LEARNING EFFICIENT TEMPORAL INFORMATION IN DEEP NETWORKS: FROM THE VIEWPOINTS OF APPLICATIONS AND MODELING

BY

CU KHOI NGUYEN MAC

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2021

Urbana, Illinois

Doctoral Committee:

Professor Minh N. Do, Chair
Professor David A. Forsyth
Professor Mark A. Hasegawa-Johnson
Assistant Professor Alexander G. Schwing
Assistant Professor Saurabh Gupta

ABSTRACT

With the introduction of deep learning, machine learning has dominated several technology areas, giving birth to high-performance applications that can even challenge human-level accuracy. However, the complexity of deep models is also exploding as a by-product of the revolution of machine learning. Such enormous model complexity has raised the new challenge of improving the efficiency in deep models to reduce deployment expense, especially for systems with high throughput demands or devices with limited power. The dissertation aims to improve the efficiency of temporal-sensitive deep models in four different directions. *First*, we develop a bandwidth extension mapping to avoid deploying multiple speech recognition systems corresponding to wideband and narrowband signals. *Second*, we apply a multi-modality approach to compensate for the performance of an excitement scoring system, where the input video sequences are aggressively down-sampled to reduce throughput. *Third*, we formulate the motion feature in the feature space by directly inducing the temporal information from intermediate layers of deep networks instead of relying on an additional optical flow stream. *Finally*, we model a spatiotemporal sampling network inspired by the human visual perception mechanism to reduce input frames and regions adaptively.

For my Mom Hằng, my Dad Lâm, and my Heart Thủy,
Who were always by my side through the toughest time.

With dearest love.

ACKNOWLEDGMENTS

I want to take a moment to thank everyone who helped me on this journey.

First and foremost, I wish to express my deepest and most sincere gratitude to my advisor, Prof. Minh N. Do, for the continuous guidance and support throughout my Ph.D. study and research, and for his enthusiasm, motivation, patience, and immense knowledge.

I appreciate the rest of my dissertation committee: Prof. David A. Forsyth, Prof. Mark A. Hasegawa-Johnson, Prof. Alexander G. Schwing, and Prof. Saurabh Gupta, for their encouragement, insightful comments, and challenging questions.

My sincere thanks also go to Dr. Dhiraj Joshi, Dr. Xiaodong Cui, Dr. JuneChul Roh, and Dr. Minh P. Vo for their mentoring and collaboration through some of proudest projects.

To my dear family, I am thankful for their unconditional love and support, without which none of my success would be possible.

Last but not least, I thank my fellow labmates in the Computational Imaging Group: Raymond Yeh, Teck Yian Lim, Vaishnavi Subramanian, Mona Zehni, Renan Rojas, Mary Dasso, Daniel Gonzales, Qian Jiang, Andy Lai, Spencer Markowitz, Corey Snyder, Trong Nguyen, Chen Chen, Benjamin Chidester, Greg Meyer, Kirk Busche, Jason Nie, and Dario Aranguiz, for the stimulating discussions, and for all the fun we have had in the last five and a half years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ASR	Automatic Speech Recognition
BWE	Bandwidth Extension
CE	Cross Entropy
CNN	Convolutional Neural Network
DNN	Deep Neural Network
GAN	Generative Adversarial Network
LCDC	Locally-Consistent Deformable Convolution
LM	Language Model
MB	Mixed-bandwidth
NB	Narrowband
OCR	Optical Character Recognition
WB	Wideband
WER	Word-Error Rate

CHAPTER 1

INTRODUCTION

1.1 Motivation

Time is an intriguing concept that forwards the continuity of almost everything humans can perceive in life. Although we cannot travel back through time, re-experiencing past events is possible by inspecting recorded data. Thanks to the advance of technology, temporal information has been stored and shared effortlessly and widely under multiple forms, *e.g.*, sensor signals, audio, and videos. Such sequential data has been increasingly attracting attention from several pieces of research across multiple domains, such as computer vision, digital signal processing, and machine learning. In addition, the popularity and availability of time-series data have given rise to the studies of video surveillance, vehicle tracking, action recognition, action detection, and speech recognition.

However, temporal information is also a factor that increases the complexity of several deep learning models since traditional approaches usually treat different time frames with equivalent inferencing processes, while it is evident that consecutive samples contain significant redundancy. Therefore, it is exceptionally crucial to improve the efficiency in deep models that rely on temporal data.

The thesis introduces four different solutions to efficiency in time-related deep neural networks, categorized as application and modeling approaches. Regarding the *application* angle, we (1) propose a bandwidth extension mapping to reduce from having two different acoustic models to only one and (2) develop an excitement scoring framework with a low frame rate and use a multi-modality approach to compensate for the performance. From the *modeling* perspective, we (3) introduce a novel method to model the temporal information directly in feature space to avoid relying on the additional data stream of optical flow, and (4) formulate an adaptive spatiotemporal sampling model, inspired by the human visual perception, to only select the frames and regions of interest instead of the whole video.

