.70             | 4.84                        |
| Autoregressive Counterpart    | 0.048 (N/A)               | 0.00                | 0.46             | 0.51             | N/A                         |

space. In Table A.1, we use both a deterministic representation of the observation, $h_t$, and a stochastic one, $z_c$, as a conditioning variable for the encoder. Similarly, we compare the use of either of these variables via concatenation with that of our modified reparameterization trick (explained in the main paper). This shows that, to condition the decoder, reparameterization is highly effective at addressing posterior collapse. Furthermore, for the encoder, a deterministic condition works better than a stochastic one. When both the encoder and decoder are conditioned via deterministic conditioning variables, i.e., row 2 in Table A.1, the model learns to ignore the latent variable and rely solely on the condition, as evidenced by the KL term tending to zero.

A.6 Experimental Results on the Penn Action Dataset

As a complementary experiment, we evaluate our approach on the Penn Action dataset, which contains 2326 sequences of 15 different actions, where for each person, 13 joints are annotated in 2D space. Most sequences have less than 50 frames and the task is to generate the next 35 frames given the first 15. Results are provided in Table A.2. Note that the upper bound for the Context metric is 0.74, i.e., the classification performance given the Penn Action ground-truth motions.

A.7 Stochastic Human Motion Prediction Architecture

Our motion prediction model follows the architecture depicted in Fig. 2 of the main paper. Below, we describe the architecture of each component in our model. Note that human poses, consisting of 32 joints in case of the Human3.6M dataset, are represented in 4D quaternion space. Thus, each pose at each time-step is represented with a vector of size $1 \times 128$. All the
tensor sizes described below ignore the mini-batch dimension for simplicity.

The observed motion encoder, or CS-VAE motion encoder, is a single layer GRU [Chung et al., 2014] network with 1024 hidden units. If the observation sequence has length \( T_{\text{obs}} \), the observed motion encoder maps \( T_{\text{obs}} \times 128 \) into a single hidden representation of size \( 1 \times 1024 \), i.e., the hidden state of the last time-step. This hidden state, \( h_t \), acts as the condition to the LCP-VAE encoder and the direct input to the CS-VAE encoder.

CS-VAE, similarly to any variational autoencoder, has an encoder and a decoder. The CS-VAE encoder is a fully-connected network with ReLU non-linearities, mapping the hidden state of the motion encoder, i.e., \( h_t \), to an embedding of size \( 1 \times 512 \). Then, to generate the mean and standard deviation vectors, we use two fully connected branches. They map the embedding of size \( 1 \times 512 \) to a mean vector of size \( 1 \times 128 \) and a standard deviation vector of size \( 1 \times 128 \), where 128 is the length of the latent variable. Note that we apply a ReLU non-linearity to the vector of standard deviations to ensure that it is non-negative. We then use the reparameterization trick [Kingma and Welling, 2013] to sample a latent variable of size \( 1 \times 128 \). The CS-VAE decoder consists of multiple fully-connected layers, mapping the latent variable to a variable of size \( 1 \times 1024 \), acting as the initial hidden state of the observed motion decoder. Note that we apply a Tanh non-linearity to the generated hidden state to mimic the properties of a GRU hidden state.

The observed motion decoder, or CS-VAE motion decoder, is similar to its motion encoder, except for the fact that it reconstructs the motion autoregressively. Additionally, it is initialized with the reconstructed hidden state, i.e., the output of the CS-VAE decoder. The output of each GRU cell at each time-step is then fed to a fully-connected layer, mapping the GRU output to a vector of size \( 1 \times 128 \), which represents a human pose with 32 joints in 4D quaternion space. To decode the motions, we use a teacher forcing technique [Williams and Zipser, 1989] during training. At each time-step, the network chooses with probability \( P_{tf} \) whether to use its own output at the previous time-step or the ground-truth pose as input. We initialize \( P_{tf} = 1 \), and decrease it linearly at each training epoch such that, after a certain number of epochs, the model becomes completely autoregressive, i.e., uses only its own output as input to the next time-step. Note that, at test time, the motions are generated completely autoregressively, i.e., with \( P_{tf} = 0 \).

Note that the future motion encoder and decoder have exactly the same architectures as the observed motion ones. The only difference is their input, where the future motion is represented by poses from \( T_{\text{obs}} \) to \( T_{\text{end}} \) in a sequence. In the following, we describe the architecture of LCP-VAE for motion prediction.

LCP-VAE is a conditional variational encoder. Its encoder’s input is a representation of future motion, i.e., the last hidden state of the future motion encoder, \( h_T \), conditioned on \( h_t \). The conditioning is done by concatenation, thus the input to the encoder is a representation of size \( 1 \times 2048 \). The LCP-VAE encoder, similarly to the CS-VAE encoder, maps its input representation to an embedding of size \( 1 \times 512 \). Then, to generate the mean and standard deviation vectors, we use two fully connected branches, mapping the embedding of size \( 1 \times 512 \) to a mean vector of size \( 1 \times 128 \) and a standard deviation vector of size \( 1 \times 128 \), where 128 is the length of the latent variable. Note that we apply a ReLU non-linearity to the vector of standard deviations to ensure that it is non-negative. To sample the latent variable, we use our extended reparameterization trick, explained in the main paper. This unifies the conditioning
and sampling of the latent variable. Then, similarly to CS–VAE, the latent variable is fed to the LCP–VAE decoder, which is a fully connected network that maps the latent representation of size $1 \times 128$ to a reconstructed hidden state of size $1 \times 1024$ for future motion prediction. Note that we apply a Tanh non-linearity to the generated hidden state to mimic the properties of a GRU hidden state.

### A.8 Diverse Image Captioning Architecture

Our diverse image captioning model follows the architecture depicted in Fig.2 of the main paper. Below, we describe the architecture of each component in our model. Note that all tensor sizes described below ignore the mini-batch dimension for simplicity.

The **image encoder** is a ResNet152 [He et al., 2016] pretrained on ImageNet [Krizhevsky et al., 2012]. Given the encoder, the conditioning signal is a $1 \times 2048$ feature representation. Note that, to avoid an undesirable equilibrium in the reconstruction loss of the CS–VAE, we freeze the ResNet152 during training.

CS–VAE is a standard variational autoencoder. Its encoder maps the input representation of size $1 \times 2048$ to an embedded representation of size $1 \times 1024$. Then, to generate the mean and standard deviation vectors, we use two fully connected branches, mapping the embedding of size $1 \times 1024$ to a mean vector of size $1 \times 256$ and a standard deviation vector of size $1 \times 256$, where 256 is the length of the latent variable. The decoder of the CS–VAE maps the sampled latent variable of size $1 \times 256$ to a representation of size $1 \times 2048$. The generated representation acts as a reconstructed image representation. During training, we learn the reconstruction by computing the smoothed $L_1$ loss between the generated representation and the image feature of the frozen ResNet152.

The **caption encoder** is a single layer GRU network with a hidden size of 1024. Each word in the caption is represented using a randomly initialized embedding layer that maps each word to a representation of size $1 \times 1024$.

LCP–VAE is a conditional variational autoencoder. As input to its encoder, we first concatenate the image representation of size $1 \times 2048$ to the caption representation of size $1 \times 1024$. The encoder then maps this representation to an embedded representation of size $1 \times 1024$. Then, to generate the mean and standard deviation vectors, we use two fully connected branches, mapping the embedding of size $1 \times 1024$ to a mean vector of size $1 \times 256$ and a standard deviation vector of size $1 \times 256$, where 256 is the length of the latent variable. To sample the latent variable, we make use of our extended reparameterization trick, explained in the main paper. This unifies the conditioning and sampling of the latent variable. The LCP–VAE decoder then maps this latent representation to a vector of size $1 \times 1024$ via a fully-connected layer. We then apply batch normalization [Ioffe and Szegedy, 2015] on the representation, which acts as the first token to the caption decoder.