1.2 Overview

In Chapter 2, we propose an application to improve efficiency of a fixed acoustic model. For automatic speech recognition (ASR), wideband (WB) and narrowband (NB) speech signals with different sampling rates typically use separate acoustic models. Therefore mixed-bandwidth (MB) acoustic modeling has important practical values for ASR system deployment. In this chapter, we investigate MB deep neural network acoustic modeling for ASR with large-scale training data (1,150 hours of WB data and 2,300 hours of NB data). Extensive experiments are conducted on eight diverse WB and NB test sets to study the performance of MB acoustic models by either upsampling or downsampling. In the meantime, we also propose a CNN-based discriminatively trained bandwidth extension (BWE) model with a VGG architecture to map the NB speech to WB speech. We show that the proposed BWE can improve the ASR performance of NB speech against the WB acoustic model. Furthermore, the MB acoustic model using WB speech and bandwidth extended NB speech can further improve the ASR performance. To deal with the large-scale training data, distributed training is carried out on multiple GPUs using synchronous data parallelism.

In Chapter 3, we direct our focus on an application of action recognition: automatically generating highlight videos to summarize exciting moments of a sport match. The production of sports highlight packages summarizing a game's most exciting moments is an essential task for broadcast media. Yet, it requires labor-intensive video editing. We propose a novel approach for auto-curating sports highlights, and demonstrate it to create a first of a kind, real-world system for the editorial aid of golf and tennis highlight reels. Our method fuses information from the players' reactions (action recognition such as high-fives and fist pumps), players' expressions (aggressive, tense, smiling and neutral), spectators (crowd cheering), commentator (tone of the voice and word analysis) and game analytics to determine the most interesting moments of a game. We accurately identify the start and end frames of key shot highlights with additional metadata, such as the player's name and the hole number, or analysts input allowing personalized content summarization and retrieval. In addition, we introduce new techniques for learning our classifiers with reduced manual training data annotation by exploiting the correlation of different modalities. Our work has been demonstrated at a major golf tournament (the 2017 Masters) and two major international tennis tournaments (the 2017 Wimbledon and US Open), successfully extracting high-

lights through the course of the sporting events [1, 2, 3]. 54% of the clips selected by our system overlapped with the official highlights reels for the 2017 Masters. Furthermore, user studies showed that 90% of the non-overlapping ones where of the same quality of the official clips for the 2017 Masters, while the automatic selection of clips for highlights of the 2017 Wimbledon and US Open agreed with human preferences 80% and 84.2% of the time, respectively.

In Chapter 4, we explore how to model temporal information more effectively, in the context of fine-grained actions. Fine-grained action detection is an important task with numerous applications in robotics and human-computer interaction. Existing methods typically utilize a two-stage approach including extraction of local spatiotemporal features followed by temporal modeling to capture long-term dependencies. While most recent papers have focused on the latter (long-temporal modeling), here, we focus on producing features capable of modeling fine-grained motion more efficiently. We propose a novel locally-consistent deformable convolution, which utilizes the change in receptive fields and enforces a local coherency constraint to capture motion information effectively. Our model jointly learns spatiotemporal features (instead of using independent spatial and temporal streams). The temporal component is learned from the feature space instead of pixel space, *e.g.*, optical flow. The produced features can be flexibly used in conjunction with other long-temporal modeling networks, *e.g.*, ST-CNN [4], DilatedTCN [5], and ED-TCN [5]. Overall, our proposed approach robustly outperforms the original long-temporal models on two fine-grained action datasets: 50 Salads [6] and GTEA [7], achieving F1 scores of 80.22% and 75.39% respectively.

In Chapter 5, we investigate adative sampling strategies in both spatial and temporal domains. Adaptive sampling that exploits the spatiotemporal redundancy in videos from scene cameras is an enabling technology for always-on action recognition on wearable devices with limited compute and battery. The commonly used fixed sampling strategy is not context-aware and may under-sample the audio-visual content. This adversely impacts both computation efficiency and accuracy. Inspired by concepts from the human visual perception mechanism, we introduce a novel adaptive spatiotemporal sampling scheme for egocentric action recognition. Our system pre-scans a global scene context at low-resolution and decides to skip or request high-resolution features at salient regions for more compute-intensive processing. The system is demonstrated on the dataset EPIC-KITCHENS and can signigficantly speed up inference times with a tolerable loss of accuracy.

CHAPTER 2

MIXED-BANDWIDTH FOR SPEECH RECOGNITION

2.1 Introduction

[1] Wideband (WB) and narrowband (NB) speech signals are two types of input signals that widely exist in speech-related applications. In automatic speech recognition (ASR), acoustic models are usually separately trained for WB and NB speech data given their distinct spectral characteristics under different sampling rates. From the system deployment's perspective, one acoustic model for both WB and NB speech would be greatly preferred. In this paper, we investigate mixed-bandwidth (MB) acoustic modeling using neural networks with deep architectures.

The goal of MB acoustic modeling is to converge the WB and NB speech to one bandwidth from which acoustic modeling is carried out. This could be accomplished either by downsampling or upsampling. In this paper we are interested in seeking answers to the following questions: (1) To converge to one bandwidth, which strategy is better, downsampling or upsampling? (2) For upsampling, how does the bandwidth extension (BWE) help in terms of ASR performance? (3) How does naive mixing compare to mixing with BWE?