The **caption decoder** is also a single layer GRU network with a hidden size of 1024. Its first token is the representation generated by the LCP–VAE decoder, while the remaining tokens are the words in the corresponding caption. To decode the caption, we use a teacher forcing technique during training. At each time-step, the network chooses with probability $P_{tf}$ whether to use its own output at the previous time-step or the ground-truth token as input. We initialize $P_{tf} = 1$, and decrease it linearly at each training epoch such that, after a certain
number of epochs, the model becomes completely autoregressive, i.e., uses only its own output as input to the next time-step. Note that, at test time, the captions are generated completely autoregressively, i.e., with $P_{ij} = 0$.

### A.9 Diverse Text Generation: Related Work

A number of studies utilize generative models for language modeling. For instance, [Fang et al., 2019] uses VAEs and LSTMs in an unconditional language modeling problem, where posterior collapse may occur if the VAE is not trained well. To handle the problem of posterior collapse in language modeling, the authors of [Fang et al., 2019] try to directly match the aggregated posterior to the prior. This can be considered an extension of variational autoencoders with a regularization to maximize the mutual information, addressing the posterior collapse. VAEs are also used for language modeling in [Li et al., 2019]. In this context, it was observed that VAEs make it hard to find a good balance between language modeling and representation learning. To improve the training of VAEs in such scenarios, the authors of [Li et al., 2019] first pretrain the inference network in an autoencoder fashion, such that the inference network learns a good representation of the data in a deterministic manner. Then, they train the whole VAE while considering a weight for the KL term during training. However, the second step modifies the way VAEs optimize the variational lower bound. Furthermore, this approach prevents the model from being trained end-to-end. Unlike these approaches, our method considers the case of *conditional* text generation, where the conditioning signal (the image to be captioned in our case) is strong enough such that the caption generator can rely solely on that.

The recent work of [Cho et al., 2019] proposes to separate diversification from generation for the tasks of sequence generation and language modeling. The diversification stage uses a mixture of experts (MoE) to sample different binary masks on the source sequence for diverse content selection. The generation stage uses a standard encoder-decoder model taking as input the selected content from the source sequence. While effective at generating diverse sequences, this approach relies heavily on the selection part, where one needs to select the most important information in the source to generate the target sequence. Thus, the diversity of the generated target sequence depends on the diversity of the selected parts of the source sequence. Similarly, the authors of [Shen et al., 2019] utilize an MoE for the task of diverse machine translation. While this task is considered to be a diverse text generation one, with indeed diverse translations generated from each source sentence, the methods addressing it rely on the availability of the a stochastic dataset, i.e., having access to multiple target sequences for each source sentence during training. By contrast, here, we design an approach that works with deterministic datasets, as those available for human motion prediction.

### A.10 Ablation Study on Diverse Image Captioning

In addition to the experiments in the main paper, in Table A.3, we evaluate our approach, the autoregressive baseline and the CVAE, in terms of BLEU scores, i.e., BLEU1, BLEU2, BLEU3, and BLEU4 of the captions generated at test time. For the autoregressive baseline,
Table A.3: BLEU scores of different orders for sampled captions from our model as well as the baselines.

| Model                                      | BLEU1 | BLEU2 | BLEU3 | BLEU4 |
|--------------------------------------------|-------|-------|-------|-------|
| Autoregressive (deterministic)             | 0.46  | 0.39  | 0.21  | 0.16  |
| Conditional VAE (best of $K$ captions)     | 0.44  | 0.38  | 0.20  | 0.17  |
| LCP–VAE (caption from mode)                | 0.44  | 0.37  | 0.20  | 0.14  |
| LCP–VAE (best of $K$ captions)             | 0.45  | 0.39  | 0.23  | 0.18  |

the model generates one caption per image, which makes it straightforward to compute the BLEU scores. For the CVAE, we consider the best BLEU score among all $K = 50$ sampled captions, according to the best-matching ground-truth caption. For our model, we consider the caption from the mode, i.e., the one sampled from $z = \mu_c$. Although the caption sampled from LCP–VAE is not chosen based on the best match with the ground-truth caption, it yields promising quality in terms of BLEU scores. For the sake of completeness and fairness, we also provide the results obtained with the best of $K$ captions for our approach, which outperform all the baselines.

### A.11 Additional Motion Prediction Qualitative Results

In this section we provide additional qualitative results for our LCP-VAE approach, illustrated in Fig. A.1 to Fig. A.5.

### A.12 Additional Image Captioning Qualitative Results

In this section we provide additional qualitative results for our LCP-VAE approach, illustrated in Fig. A.6 to Fig. A.10.
Figure A.1: Qualitative evaluation of the diversity in human motion. The first row illustrates the ground-truth motion. The first six poses of each row depict the observation (the condition) and the rest are sampled from our model. Each row is a randomly sampled motion (not cherry picked). As can be seen, all sampled motions are natural, with a smooth transition from the observed to the generated ones. The diversity increases as we increase the sequence length.
Figure A.2: Additional qualitative evaluation of the diversity in human motion.

Figure A.3: Additional qualitative evaluation of the diversity in human motion.
Figure A.4: Additional qualitative evaluation of the diversity in human motion.

Figure A.5: Additional qualitative evaluation of the diversity in human motion.
Appendix A

Figure A.6: Qualitative evaluation of the diversity in generated captions. While captions generated by our approach are diverse, they all describe the image properly. The caption from mode also usually achieves a good descriptive caption.

Figure A.7: Additional qualitative evaluation of the diversity in generated captions.

Draft Copy – 12 October 2020
§A.12 Additional Image Captioning Qualitative Results

Figure A.8: Additional qualitative evaluation of the diversity in generated captions.

Figure A.9: Additional qualitative evaluation of the diversity in generated captions.
Figure A.10: Additional qualitative evaluation of the diversity in generated captions.

ground-truth Captions:
1. A woman skiing down a snow covered slope next to two brown dogs.
2. A woman with two dogs skiing cross country.
3. Two yellow dogs are in the snow as a woman is cross country skiing.
4. A woman is cross country skiing along with two dogs in the snow.
5. Two dogs are sitting in the snow with its owner.

Caption from Autoregressive:
a man standing in the snow with his skis.

Caption from Conditional VAE:
a man is walking down a snowy hill with a dog.

Captions from CPP-VAE’s sampling:
1. a woman on a snow board grinds on a snowy hill.
2. a man in a red jacket and hat is skiing down a hill.
3. a group of dog is laying on a snow covered bank.
4. a man in a red jacket and black pants skiing down a hill.
5. a woman on skis in the snow with a dog.
6. a dog with another dog is walking on the snow covered path.
7. a dog is lying down on a snow covered slope.
8. a man is standing in the snow with a snowboard attached to his feet.
9. a dog is walking on the snow with a leash attached to its mouth.

Caption from CPP-VAE’s mode:
a couple of dogs are standing on a snowy hill with a person in the background.
Bibliography

ALEMI, A. A.; POOLE, B.; FISCHER, I.; DILLON, J. V.; SAUROUS, R. A.; AND MURPHY, K., 2017. Fixing a broken elbo. arXiv preprint arXiv:1711.00464, (2017). (cited on pages 77 and 96)

ALIAKBARIAN, M. S.; SALEH, F.; FERNANDO, B.; SALZMANN, M.; PETERSSON, L.; AND ANDERSSON, L., 2016. Deep action-and context-aware sequence learning for activity recognition and anticipation. arXiv preprint arXiv:1611.05520, (2016). (cited on pages 5 and 74)

ALIAKBARIAN, M. S.; SALEH, F. S.; SALZMANN, M.; FERNANDO, B.; PETERSSON, L.; AND ANDERSSON, L., 2018a. Viena2: A driving anticipation dataset. In Asian Conference on Computer Vision, 449–466. Springer. (cited on page 3)

ALIAKBARIAN, M. S.; SALEH, F. S.; SALZMANN, M.; FERNANDO, B.; PETERSSON, L.; AND ANDERSSON, L., 2018b. Viena2: A driving anticipation dataset. arXiv preprint arXiv:1810.09044, (2018). (cited on page 88)

ALIAKBARIAN, S.; SALEH, F. S.; SALZMANN, M.; PETERSSON, L.; AND GOULD, S., 2019. Sampling good latent variables via cpp-vaes: Vaes with condition posterior as prior. arXiv preprint arXiv:1912.08521, (2019). (cited on page 4)

ALIAKBARIAN, S.; SALEH, F. S.; SALZMANN, M.; PETERSSON, L.; AND GOULD, S., 2020a. A stochastic conditioning scheme for diverse human motion prediction. In The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). (cited on page 4)

ALIAKBARIAN, S.; SALEH, F. S.; SALZMANN, M.; PETERSSON, L.; GOULD, S.; AND HABIBIAN, A., 2020b. A stochastic conditioning scheme for diverse human motion prediction. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE. (cited on pages 76, 78, 79, 84, 85, 86, and 97)

BAO, J.; CHEN, D.; WEN, F.; LI, H.; AND HUA, G., 2017. Cvae-gan: fine-grained image generation through asymmetric training. In Proceedings of the IEEE International Conference on Computer Vision, 2745–2754. (cited on page 60)

BARSOUM, E.; KENDER, J.; AND LIU, Z., 2018. Hp-gan: Probabilistic 3d human motion prediction via gan. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 1418–1427. (cited on pages 59, 60, 61, 62, 67, 68, 70, 71, 73, 75, 76, 78, 84, 85, 86, and 97)
BERTASIUS, G.; SHI, J.; AND TORRESANI, L., 2015a. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, 4380–4389. (cited on page 5)

BERTASIUS, G.; SHI, J.; AND TORRESANI, L., 2015b. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4380–4389. (cited on page 21)

BILEN, H.; FERNANDO, B.; GAVVES, E.; VEDALDI, A.; AND GOULD, S., 2016a. Dynamic image networks for action recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 3034–3042. (cited on pages 15 and 37)

BILEN, H.; FERNANDO, B.; GAVVES, E.; VEDALDI, A.; AND GOULD, S., 2016b. Dynamic image networks for action recognition. In IEEE International Conference on Computer Vision and Pattern Recognition CVPR. (cited on pages 30 and 37)

BOWMAN, S. R.; VILNIS, L.; VINYALS, O.; DAI, A. M.; JOZEFOWICZ, R.; AND BENGIO, S., 2015. Generating sentences from a continuous space. arXiv preprint arXiv:1511.06349, (2015). (cited on pages 6, 65, 77, 82 and 96)

BÜTEPAGE, J.; BLACK, M. J.; KRAGIC, D.; AND KJELLSTROM, H., 2017. Deep representation learning for human motion prediction and classification. In Proceedings of the IEEE conference on computer vision and pattern recognition, 6158–6166. (cited on page 61)

BÜTEPAGE, J.; KJELLSTRÖM, H.; AND KRAGIC, D., 2018. Anticipating many futures: Online human motion prediction and generation for human-robot interaction. In 2018 IEEE International Conference on Robotics and Automation (ICRA), 1–9. IEEE. (cited on pages 61 and 76)

CARREIRA, J. AND ZISSERMAN, A., 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 6299–6308. (cited on page 5)

CHAN, F.-H.; CHEN, Y.-T.; XIANG, Y.; AND SUN, M., 2016. Anticipating accidents in dashcam videos. In Asian Conference on Computer Vision, 136–153. Springer. (cited on page 41)

CHEN, W.; XIONG, C.; XU, R.; AND CORSO, J. J., 2014. Actionness ranking with lattice conditional ordinal random fields. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 748–755. (cited on pages 16 and 18)

CHEN, X.; KINGMA, D. P.; SALIMANS, T.; DUAN, Y.; DHARIWAL, P.; SCHULMAN, J.; SUTSKEVER, I.; AND ABBEEL, P., 2016. Variational lossy autoencoder. arXiv preprint arXiv:1611.02731, (2016). (cited on pages 77 and 96)

CHÉRON, G.; LAPTEV, I.; AND SCHMID, C., 2015. P-cnn: Pose-based cnn features for action recognition. In Proceedings of the IEEE International Conference on Computer Vision, 3218–3226. (cited on pages 16 and 18)
CHO, J.; SEO, M.; AND HAJISHIRZI, H., 2019. Mixture content selection for diverse sequence generation. *arXiv preprint arXiv:1909.01953*, (2019). (cited on page 101)

CHO, K.; VAN MERRIËNBOER, B.; GULCEHRE, C.; BAHDANAU, D.; BOUGARES, F.; SCHWENK, H.; AND BENGIO, Y., 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, (2014). (cited on pages 8 and 68)

CHOLLET, F., 2015. keras, [https://github.com/fchollet/keras](https://github.com/fchollet/keras) (cited on page 24)

CHUNG, J.; GULCEHRE, C.; CHO, K.; AND BENGIO, Y., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, (2014). (cited on pages 8 and 99)

DAVIDSON, T. R.; FALORSI, L.; DE CAO, N.; KIPF, T.; AND TOMCZAK, J. M., 2018. Hyperspherical variational auto-encoders. *arXiv preprint arXiv:1804.00891*, (2018). (cited on pages 77 and 96)

DIBA, A.; MOHAMMAD PAZANDEH, A.; PIRSIAVASH, H.; AND VAN GOOL, L., 2016a. Deepcamp: Deep convolutional action & attribute mid-level patterns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3557–3565. (cited on page 16)

DIBA, A.; MOHAMMAD PAZANDEH, A.; PIRSIAVASH, H.; AND VAN GOOL, L., 2016b. Deepcamp: Deep convolutional action & attribute mid-level patterns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3557–3565. (cited on page 18)

DIENG, A. B.; KIM, Y.; RUSH, A. M.; AND BLEI, D. M., 2018. Avoiding latent variable collapse with generative skip models. *arXiv preprint arXiv:1807.04863*, (2018). (cited on pages 77 and 96)

DONAHUE, J.; ANNE HENDRICKS, L.; GUADARRAMA, S.; ROHRBACH, M.; VENUGOPALAN, S.; SAENKO, K.; AND DARRELL, T., 2015a. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2625–2634. (cited on pages xxii, 18, 30, 32, 37, 43, and 47)

DONAHUE, J.; ANNE HENDRICKS, L.; GUADARRAMA, S.; ROHRBACH, M.; VENUGOPALAN, S.; SAENKO, K.; AND DARRELL, T., 2015b. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2625–2634. (cited on pages 8, 15, and 16)

DONG, C.; DOLAN, J. M.; AND LITKOUHI, B., 2017. Intention estimation for ramp merging control in autonomous driving. In *Intelligent Vehicles Symposium (IV), 2017 IEEE*, 1584–1589. IEEE. (cited on page 38)
ENGEL, J.; HOFFMAN, M.; AND ROBERTS, A., 2017. Latent constraints: Learning to generate conditionally from unconditional generative models. *arXiv preprint arXiv:1711.05772*, (2017). (cited on page [60])

ESSER, P.; SUTTER, E.; AND OMMER, B., 2018. A variational u-net for conditional appearance and shape generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8857–8866. (cited on page [60])

FABIAN CABA HEILBRON, B. G.; VICTOR ESCORCIA AND NIEBLES, J. C., 2015. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 961–970. (cited on page [42])