Furthermore, we are interested in real-world cases where large amounts of WB and NB training data are available and their amounts may be unbalanced. Specifically, we investigate MB deep convolutional neural network (CNN) acoustic models using 1,150 hours of WB speech and 2,300 hours of NB speech. We evaluate the ASR performance of the acoustic models on a wide variety of WB and NB test sets collected from diverse ASR scenarios.

We also propose a discriminatively trained BWE based on CNNs with a VGG architecture to map the upsampled NB speech to WB speech. Different from most

[1] Chapter 2 has been published in *The 20th Annual Conference of the International Speech Communication Association (INTERSPEECH)* 2019 [8], Copyright ISCA.

of the BWE techniques in the literature where the BWE is typically treated as a regression problem and the mapping is estimated under a reconstructive objective function (*e.g.*, minimum mean squared error (MMSE)), the proposed BWE is treated as a classification problem and the mapping is directly optimized towards a given WB acoustic model under the cross-entropy (CE) objective function. Therefore, the BWE mapping is more closely connected to the final ASR performance, which is the ultimate goal of interest. We study the performance of the NB speech mapped by the proposed BWE against the WB CNN model and also that after mixing with WB and BWE-mapped NB speech.

Training deep CNN acoustic models using approximately 3,500 hours of speech data is computationally demanding. We resort to parallel computing for stochastic gradient descent (SGD) based network optimization with multiple GPUs. The system design and engineering consideration will be addressed in the system implementation.

The remainder of the paper is organized as follows. Sec. 2.2 will discuss the related work on MB acoustic modeling and BWE. Sec. 2.3 gives the mathematical formulation of proposed BWE. Sec. 2.4 is devoted to the system implementation including the model architectures and parallel computing. Experimental results are provided in Sec. 2.5 followed by a discussion and summary in Sec. 2.6.

2.2 Related Work

BWE has been an active research topic in communication and acoustics processing. NB speech signals, such as telephony speech signals, suffer from degraded quality and intelligibility due to the lack of high frequency spectral information eliminated by the low-pass band limitation of communication channels. Over the years, extensive research has been carried out on BWE to compensate this degradation so as to improve the speech quality and intelligibility [9, 10, 11, 12, 13, 14]. BWE aims to estimate the missing high frequency spectral components and, therefore, effectively "extend" the bandwidth of the speech signals.

Specific to ASR, given that deep neural networks (DNNs) are universal approximators [15, 16], DNN-based BWE approaches are well justified and have been widely explored using various deep architectures [13, 14, 17]. In [18], the NB data is used to leverage the training of WB acoustic models in a GMM-HMM framework and the missing components in upsampled NB speech features are

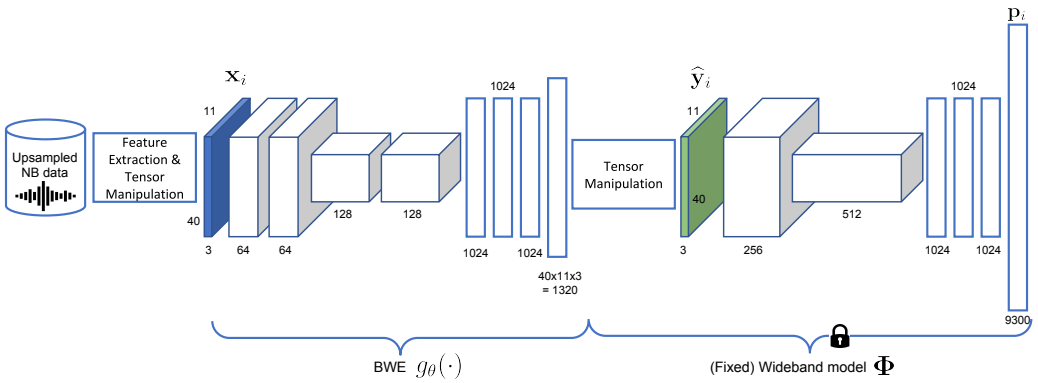


Figure 2.1: Illustration of the training of the BWE mapping. The mapping is realized as a CNN with a VGG architecture. Its output is connected to the WB CNN acoustic model after tensor manipulation. The WB CNN is fixed and the BWE CNN is optimized under the CE criterion.

dealt with using the expectation-maximization (EM) algorithm [19] for the MB GMM-HMM acoustic models. MB training in [20] follows a similar argument of [18] as a missing feature problem but the problem is addressed in a DNN-HMM framework where no explicit BWE is assumed. Acoustic models in both [18] and [20] are trained on a small amount training data ($<$ 100 hours). A joint MB training scheme is studied in [21] using 1,000 hours of WB data and 1,000 hours of NB data where a fully connected (FC) feedforward DNN is used to capture the BWE mapping from NB to WB speech with MMSE-based pretraining based on parallel speech data. The MB models are trained with various upsampling and downsampling scenarios.

In terms of the amount of training data (on the order of magnitude of 1,000 hours) and the variety of mixing strategies, the work reported in [21] is the most related to the work presented in this paper where the major difference is the way BWE is estimated and acoustic model architectures. In this paper we propose a discriminatively trained BWE using CNNs that assume a VGG architecture [22]. In our pilot experiments, we find that CNN-based BWE outperforms FC feedforward DNN-based BWE and discriminatively trained BWE outperforms MMSE-based BWE when all using logmel features. Furthermore, we use larger amounts of MB training data with an unbalanced distribution and multiple diverse test sets to avoid overtuning the system to specific test sets.