FANG, L.; LI, C.; GAO, J.; DONG, W.; AND CHEN, C., 2019. Implicit deep latent variable models for text generation. *arXiv preprint arXiv:1908.11527*, (2019). (cited on page [101])

FEICHTENHOFER, C.; PINZ, A.; AND WILDES, R. P., 2017. Spatiotemporal multiplier networks for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4768–4777. (cited on page [45])

FEICHTENHOFER, C.; PINZ, A.; AND ZISSERMAN, A., 2016a. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1933–1941. (cited on pages [xxiii 5 37 43 and 47])

FEICHTENHOFER, C.; PINZ, A.; AND ZISSERMAN, A., 2016b. Convolutional two-stream network fusion for video action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*. (cited on pages [16 17 18 19 and 32])

FERNANDO, B.; ANDERSON, P.; HUTTER, M.; AND GOULD, S., 2016a. Discriminative hierarchical rank pooling for activity recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1924–1932. (cited on page [15])

FERNANDO, B.; ANDERSON, P.; HUTTER, M.; AND GOULD, S., 2016b. Discriminative hierarchical rank pooling for activity recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*. (cited on pages [16 18 and 30])

FERNANDO, B.; GAVVES, E.; ORAMAS, J.; GHODRATI, A.; AND TUYTELAARS, T., 2016c. Rank pooling for action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39, 4 (2016), 773–787. (cited on page [15])

FERNANDO, T.; DENMAN, S.; SRIDHARAN, S.; AND FOOKES, C., 2017. Going deeper: Autonomous steering with neural memory networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 214–221. (cited on page [45])

FRAGKIADAKI, K.; LEVINE, S.; FELSEN, P.; AND MALIK, J., 2015. Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, 4346–4354. (cited on pages [60 72 74 77 and 86])

Draft Copy – 12 October 2020
GAIDON, A.; WANG, Q.; CABON, Y.; AND VIG, E., 2016. Virtual worlds as proxy for multi-object tracking analysis. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4340–4349. (cited on page 52)

GANIN, Y. AND LEMPITSKY, V., 2015. Unsupervised domain adaptation by backpropagation. In International Conference on Machine Learning, 1180–1189. (cited on page 50)

GERS, F. A.; SCHMIDHUBER, J.; AND CUMMINS, F., 1999. Learning to forget: Continual prediction with lstm. (1999). (cited on page 9)

GHOSH, P.; SONG, J.; AKSAN, E.; AND HILLIGES, O., 2017. Learning human motion models for long-term predictions. In 2017 International Conference on 3D Vision (3DV), 458–466. IEEE. (cited on pages 60, 72, 77, and 86)

GIRSHICK, R., 2015. Fast r-cnn. In Proceedings of the IEEE international conference on computer vision, 1440–1448. (cited on page 5)

GIRSHICK, R.; DONAHUE, J.; DARRELL, T.; AND MALIK, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, 580–587. (cited on page 5)

GKIOXARI, G. AND MALIK, J., 2015. Finding action tubes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 759–768. (cited on pages XXI, 50, and 31)

GOYAL, A. G. A. P.; SORDONI, A.; CÔTÉ, M.-A.; KE, N. R.; AND BENGIO, Y., 2017. Z-forcing: Training stochastic recurrent networks. In Advances in neural information processing systems, 6713–6723. (cited on pages 77 and 96)

GRAVES, A.; MOHAMED, A.-R.; AND HINTON, G., 2013. Speech recognition with deep recurrent neural networks. In 2013 IEEE international conference on acoustics, speech and signal processing, 6645–6649. IEEE. (cited on page 8)

GUI, L.-Y.; WANG, Y.-X.; LIANG, X.; AND MOURA, J. M., 2018a. Adversarial geometry-aware human motion prediction. In Proceedings of the European Conference on Computer Vision (ECCV), 786–803. (cited on pages 59, 60, 72, 74, 75, 77, 83, and 86)

GUI, L.-Y.; WANG, Y.-X.; RAMANAN, D.; AND MOURA, J. M., 2018b. Few-shot human motion prediction via meta-learning. In Proceedings of the European Conference on Computer Vision (ECCV), 432–450. (cited on pages 60, 77, and 86)

GULRAJANI, I.; KUMAR, K.; Ahmed, F.; TAIGA, A. A.; VISIN, F.; VAZQUEZ, D.; AND COURVILLE, A., 2016. Pixelvae: A latent variable model for natural images. arXiv preprint arXiv:1611.05013, (2016). (cited on pages 77 and 96)

GUU, K.; HASHIMOTO, T. B.; OREN, Y.; AND LIANG, P., 2018. Generating sentences by editing prototypes. Transactions of the Association of Computational Linguistics, 6 (2018), 437–450. (cited on pages 77 and 96)
HE, J.; SPOKOYNY, D.; NEUBIG, G.; AND BERG-KIRKPATRICK, T., 2019. Lagging inference networks and posterior collapse in variational autoencoders. *arXiv preprint arXiv:1901.05534*, (2019). (cited on pages 77 and 96)

HE, K.; ZHANG, X.; REN, S.; AND SUN, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778. (cited on pages 5 and 100)

HERATH, S.; HARANDI, M.; AND PORIKLI, F., 2017. Going deeper into action recognition: A survey. *Image and Vision Computing*, 60 (2017), 4 – 21. Regularization Techniques for High-Dimensional Data Analysis. (cited on page 30)

HOCREITER, S. AND SCHMIDHUBER, J., 1997. Long short-term memory. *Neural computation*, 9, 8 (1997), 1735–1780. (cited on page 8)

HUANG, G.; LIU, Z.; VAN DER MAATEN, L.; AND WEINBERGER, K. Q., 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700–4708. (cited on pages 5, 6, and 7)

HUANG, G.; LIU, Z.; WEINBERGER, K. Q.; AND VAN DER MAATEN, L., 2016. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, (2016). (cited on pages 35 and 45)

HUANG, X.; LIU, M.-Y.; BELONGIE, S.; AND KAUTZ, J., 2018. Multimodal unsupervised image-to-image translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 172–189. (cited on page 66)

IOFFE, S. AND SZEGEDY, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, (2015). (cited on page 100)

IONESCU, C.; PAPAVA, D.; OLARU, V.; AND SMINCHISESCU, C., 2014. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36, 7 (Jul 2014), 1325–1339. (cited on pages 67 and 83)

JAIN, A.; KOPPULA, H. S.; RAGHAVAN, B.; SOH, S.; AND SAXENA, A., 2015. Car that knows before you do: Anticipating maneuvers via learning temporal driving models. In *Proceedings of the IEEE International Conference on Computer Vision*, 3182–3190. (cited on page 38)

JAIN, A.; KOPPULA, H. S.; SOH, S.; RAGHAVAN, B.; SINGH, A.; AND SAXENA, A., 2016a. Brain4cars: Car that knows before you do via sensory-fusion deep learning architecture. *arXiv preprint arXiv:1601.00740*, (2016). (cited on pages 37, 38, 39, 41, 42, and 43)

JAIN, A.; SINGH, A.; KOPPULA, H. S.; SOH, S.; AND SAXENA, A., 2016b. Recurrent neural networks for driver activity anticipation via sensory-fusion architecture. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 3118–3125. IEEE. (cited on pages 15 and 38)
JAIN, A.; SINGH, A.; KOPPULA, H. S.; SOH, S.; AND SAXENA, A., 2016c. Recurrent neural networks for driver activity anticipation via sensory-fusion architecture. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 3118–3125. IEEE.  (cited on pages 17, 24, 25, 26, 27, and 28)

JAIN, A.; ZAMIR, A. R.; SAVARESE, S.; AND SAXENA, A., 2016d. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5308–5317. (cited on pages 60, 72, 77, 83, and 86)

JAIN, M.; VAN GEMERT, J.; JÉGOU, H.; BOUTHEMY, P.; AND SNOEK, C. G., 2014. Action localization with tubelets from motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 740–747. (cited on pages 16 and 18)

JHUANG, H.; GALL, J.; ZUFFI, S.; SCHMID, C.; AND BLACK, M. J., 2013a. Towards understanding action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, 3192–3199.  (cited on pages 17 and 23)

JHUANG, H.; GALL, J.; ZUFFI, S.; SCHMID, C.; AND BLACK, M. J., 2013b. Towards understanding action recognition. In *International Conf. on Computer Vision (ICCV)*, 3192–3199.  (cited on page 45)

KARPATHY, A.; TODERICI, G.; SHETTY, S.; LEUNG, T.; SUKTHANKAR, R.; AND FEI-FEI, L., 2014a. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 1725–1732.  (cited on pages 5 and 16)

KARPATHY, A.; TODERICI, G.; SHETTY, S.; LEUNG, T.; SUKTHANKAR, R.; AND FEI-FEI, L., 2014b. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 1725–1732.  (cited on pages 18, 30, and 32)

KAY, W.; CARREIRA, J.; SIMONYAN, K.; ZHANG, B.; HILLIER, C.; VIJAYANARASIMHAN, S.; VIOLA, F.; GREEN, T.; BACK, T.; NATSEV, P.; ET AL., 2017. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, (2017).  (cited on page 42)

KIM, Y.; WISEMAN, S.; MILLER, A. C.; SONTAG, D.; AND RUSH, A. M., 2018. Semi-amortized variational autoencoders. *arXiv preprint arXiv:1802.02550*, (2018).  (cited on pages 77 and 96)

KINGMA, D. P. AND BA, J., 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, (2014).  (cited on page 68)

KINGMA, D. P. AND WELLING, M., 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, (2013).  (cited on pages 10, 12, 64, 80, 93, 95, and 99)

KLINGELSCHMITT, S.; DAMEROW, F.; WILBERT, V.; AND EGGERT, J., 2016. Probabilistic situation assessment framework for multiple, interacting traffic participants in generic traffic scenes. In *Intelligent Vehicles Symposium (IV), 2016 IEEE*, 1141–1148. IEEE.  (cited on page 38)
KONG, Y.; KIT, D.; AND FU, Y., 2014. A discriminative model with multiple temporal scales for action prediction. In European Conference on Computer Vision, 596–611. Springer. (cited on page 17)

KOIJ, J. F. P.; SCHNEIDER, N.; FLOHR, F.; AND GAVRILA, D. M., 2014a. Context-based pedestrian path prediction. In European Conference on Computer Vision, 618–633. Springer. (cited on page 39)

KOIJ, J. F. P.; SCHNEIDER, N.; FLOHR, F.; AND GAVRILA, D. M., 2014b. Context-based pedestrian path prediction. In European Conference on Computer Vision, 618–633. Springer. (cited on pages 41 and 49)

KOPPULA, H. S. AND SAXENA, A., 2016. Anticipating human activities using object affordances for reactive robotic response. IEEE transactions on pattern analysis and machine intelligence, 38, 1 (2016), 14–29. (cited on page 37)

KRIZHEVSKY, A.; SUTSKEVER, I.; AND HINTON, G. E., 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, 1097–1105. (cited on pages 5, 6, and 100)

KUEHNE, H.; JHUANG, H.; GARROTE, E.; POGGIO, T.; AND SERRE, T., 2011. HMDB: a large video database for human motion recognition. In Proceedings of the International Conference on Computer Vision (ICCV). (cited on page 42)

KULKARNI, T. D.; WHITNEY, W. F.; KOHLI, P.; AND TENENBAUM, J., 2015. Deep convolutional inverse graphics network. In Advances in neural information processing systems, 2539–2547. (cited on page 60)

KULLBACK, S. AND LEIBLER, R. A., 1951. On information and sufficiency. The annals of mathematical statistics, 22, 1 (1951), 79–86. (cited on page 10)

KUMAR, M.; BABAEIZADEH, M.; ERHAN, D.; FINN, C.; LEVINE, S.; DING, L.; AND KINGMA, D., 2019. Videoflow: A flow-based generative model for video. arXiv preprint arXiv:1903.01434, (2019). (cited on page 97)

KUNDU, J. N.; GOR, M.; AND BABU, R. V., 2018. Bihmp-gan: Bidirectional 3d human motion prediction gan. arXiv preprint arXiv:1812.02591, (2018). (cited on pages 59, 60, 61, 75, 76, and 78)

LARSEN, A. B. L.; SØNDERBY, S. K.; LAROCHELLE, H.; AND WINther, O., 2015. Autoencoding beyond pixels using a learned similarity metric. arXiv preprint arXiv:1512.09300, (2015). (cited on page 60)
Laviers, K.; Sukthankar, G.; Aha, D. W.; Molineaux, M.; Darken, C.; et al., 2009. Improving offensive performance through opponent modeling. In Proceedings of AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 58–63. (cited on pages 25 and 31)

Li, B.; He, J.; Neubig, G.; Berg-Kirkpatrick, T.; and Yang, Y., 2019. A surprisingly effective fix for deep latent variable modeling of text. arXiv preprint arXiv:1909.00868, (2019). (cited on pages 77, 96, and 101)

Li, C.; Zhang, Z.; Sun Lee, W.; and Hee Lee, G., 2018a. Convolutional sequence to sequence model for human dynamics. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 5226–5234. (cited on pages 61 and 83)

Li, C.; Zhong, Q.; Xie, D.; and Pu, S., 2018b. Co-occurrence feature learning from skeleton data for action recognition and detection with hierarchical aggregation. arXiv preprint arXiv:1804.06055, (2018). (cited on pages 37, 38, 70, 84, and 85)

Li, X.; Li, L.; Flohr, F.; Wang, J.; Xiong, H.; Bernhard, M.; Pan, S.; Gavrila, D. M.; and Li, K., 2017. A unified framework for concurrent pedestrian and cyclist detection. IEEE transactions on intelligent transportation systems, 18, 2 (2017), 269–281. (cited on pages 37 and 38)

Li, Y.; Li, W.; Mahadevan, V.; and Vasconcelos, N., 2016. Vlad3: Encoding dynamics of deep features for action recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1951–1960. (cited on pages 16, 18, 30, and 32)

Liebner, M.; Ruhhammer, C.; Klanner, F.; and Stiller, C., 2013. Generic driver intent inference based on parametric models. In Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on, 268–275. IEEE. (cited on page 37)

Lin, M.; Chen, Q.; and Yan, S., 2013. Network in network. In arXiv preprint arXiv:1312.4400. (cited on page 21)

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L., 2014. Microsoft coco: Common objects in context. In European conference on computer vision, 740–755. Springer. (cited on page 87)

Lin, X. and Amer, M. R., 2018. Human motion modeling using dvgans. arXiv preprint arXiv:1804.10652, (2018). (cited on pages 59, 61, 76, and 78)

Liu, J.; Shahroudy, A.; Xu, D.; and Wang, G., 2016. Spatio-temporal lstm with trust gates for 3d human action recognition. In European Conference on Computer Vision, 816–833. Springer. (cited on pages 16 and 18)

Liu, X.; Gao, J.; Celikyilmaz, A.; Carin, L.; et al., 2019. Cyclical annealing schedule: A simple approach to mitigating kl vanishing. arXiv preprint arXiv:1903.10145, (2019). (cited on pages 77 and 96)
LONG, J.; SHELHAMER, E.; AND DARRELL, T., 2015. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, 3431–3440. (cited on page 5)

LUCAS, T. AND VERBEEK, J., 2018. Auxiliary guided autoregressive variational autoencoders. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 443–458. Springer. (cited on pages 77 and 96)