2.3 Bandwidth Extension

Let $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ denote a sequence of n NB features in certain feature domain, $\mathbf{x}_i \in \mathbb{R}^{d_x}$. We want to estimate a mapping function f_θ with some parameter θ to map the NB feature sequence $\mathbf{X}$ to a WB feature sequence $\hat{\mathbf{Y}} = \{\hat{\mathbf{y}}_1, \cdots, \hat{\mathbf{y}}_n\}$, $\hat{\mathbf{y}}_i \in \mathbb{R}^{d_y}$, where $\hat{\mathbf{y}}_i = f_\theta(\mathbf{x}_i)$. A loss function

$$\mathcal{L}(l_i, \hat{\mathbf{y}}_i) \triangleq \mathcal{L}(l_i, f_\theta(\mathbf{x}_i)) \tag{2.1}$$

is defined to measure the closeness of the mapped WB features $\hat{\mathbf{y}}_i$ and their labels l_i. We want to optimize the parameter θ such that it minimizes the following empirical risk:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(l_i, f_\theta(\mathbf{x}_i)). \tag{2.2}$$

Depending on whether the problem is viewed as a regression or classification problem, the labels l_i are chosen differently.

2.3.1 Loss Functions

In most cases, BWE is treated as a regression problem and the mapping function is estimated under the MMSE criterion as follows [21, 17]

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \| \mathbf{y}_i - f_\theta(\mathbf{x}_i) \|_2^2, \tag{2.3}$$

where the labels $l_i = \mathbf{y}_i$, which is the ground truth WB counterpart of the NB speech features. This requires parallel WB and NB data and is usually accomplished by downsampling the WB speech to create the feature pairs. The mapping functions are learned in a reconstructive way to minimize the L_2 distance between the mapped NB features and their WB counterparts.

Since ASR is a classification problem by nature, it is desirable to have the BWE mapping estimated with a matched objective. In this paper, we propose another way of estimating the BWE mapping function. Suppose we have a WB neural network acoustic model $\mathbf{\Phi}$ which takes the WB speech features as input and outputs the posterior probabilities p_{ik} with respect to context-dependent phone

classes after the softmax layer for feature i and class k. We use the BWE mapping function g_θ to map the upsampled NB speech features to WB features which are directly fed into the WB acoustic model $\mathbf{\Phi}$ to generate posteriors $p_i = \mathbf{\Phi}(g_\theta(\mathbf{x}_i))$ where $\mathbf{\Phi}$ is fixed and g_θ is subject to optimization. In this case, the mapping function is $f_\theta \triangleq \mathbf{\Phi} \circ g_\theta$.

The CE loss function is defined between the posterior probabilities p_i and the class labels l_i:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{n} \sum_i \mathcal{L}(l_i, \mathbf{\Phi}(g_\theta(\mathbf{x}_i))) = \underset{\theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i,k} l_{ik} \log \frac{1}{p_{ik}}. \tag{2.4}$$

The labels l_i are generated by aligning the upsampled NB features against the WB acoustic model. The training strategy is illustrated in Fig 2.1.

2.3.2 Mapping Functions

The mapping f_θ is usually selected from a certain function family and its parameters θ are optimized against the loss function. There are a variety of families of functions from which the mapping has been chosen for BWE, notably linear functions, Gaussian mixture models (GMMs) and HMMs [10]. Most recently, DNNs as universal approximators have been dominant to model the mapping. In this paper, the mapping function involves a composite of two CNNs. One is used to map the upsampled NB speech to WB which is subject to optimized and the other is an existing WB acoustic model which is fixed.

2.4 System Implementation

We describe the detail of our system implementation in this section, starting with the feature space of input signals, followed by the model architectures and parallel computing technique for training.

2.4.1 Feature Space

The WB speech is sampled at 16KHz while the NB speech at 8KHz. The input feature space consists of 40-dimensional logmel features after application of first

a global cepstral mean normalization (CMN) and then an utterance-based CMN. There are three input feature maps to the CNNs, the static logmel features and their delta and double delta, all with a temporal context of 11 frames. For upsampled NB speech signals, they go through the WB Mel filter banks after upsampling in the time domain, which gives rise to zeros in the outputs of the upper Mel filter bins (the zero-padding effect).

2.4.2 Models

Acoustic models CNN acoustic models are used for WB baseline, NB baseline and MB models, which have the same configuration. There are two convolutional layers and each convolutional layer is followed by a max-pooling layer. The first convolutional layer uses 5×5 kernels with a stride is 1×1 and padding 2×2. The second convolutional layer uses the same kernel, stride and padding sizes as those of the first convolutional layer. Both max-pooling layers use a kernel of 2×2 and stride of 2×2. On top of the convolutional and pooling layers are three FC layers with 1,024 hidden units. All activation functions are ReLU except the last FC layer which uses sigmoid. The output softmax layer has 9,300 output units. We investigate two model capacities in the experiments, one with 128 and 256 feature maps for the two respective convolutional layers and the other 256 and 512 feature maps.