MA, S.; SIGAL, L.; AND SCLAROFF, S., 2016. Learning activity progression in lstms for activity detection and early detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1942–1950. (cited on pages 15, 17, 24, 25, 26, 30, 43, and 54)

MAATEN, L. V. D. AND HINTON, G., 2008. Visualizing data using t-sne. Journal of machine learning research, 9, Nov (2008), 2579–2605. (cited on page 85)

MAO, W.; LIU, M.; SALZMANN, M.; AND LI, H., 2019a. Learning trajectory dependencies for human motion prediction. arXiv preprint arXiv:1908.05436, (2019). (cited on pages 60, 61, and 72)

MAO, W.; LIU, M.; SALZMANN, M.; AND LI, H., 2019b. Learning trajectory dependencies for human motion prediction. In ICCV. (cited on pages 75, 77, 83, and 86)

MARTINEZ, J.; BLACK, M. J.; AND ROMERO, J., 2017. On human motion prediction using recurrent neural networks. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4674–4683. IEEE. (cited on pages 59, 60, 72, 74, 75, 77, 83, and 86)

MARUF, S.; SALEH, F.; AND HAFFARI, G., 2019. A survey on document-level machine translation: Methods and evaluation. arXiv preprint arXiv:1912.08494, (2019). (cited on page 8)

MORRIS, B.; DOSHI, A.; AND TRIVEDI, M., 2011. Lane change intent prediction for driver assistance: On-road design and evaluation. In Intelligent Vehicles Symposium (IV), 2011 IEEE, 895–901. IEEE. (cited on page 38)

NOH, H.; HONG, S.; AND HAN, B., 2015. Learning deconvolution network for semantic segmentation. In Proceedings of the IEEE international conference on computer vision, 1520–1528. (cited on page 5)

ohn-bar, e.; martin, s.; tawari, a.; and trivedi, m. m., 2014. Head, eye, and hand patterns for driver activity recognition. In Pattern Recognition (ICPR), 2014 22nd International Conference on, 660–665. IEEE. (cited on page 38)

olabiyi, o.; martinson, e.; chintalapudi, v.; and guo, r., 2017. Driver action prediction using deep (bidirectional) recurrent neural network. arXiv preprint arXiv:1706.02257, (2017). (cited on pages 38 and 41)
ONEATA, D.; REVAUD, J.; VERBEEK, J.; AND SCHMID, C., 2014. Spatio-temporal object detection proposals. In *European Conference on Computer Vision*, 737–752. Springer. (cited on pages 16 and 18)

PASCANU, R.; MIKOLOV, T.; AND BENGIO, Y., 2013. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, 1310–1318. (cited on page 68)

PASZKE, A.; GROSS, S.; CHINTALA, S.; CHANAN, G.; YANG, E.; DeVITO, Z.; LIN, Z.; DESMAISON, A.; ANTIGA, L.; AND LERER, A., 2017. Automatic differentiation in pytorch. (2017). (cited on page 68)

PAVLLO, D.; FEICHTENOFER, C.; AULI, M.; AND GRANGIER, D., 2019. Modeling human motion with quaternion-based neural networks. *arXiv preprint arXiv:1901.07677*, (2019). (cited on pages 59, 60, 64, 74, 75, and 77)

PAVLLO, D.; GRANGIER, D.; AND AULI, M., 2018. Quaternion: A quaternion-based recurrent model for human motion. *arXiv preprint arXiv:1805.06485*, (2018). (cited on pages 59, 60, 64, 74, 75, 77, and 83)

PENTLAND, A. AND LIU, A., 1999. Modeling and prediction of human behavior. *Neural computation*, 11, 1 (1999), 229–242. (cited on page 44)

POOL, E. A.; KOOIJ, J. F.; AND GAVRILA, D. M., 2017. Using road topology to improve cyclist path prediction. In *Intelligent Vehicles Symposium (IV), 2017 IEEE*, 289–296. IEEE. (cited on pages 38, 39, and 40)

RAMANATHAN, V.; HUANG, J.; ABU-EL-HAIJA, S.; GORBAN, A.; MURPHY, K.; AND FEI-FEI, L., 2016. Detecting events and key actors in multi-person videos. In *Proceedings of the IEEE Conference on ComputerVision and Pattern Recognition*, 3043–3053. (cited on page 37)

RASOLLI, A.; KOTSERUBA, I.; AND TSOTSOS, J. K., 2017. Agreeing to cross: How drivers and pedestrians communicate. *arXiv preprint arXiv:1702.03555*, (2017). (cited on pages 38, 39, 40, 42, and 49)

RAZAVI, A.; OORD, A. V. D.; POOLE, B.; AND VINYALS, O., 2019. Preventing posterior collapse with delta-vaes. *arXiv preprint arXiv:1901.03416*, (2019). (cited on pages 77 and 96)

REDMON, J.; DIVVALA, S.; GIRSHICK, R.; AND FARHADI, A., 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 779–788. (cited on page 5)

REN, S.; HE, K.; GIRSHICK, R.; AND SUN, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, 91–99. (cited on page 5)
Richter, S. R.; Hayder, Z.; and Koltun, V., 2017. Playing for benchmarks. In *International Conference on Computer Vision*. IEEE. (cited on pages 39, 40 and 42)

Richter, S. R.; Vineet, V.; Roth, S.; and Koltun, V., 2016. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision*, 102–118. Springer. (cited on pages 40 and 42)

Rockstar-Games, 2018a. *Grand Theft Auto V: PC single-player mods*. [http://tinyurl.com/yc8kq7vn](http://tinyurl.com/yc8kq7vn) (cited on page 38)

Rockstar-Games, 2018b. Policy on posting copyrighted Rockstar Games material. [http://tinyurl.com/yc8kq7vn](http://tinyurl.com/yc8kq7vn) (cited on page 39)

Rodriguez, C.; Fernando, B.; and Li, H., 2018. Action anticipation by predicting future dynamic images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 0–0. (cited on page 17)

Rodriguez, M. D.; Ahmed, J.; and Shah, M., 2008. Action mach a spatio-temporal maximum average correlation height filter for action recognition. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 1–8. IEEE. (cited on page 42)

Ros, G.; Sellart, L.; Villalonga, G.; Maidanik, E.; Molero, F.; Garcia, M.; Cedeño, A.; Perez, F.; Ramirez, D.; Escobar, E.; et al., 2017. Semantic segmentation of urban scenes via domain adaptation of synthia. In *Domain Adaptation in Computer Vision Applications*, 227–241. Springer. (cited on page 42)

Rueda-Domingo, P., T and Lardelli-Claret; Luna del Castillo, J.; Jimenez-Moleon, J.; Garcia-Martin, M.; and Bueno-Cavanillas, A., 2004. The influence of passengers on the risk of the driver causing a car collision in Spain: Analysis of collisions from 1990 to 1999. *Accident Analysis and Prevention*, (2004). (cited on page 28)

Ryoo, M.; Chen, C.-C.; Aggarwal, J.; and Roy-Chowdhury, A., 2010. An overview of contest on semantic description of human activities (sdha) 2010. In *Recognizing Patterns in Signals, Speech, Images and Videos*, 270–285. Springer. (cited on pages 17, 23, 25 and 31)

Ryoo, M. S., 2011. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, 1036–1043. IEEE. (cited on pages 15, 17, 25, 31 and 42)

Ryoo, M. S. and Aggarwal, J. K., 2009. Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities. In *Computer vision, 2009 IEEE 12th international conference on*, 1593–1600. IEEE. (cited on pages 17 and 42)