BWE Models The BWE mapping network is also a CNN. The design of the network follows the VGG architecture that uses small convolutional kernels, small stride and small pooling kernels but, in the meantime, uses increased depth of the convolutional layers and reduced max-pooling layers. Specifically, we use four convolutional layers, two max-pooling layers and three FC layers. Every two convolutional layers are followed by one max-pooling layer. The first two convolutional layers use 3×3 kernels with a stride 1×1 and padding 1×1. The second two convolutional layers again use 3×3 kernels with a stride 1×1 and padding 1×1. The two max-pooling layers use 2×2 kernels with a stride 1×1. The three FC layers have 1,024 hidden units. All activation functions are ReLU except the last FC layer which uses tanh. We also investigate two model capacities, one with 64 feature maps for the two convolutional layers and 128 feature maps for the next two convolutional layers, and the other 128 and 256 features maps. They are indicated when reporting the results.

2.4.3 Parallel Computing

The networks are optimized under the CE criterion using the SGD algorithm. Learning rate starts as 0.01 for 10 epochs and is annealed by half every epoch for the next 10. We apply synchronous data parallelism on multi-GPUs (eight Nvidia v100s), within the same server, to accelerate training. To minimize parameters/-gradients copy, we remap each trainable layer's gradient buffer to one consecutive region upfront. For each iteration, after each learner finishes backward propagation, we use NCCL[23] to sum all learners' gradients, via an Allreduce call, back to each learners' gradients region, before weights update. By doing this, 8-GPU training achieves almost 8x speedup. In our experiments, each GPU receives a mini-batch of size 512, bringing the total batch size per iteration to 4,096. Another benefit of using multi-GPU is each learner effectively pre-load training data for others, which reduces I/O time to negligible.

2.5 Experimental Results

There are 1,150 hours of WB training data which consists of 420 hours of Broadcast News data, 450 hours of internal dictation data, 100 hours of meeting data, 140 hours of hospitality (travel and hotel reservation) data and 40 hours of accented data. There are 2,300 hours of NB training data which consists of 2,000 hours of Switchboard data and 300 hours of IBM call center data. In practice, we often find that the amounts of available WB and NB training data are unbalanced. In our case, the amounts of NB data is larger than that of the WB data.

We choose 4 WB test sets and 4 NB test sets for our experiments whose description and statistics are given in Table 2.1. The decoding vocabulary comprises of 250K words and the language model (LM) is a 4-gram LM with modified Kneser-Ney Smoothing consisting of 200M n-grams. The LM training data is selected from a broad variety of sources. The word error rates (WERs) of various models on the 8 test sets are shown in Table 2.2.

Baselines The WERs of WB and NB CNN baselines are shown in the first two rows in Table 2.2. The numbers of feature maps in the convolutional layers are indicated in the parentheses (*e.g.*, [128, 256] vs. [256, 512]). The underlined WERs indicate a change of sampling rate of the test data in order to be decoded by the acoustic model. For instance, the NB test sets are upsampled to decode against the

Table 2.1: WB and NB datasets used for evaluation.

		Description	Hours
WB	WS1	Dev04f test set from Broadcast News	2.21
	WS2	Commercial services help desk	0.34
	WS3	Hospitality domain 1	1.21
	WS4	Hospitality domain 2	0.81
NB	NS1	Hub5-2000 test set from Switchboard	2.10
	NS2	Technical support	4.09
	NS3	Commercial services help desk	3.01
	NS4	Multi-domain command and control	12.78

WB CNN and vice versa. Obviously, without any compensation, mismatched test data and acoustic model give rise to significant degradation of the performance. (17.2% $\rightarrow$ 23.6% for WB test sets and 17.8% $\rightarrow$ 25.0% for NB test sets on average.) We also carry out an experiment (third row) where only the WB training data is downsampled to train a CNN acoustic model which is used to decode the NB test sets. This model also gives a 25% average WER which is significantly worse than the matched training with NB data only. The performance gap may be due to the mismatched data but also to the mismatched domains.

Naive Mixing The second block of Table 2.2 presents the performance of MB models trained using the naive mixing strategy where WB data and upsampled NB data are mixed for the training of a CNN acoustic model. The CNN model with [128,256] feature maps obtains about the same average WER as the NB CNN baseline (17.8%) but slightly worse average WER than the WB CNN baseline (17.6%). Since the amount of training data increases after mixing, it is reasonable to increase the capacity of the MB model. With doubled feature maps in the convolutional layers ([256,512]) the MB CNN has a lower average WER (17.4%) on the NB test sets and only 0.3% absolute worse on the WB test sets. On the other hand, however, if the MB model is trained using NB data and downsampled WB data, its performance is far inferior on both WB and NB test sets. Therefore, from the table we can tell that upsampling the NB data and then mixing with the WB data appears to be a better strategy for MB modeling.

BWE The third block of Table 2.2 presents the performance of the proposed BWE approach. The VGG architecture of the BWE network is discriminatively trained with respect the WB CNN baseline model. With 64 feature maps in the first two convolutional layers and 128 feature maps in the second two convolutional layers, the BWE can significantly improve the average WER from 25.0%

Table 2.2: Word error rates (WERs) of WB, NB, and MB models on 8 test sets. The WERs are reported in 5 blocks from top to bottom representing various experimental conditions.

	WB					NB				
	WS1	WS2	WS3	WS4	**Avg**	NS1	NS2	NS3	NS4	**Avg**
WB baseline ([128,256])	15.4	14.9	9.1	29.2	**17.2**	25.1	39.0	13.7	22.0	**25.0**
NB baseline ([128,256])	21.3	16.8	15.6	40.5	**23.6**	13.5	25.0	12.8	19.7	**17.8**
WB↓ ([128,256])	17.9	17.5	10.9	33.9	**20.1**	26.0	39.0	12.9	21.9	**25.0**
NaiveMix (WB+NB↑,[128,256])	17.1	13.0	12.2	27.9	**17.6**	13.8	25.5	12.2	19.6	**17.8**
NaiveMix (WB+NB↑,[256,512])	16.5	12.8	11.8	28.8	**17.5**	13.4	25.2	11.8	19.2	**17.4**
NaiveMix (WB↓+NB,[128,256])	18.9	17.2	13.3	35.9	**21.3**	14.0	26.2	12.5	19.1	**18.0**
BWE ([64,128])	-	-	-	-	**-**	15.2	27.8	12.4	20.2	**18.9**
BWE ([128,256])	-	-	-	-	**-**	14.9	27.4	12.2	20.0	**18.6**
nBWE ([64,128])	-	-	-	-	**-**	15.0	27.6	12.4	19.6	**18.7**
Mix (WB+NB↑+BWE, [128,256])	16.5	14.2	10.1	29.9	**17.7**	13.6	25.6	12.2	19.7	**17.8**
Mix (WB+NB↑+BWE, [256,512])	16.0	14.6	9.7	29.9	**17.6**	13.7	25.4	12.2	19.6	**17.7**
Mix (WB+NB↑+nBWE, [128,256])	16.4	14.3	10.0	30.9	**17.9**	13.7	25.6	12.1	19.5	**17.7**
MixFT (WB+NB↑+BWE, [128,256])	16.6	14.8	9.9	29.2	**17.6**	13.6	25.6	12.5	19.7	**17.9**
MixFT (WB+NB↑+BWE, [256,512])	16.1	15.1	9.7	29.3	**17.6**	13.7	25.5	12.4	19.6	**17.8**
MixFT (WB+NB↑+nBWE, [128,256])	16.2	14.4	9.8	30.3	**17.7**	13.6	25.4	12.0	19.6	**17.7**

to 18.9%. If increase the network capacity with doubled feature maps, the average WER can be further improved to 18.6%. The last row of this block shows the performance of BWE trained in a denoising manner (denoted nBWE) where zero-mean Gaussian noise with variance of 0.01 is added to the input upsampled NB logmel features. It indicates that the denoising BWE can improve the generalization of the mapping and gives better performance under the same model capacity (18.9% $\rightarrow$ 18.7%). Note that the BWE achieves improvement across all the four NB test sets against the WB CNN model compared to simple upsampling. In some test sets, BWE also yields better performance than the NB baseline. In the following experiments, we will stick to the BWE CNN configuration with the [64, 128] feature maps.

Mixing with BWE The fourth block of Table 2.2 shows the performance of the mixing strategy of using WB data and BWE-mapped upsampled NB data from which the MB CNN models are trained. As shown by the table, the MB models further improve the WERs from the BWE alone. Using larger model capacity helps. Overall, it is slightly better than the naive mixing strategy when the model capacity is [128, 256] on the NB test sets. Denoising BWE helps the NB test sets but hurts the WB test sets.

Fine-tune The last block of Table 2.2 shows the WERs after fine-tuning the mixing with BWE. In the fine-tuning, the output of the BWE CNN is connected to the input of the MB CNN which is fixed. The BWE CNN is fine-tuned with a smaller learning rate for 6 epochs. After that, another MB CNN is trained using the fine-tuned BWE with a smaller learning rate (1/10 of the original learning rate) for another 6 epochs. The improvement given by this finetuning, as can be observed from the table, is only marginal.

2.6 Conclusion

As can be observed from the breakdown performance in Table 2.2, consistent improvements of one technique across all test sets are rarely observed. The conclusion drawn from one particular test set by one technique may not generalize to other test sets, although the average WERs can give us a good idea on the overall performance of certain technique. That is the reason we believe it is important to evaluate the BWE and mixed strategies extensively on diverse test sets from various domains and conditions.

In ASR applications, it is desirable to use one unified acoustic model for both WB and NB speech data. The experimental results in Sec. 2.5 show that it is possible to train a MB model with an appropriate strategy. Upsampling the NB data appears to be more helpful than downsampling the WB data. In addition, naive mixing with appropriately increased model capacity, due to increased training data after mixing, can give competitive ASR performance compared to separate WB and NB models individually. In our investigated case, the best MB model yields lower average WER than the NB baseline and only slight degradation over the WB baseline. Although naive mixing assumes no explicit BWE, one would expect the DNNs will implicitly learn the mapping from the zero-padded upper frequency bins of NB speech to WB speech.