Sadegh Aliakbarian, M.; Sadat Saleh, F.; Salzmann, M.; Fernando, B.; Pettersson, L.; and Andersson, L., 2017. Encouraging lstms to anticipate actions very early. In *Proceedings of the IEEE International Conference on Computer Vision*, 280–289. (cited on pages 3, 37, 43, 51, 53 and 54)
SALEH, F.; ALIAKBARIAN, M. S.; SALZMANN, M.; PETERSSON, L.; GOULD, S.; AND ALVAREZ, J. M., 2016. Built-in foreground/background prior for weakly-supervised semantic segmentation. In European Conference on Computer Vision, 413–432. Springer. (cited on page 5)

SALEH, F.; ALIAKBARIAN, S.; SALZMANN, M.; AND GOULD, S., 2020. Artist: Autoregressive trajectory inpainting and scoring for tracking. arXiv preprint arXiv:2004.07482, (2020). (cited on page 88)

SALEH, F. S.; ALIAKBARIAN, M. S.; SALZMANN, M.; AND GOULD, S., 2017a. Bringing background into the foreground: Making all classes equal in weakly-supervised video semantic segmentation. In 2017 IEEE international conference on computer vision (ICCV), 2125–2135. IEEE. (cited on page 5)

SALEH, F. S.; ALIAKBARIAN, M. S.; SALZMANN, M.; PETERSSON, L.; AND ALVAREZ, J. M., 2018. Effective use of synthetic data for urban scene semantic segmentation. In European Conference on Computer Vision, 86–103. Springer. (cited on page 21)

SALEH, F. S.; ALIAKBARIAN, M. S.; SALZMANN, M.; PETERSSON, L.; AND ALVAREZ, J. M., 2017b. Incorporating network built-in priors in weakly-supervised semantic segmentation. IEEE transactions on pattern analysis and machine intelligence, 40, 6 (2017), 1382–1396. (cited on page 5)

SALIMANS, T.; GOODFELLOW, I.; ZAREMBA, W.; CHEUNG, V.; RADFORD, A.; AND CHEN, X., 2016. Improved techniques for training gans. In Advances in neural information processing systems, 2234–2242. (cited on page 66)

SCHULZ, A. T. AND STIEFELHAGEN, R., 2015. A controlled interactive multiple model filter for combined pedestrian intention recognition and path prediction. In Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on, 173–178. IEEE. (cited on pages 38 and 40)

SEMENIUTA, S.; SEVERYN, A.; AND BARTH, E., 2017. A hybrid convolutional variational autoencoder for text generation. arXiv preprint arXiv:1702.02390, (2017). (cited on pages 77 and 96)

SHEN, T.; OTT, M.; AULI, M.; AND RANZATO, M., 2019. Mixture models for diverse machine translation: Tricks of the trade. arXiv preprint arXiv:1902.07816, (2019). (cited on page 101)

SHI, Y.; FERNANDO, B.; AND HARTLEY, R., 2018. Action anticipation with rbf kernelized feature mapping rnn. In Proceedings of the European Conference on Computer Vision (ECCV), 301–317. (cited on page 74)

SIGURDSSON, G. A.; VAROL, G.; WANG, X.; FARHADI, A.; LAPTEV, I.; AND GUPTA, A., 2016. Hollywood in homes: Crowdsourcing data collection for activity understanding. In European Conference on Computer Vision, 510–526. Springer. (cited on page 42)
SIMONYAN, K. AND ZISSERMAN, A., 2014a. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, 568–576. (cited on page 5)

SIMONYAN, K. AND ZISSERMAN, A., 2014b. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, 568–576. (cited on pages 16, 17, 18, 19, 30, 31, and 32)

SIMONYAN, K. AND ZISSERMAN, A., 2014c. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, (2014). (cited on pages 5 and 6)

SIMONYAN, K. AND ZISSERMAN, A., 2014d. Very deep convolutional networks for large-scale image recognition. In *arXiv preprint arXiv:1409.1556*. (cited on page 20)

SMITH, R., 2007. An overview of the tesseract ocr engine. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, vol. 2, 629–633. IEEE. (cited on page 40)

SOHN, K.; LEE, H.; AND YAN, X., 2015. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, 3483–3491. (cited on pages 12 and 61)

SOOMRO, K.; IDREES, H.; AND SHAH, M., 2016a. Online localization and prediction of actions and interactions. *arXiv preprint arXiv:1612.01194*, (2016). (cited on pages xxi, 13, 17, 24, 25, 30, 31, 42, and 54)

SOOMRO, K.; IDREES, H.; AND SHAH, M., 2016b. 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*, 2648–2657. (cited on pages xxi, 13, 17, 24, 25, 30, 31, 37, 42, and 54)

SOOMRO, K.; ZAMIR, A. R.; AND SHAH, M., 2012a. Ucf101: A dataset of 101 human actions classes from videos in the wild. (cited on pages 17 and 23)

SOOMRO, K.; ZAMIR, A. R.; AND SHAH, M., 2012b. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, (2012). (cited on pages 42 and 45)

SRIVASTAVA, N.; MANSIMOV, E.; AND SALAKHUTDINOV, R., 2015. Unsupervised learning of video representations using lstms. *CoRR, abs/1502.04681*, 2 (2015). (cited on pages 30 and 32)

SUTSEKEVER, I.; VINYALS, O.; AND LE, Q. V., 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112. (cited on page 8)

TAWARI, A.; SIVARAMAN, S.; TRIVEDI, M. M.; SHANNON, T.; AND TIPPELHOFER, M., 2014. Looking-in and looking-out vision for urban intelligent assistance: Estimation of driver attentive state and dynamic surround for safe merging and braking. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, 115–120. IEEE. (cited on page 38)
TOLSTIKHIN, I.; BOUSQUET, O.; GELLY, S.; AND SCHOLKOPF, B., 2017. Wasserstein auto-encoders. arXiv preprint arXiv:1711.01558, (2017). (cited on pages 77 and 96)

TRAN, D.; BOURDEV, L.; FERGUS, R.; TORRESANI, L.; AND PALURI, M., 2015a. Learning spatiotemporal features with 3d convolutional networks. In Proceedings of the IEEE international conference on computer vision, 4489–4497. (cited on page 16)

TRAN, D.; BOURDEV, L.; FERGUS, R.; TORRESANI, L.; AND PALURI, M., 2015b. Learning spatiotemporal features with 3d convolutional networks. In 2015 IEEE International Conference on Computer Vision (ICCV), 4489–4497. IEEE. (cited on pages 18 and 30)

TYCHSEN-SMITH, L. AND PETERSSON, L., 2017. Denet: Scalable real-time object detection with directed sparse sampling. In Proceedings of the IEEE international conference on computer vision, 428–436. (cited on page 5)

VAN DEN BERGH, M.; ROIG, G.; BOIX, X.; MANEN, S.; AND VAN GOOL, L., 2013. Online video seeds for temporal window objectness. In Proceedings of the IEEE International Conference on Computer Vision, 377–384. (cited on pages 16 and 18)

VAN DEN OORD, A.; VINYALS, O.; ET AL., 2017. Neural discrete representation learning. In Advances in Neural Information Processing Systems, 6306–6315. (cited on pages 77 and 96)

VOLVO, 2017. Volvo trucks safety report 2017. Technical report, Accident Research Team, Volvo Group. (cited on page 40)

VONDRICK, C.; PIRSIAVASH, H.; AND TORRALBA, A., 2016. Anticipating visual representations from unlabeled video. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 98–106. (cited on pages 18 and 37)

WALKER, J.; MARINO, K.; GUPTA, A.; AND HEBERT, M., 2017. The pose knows: Video forecasting by generating pose futures. In Computer Vision (ICCV), 2017 IEEE International Conference on, 3352–3361. IEEE. (cited on pages 59, 60, 61, 62, 67, 71, 73, 75, 76, 78, 82, 85, 86, and 97)