In our pilot experiments, MMSE-based BWE turned out not to be very helpful. Given the space constraint, we did not report its performance. Compared to the MMSE-based BWE, the proposed discriminatively trained BWE that directly connects its training with the output of the WB acoustic model can yield decent gains. On top of that, the mixing with WB and BWE-mapped NB data can further improve the performance. With the same model capacity ([128, 256]), this mixing strategy slightly outperforms the naive mixing. However, increasing the capacity does not give the same amount of gains observed in the naive mixing. The proposed BWE approach can work with various existing DNN-based acoustic models for discriminative training. In principle the CNN-based BWE can connect to a wide variety of existing models with different architectures after appropriate tensor manipulation. We sense that the BWE performance can be further improved with better techniques and hopefully mixing with BWE can lead to better ASR performance too.

In summary, we have investigated in this paper the MB acoustic modeling for ASR with large-scale WB and NB training data. A CNN-based discriminatively trained BWE approach is proposed and studied for its effectiveness. Various mixing strategies through upsampling, downsampling and BWE are evaluated on diverse test sets.

CHAPTER 3

AUTOMATIC CURATION OF SPORTS HIGHLIGHTS

3.1 Introduction

[1] The tremendous growth of video data has resulted in a significant demand for tools that can accelerate and simplify the production of sports highlight packages for more effective browsing, searching, and content summarization. In a major professional golf tournament such as the Masters, for example–with 90 golfers playing multiple rounds over four days–video from every tee, every hole, and multiple camera angles can quickly add up to hundreds of hours of footage. Wimbledon, the oldest tennis tournament, hosts around 250 singles matches alone over the course of 13 days, producing several hundreds of hours of video. Yet, most of the process for producing highlight reels in those tournaments is still manual, labor-intensive, and not scalable.

In this paper, we present a novel approach for auto-curating sports highlights, showcasing its application for golf (the 2017 Masters) and tennis (the 2017 Wimbledon and US Open). Our approach combines information from the *player*, *spectators*, and the *commentator* to determine a game's most exciting moments. We measure the excitement level of video segments based on the following main multimodal markers:

- **Player reaction:** visual action recognition of player's celebration (such as high fives or fist pumps) and facial expression recognition;
- **Spectators:** audio measurement of crowd cheers;
- **Commentator:** excitement measure based on the commentator's tone of the voice, as well as exciting words or expressions used, such as "beautiful shot."

[1] Chapter 3 has been published in *IEEE Transactions on Multimedia* 2018 [24], Copyright IEEE.

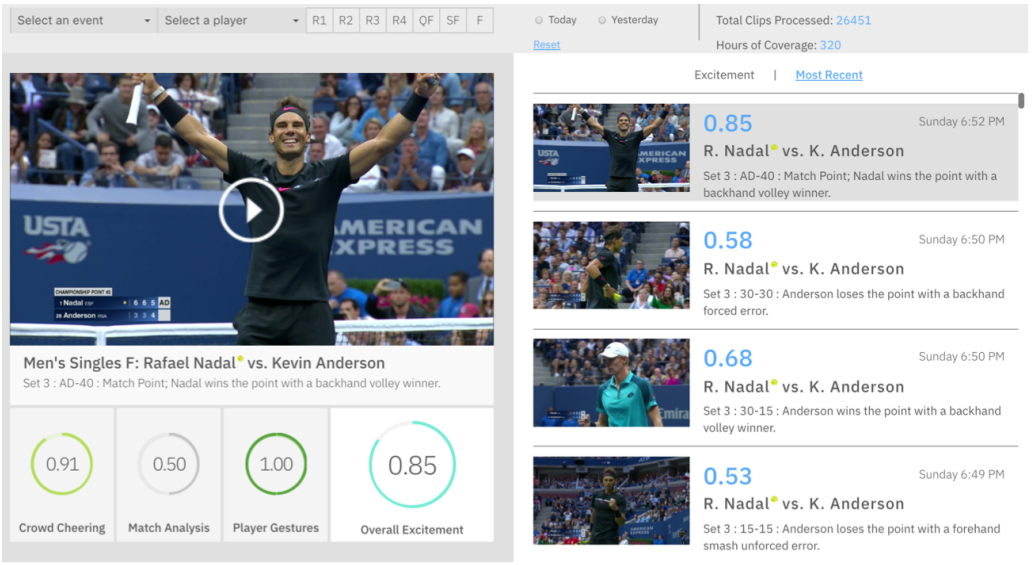


Figure 3.1: The H5 system dashboard for auto-curation of sports highlights. Highlights are identified in near real-time (shown in the right panel) with an associated excitement level score. The user can click on the icons in the right panel to play the associated video in the center, along with the scores for different excitement measures.