WALTISBERG, D.; YAO, A.; GALL, J.; AND VAN GOOL, L., 2010. Variations of a hough-voting action recognition system. In Recognizing patterns in signals, speech, images and videos, 306–312. Springer. (cited on page 31)

WANG, B.; ADELI, E.; KUANG CHIU, H.; HUANG, D.-A.; AND NIEBLES, J. C., 2019. Imitation learning for human pose prediction. arXiv preprint arXiv:1909.03449, (2019). (cited on page 72)

WANG, H. AND SCHMID, C., 2013a. Action recognition with improved trajectories. In Proceedings of the IEEE international conference on computer vision, 3551–3558. (cited on page 15)
WANG, H. AND SCHMID, C., 2013b. Action recognition with improved trajectories. In Proceedings of the IEEE International Conference on Computer Vision, 3551–3558. (cited on pages 16, 18, and 30)

WANG, L.; QIAO, Y.; AND TANG, X., 2015. Action recognition with trajectory-pooled deep-convolutional descriptors. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4305–4314. (cited on pages 16, 18, and 30)

WANG, L.; QIAO, Y.; TANG, X.; AND VAN GOOL, L., 2016a. Actionness estimation using hybrid fully convolutional networks. arXiv preprint arXiv:1604.07279, (2016). (cited on pages xxi, 16, 18, 30, and 31)

WANG, L.; XIONG, Y.; WANG, Z.; QIAO, Y.; LIN, D.; TANG, X.; AND VAN GOOL, L., 2016b. Temporal segment networks: Towards good practices for deep action recognition. In European conference on computer vision, 20–36. Springer. (cited on pages 5, 15, and 16)

WANG, L.; XIONG, Y.; WANG, Z.; QIAO, Y.; LIN, D.; TANG, X.; AND VAN GOOL, L., 2016c. Temporal segment networks: towards good practices for deep action recognition. In European Conference on Computer Vision, 20–36. Springer. (cited on pages 16, 18, 32, 37, and 46)

WANG, X.; FARHADI, A.; AND GUPTA, A., 2016d. Actions ~ transformations. In IEEE International Conference on Computer Vision and Pattern Recognition CVPR. IEEE. (cited on page 18)

WANG, X. AND JI, Q., 2017. Hierarchical context modeling for video event recognition. IEEE transactions on pattern analysis and machine intelligence, 39, 9 (2017), 1770–1782. (cited on page 37)

WILLIAMS, R. J. AND ZIPSER, D., 1989. A learning algorithm for continually running fully recurrent neural networks. Neural computation, 1, 2 (1989), 270–280. (cited on pages 68 and 99)

WU, C.-Y.; FEICHTENHOFER, C.; FAN, H.; HE, K.; KRAHENBUHL, P.; AND GIRSHICK, R., 2019. Long-term feature banks for detailed video understanding. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 284–293. (cited on page 5)

WU, R.; YAN, S.; SHAN, Y.; DANG, Q.; AND SUN, G., 2015. Deep image: Scaling up image recognition. arXiv preprint arXiv:1501.02876, 7, 8 (2015). (cited on page 24)

XU, J. AND DURRETT, G., 2018. Spherical latent spaces for stable variational autoencoders. arXiv preprint arXiv:1808.10805, (2018). (cited on pages 77 and 96)

YAN, X.; RASTOGI, A.; VILLENGAS, R.; SUNKAVALLI, K.; SHECHTMAN, E.; HADAP, S.; YUMER, E.; AND LEE, H., 2018. Mt-vae: Learning motion transformations to generate multimodal human dynamics. In European Conference on Computer Vision, 276–293. Springer. (cited on pages xvii, xix, 59, 61, 62, 66, 67, 68, 70, 71, 73, 76, 78, 79, 84, 85, 86, and 96)
YAN, X.; YANG, J.; SOHN, K.; AND LEE, H., 2016. Attribute2image: Conditional image generation from visual attributes. In European Conference on Computer Vision, 776–791. Springer. (cited on page 60)

YANG, D.; HONG, S.; JANG, Y.; ZHAO, T.; AND LEE, H., 2019. Diversity-sensitive conditional generative adversarial networks. In International Conference on Learning Representations. https://openreview.net/forum?id=rJliMh09F7 (cited on pages 66 and 84)

YANG, Z.; HU, Z.; SALAKHUTDINOV, R.; AND BERG-KIRKPATRICK, T., 2017. Improved variational autoencoders for text modeling using dilated convolutions. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, 3881–3890. JMLR. org. (cited on pages 77 and 96)

YU, G. AND YUAN, J., 2015. Fast action proposals for human action detection and search. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1302–1311. (cited on pages 16 and 18)

YU, G.; YUAN, J.; AND LIU, Z., 2012. Predicting human activities using spatio-temporal structure of interest points. In Proceedings of the 20th ACM international conference on Multimedia, 1049–1052. ACM. (cited on page 17)

YU, T.-H.; KIM, T.-K.; AND CIPOLLA, R., 2010. Real-time action recognition by spatiotemporal semantic and structural forests. In BMVC, vol. 2, 6. (cited on page 31)

YUAN, F.; PRINET, V.; AND YUAN, J., 2010. Middle-level representation for human activities recognition: the role of spatio-temporal relationships. In ECCV, 168–180. Springer. (cited on page 31)

YUAN, Y. AND KITANI, K., 2019. Diverse trajectory forecasting with determinantal point processes. arXiv preprint arXiv:1907.04967, (2019). (cited on pages 66 and 84)

YUE-HEI NG, J.; HAUSKNECHT, M.; VIJAYANARASIMHAN, S.; VINYALS, O.; MONGA, R.; AND TODERICI, G., 2015. Beyond short snippets: Deep networks for video classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4694–4702. (cited on pages 30 and 32)

ZEILER, M. D. AND FERGUS, R., 2014. Visualizing and understanding convolutional networks. In European Conference on Computer Vision, 818–833. Springer. (cited on page 21)

ZHANG, B.; WANG, L.; WANG, Z.; QIAO, Y.; AND WANG, H., 2016. Real-time action recognition with enhanced motion vector cnns. arXiv preprint arXiv:1604.07669, (2016). (cited on page 30)

ZHANG, W.; ZHU, M.; AND DERPANIS, K. G., 2013. From actemes to action: A strongly-supervised representation for detailed action understanding. In Proceedings of the IEEE International Conference on Computer Vision, 2248–2255. (cited on page 83)
ZHao, H.; Shi, J.; Qi, X.; Wang, X.; and Jia, J., 2017a. Pyramid scene parsing network. In Proceedings of the IEEE conference on computer vision and pattern recognition, 2881–2890. (cited on page 5)

ZHao, S.; Song, J.; and Ermon, S., 2017b. Infovae: Information maximizing variational autoencoders. arXiv preprint arXiv:1706.02262, (2017). (cited on pages 77 and 96)

ZHao, X.; Li, X.; Pang, C.; Zhu, X.; and Sheng, Q. Z., 2013. Online human gesture recognition from motion data streams. In Proceedings of the 21st ACM international conference on Multimedia, 23–32. ACM. (cited on page 17)

ZHou, B.; Khosla, A.; Lapedriza, A.; Oliva, A.; and Torralba, A., 2016a. Learning deep features for discriminative localization. In Proceedings of the IEEE conference on computer vision and pattern recognition, 2921–2929. (cited on page 6)

ZHou, B.; Khosla, A.; Lapedriza, A.; Oliva, A.; and Torralba, A., 2016b. Learning deep features for discriminative localization. In IEEE International Conference on Computer Vision and Pattern Recognition CVPR. (cited on page 21)

Zyner, A.; Worrall, S.; Ward, J.; and Nebot, E., 2017. Long short term memory for driver intent prediction. In Intelligent Vehicles Symposium (IV), 2017 IEEE, 1484–1489. IEEE. (cited on page 38)