For golf, these indicators are used along with the detection of TV graphics (*e.g.*, lower third banners) and shot-boundary detection to accurately identify the start and end frames of key shot highlights with an overall excitement score. For tennis, the start and end points for an exemplar highlight shot can be accurately determined based on input from court-side statisticians and analysts who actively annotate tennis matches in real time. Video segments are then added to an interactive dashboard for quick review and retrieval by a video editor or broadcast producer, speeding up the process by which those highlights can then be shared with fans eager to see the latest action. Fig. 3.1 shows the interface of our system, called High-Five (**High**lights **F**rom **I**ntelligent **V**ideo **E**ngine), H5 in short. The first prototype of IBM H5 [25, 26] was deployed at the 2017 Masters golf tournament, extracting highlights live from multiple video streams over the course of four days. Based on its success, H5 was further adapted to tennis content and employed during the 2017 Wimbledon and US Open tennis tournaments. This adapted H5 prototype introduced the use of *player expression:* expression on the face of the tennis player (*i.e.*, aggressive, tense, smiling, neutral) to improve the player's reaction marker. Since tennis commentary tends to be less colorful and

excited in tone, the tennis H5 prototype did not employ commentator based markers. The system was successfully employed as the official highlights provider for the Wimbledon and US Open tennis tournaments in 2017.

Besides incorporating multimodal (audio, visual, text) and multi-source (crowd audio, commentator speech, player body, player face, overlaid text, speech text) information for highlights detection, we also exploit how one modality can guide the learning of another modality, with the goal of reducing the cost of manual training data annotation. In particular, we show that we can use TV graphics and OCR as a proxy to build rich feature representations for golf player recognition from *unlabeled* videos, without requiring costly training data annotation. Our audio-based classifiers also rely on feature representations learned from unlabeled video [27], and are used to constrain the training data collection of other modalities (*e.g.*, we use the crowd cheer detector to select training data for player reaction recognition).

Personalized highlight extraction and retrieval is another unique feature of our system. In golf, by leveraging TV graphics and OCR, our method automatically gathers information about the golf player's name and the hole number. This meta-data is matched with relevant highlight segments, enabling searches like "show me all highlights of player X at hole Y during the tournament" (where X and Y are arguments of the query) and personalized highlights generation based on a viewer's favorite players. For tennis, information about players is extracted by meta-data provided by analysts and court-side statisticians, thus allowing the same type of personalization when combined with the analyzed video.

The key **contributions** of our work are listed below:

- We present a first-of-kind system for automatically extracting sport highlights by fusing multimodal excitement measures from the player, spectators, and commentator. By either automatically extracting metadata via TV graphics and OCR or obtaining it from court-side statisticians, we allow personalized highlight retrieval or alerts based on player name, hole or field number, location, and time.

- We introduce novel techniques for learning our multimodal classifiers without requiring costly manual training data annotation. In particular, we build rich feature representations for player recognition without manually annotated training examples.

- We provide an extensive evaluation of our work, showing the importance

of each component in our multimodal approach through ablation studies. We compare our results with professionally curated golf highlights. We also provide an extensive user study comparing highlights automatically produced of our H5 system to human preferences by employing Amazon Mechanical Turk, for both golf and tennis.

Our system was successfully demonstrated and deployed at major international golf and tennis tournaments in 2017, extracting highlights from multiple live channels during the course of the tournaments [1, 2, 3].

3.2 Related Work

Video Summarization. There is a long history of research [28, 29, 30], which aims at producing short videos or keyframes that summarize the main content of long full-length videos, by looking at eliminating redundancy either at signal level (feature dimensionality reduction [31]) or in semantic content [30]. Our work also aims at summarizing video content, but instead of optimizing for representativeness and diversity, as traditional video summarization methods do, our goal is to find highlights or exciting moments in the videos. A few recent methods address the problem of highlight detection in consumer videos [32, 33, 34]. Instead our focus is on sports videos, which offer more structure and objective metrics than unconstrained consumer videos.

Automatic Trailer Generation. Another sub-area of video summarization involving multimodal video analysis that goes beyond content recognition, and focuses instead on affective responses evoked by the video, is movie trailer generation [35, 36, 37]. Evangelopoulos et al. [36] model and combine audio, visual and textual saliency to select the most relevant scenes in a movie. In this space, works focus on detecting content with the highest emotional impact based on movie genre. For instance, in horror movies scenes evoking feelings of suspense or fear are important [38]. In sports, on the other hand, only positive emotions connected to excitement are relevant. Furthermore, differently from this line of research, the focus of our work is on identifying and measuring subjects reactions (players, crowd, and commentator) directly in the video stream, rather than inferring reactions which are supposed to be evoked by inspected content which is deemed as "impressive" [35].

Sports Highlights Generation. Several methods have been proposed to automatically extract highlights from sports videos based on audio and visual cues. Example approaches include the analysis of replays [39, 40, 41], crowd cheering [42, 43], motion features [44], and closed captioning [45]. More recently, Bettadapura et al. [46] used contextual cues from the environment to understand the excitement levels within a basketball game. Tang and Boring [47] proposed to automatically produce highlights by analyzing social media services such as twitter. Decroos et al. [48] developed a method for forecasting sports highlights to achieve more effective coverage of multiple games happening at the same time. Our proposed approach offers a unique combination of excitement measures extracted from live video streams to produce highlights, including information from the *spectators*, the *commentator*, and the *player* reaction. As such, our system incorporates and combines most of the information employed by previous works (audio, visual, text). It could also be easily extended to integrate other sources of attention or excitement, such as social media feeds or production cues (replays, closed captions, etc.). In addition, we enable personalized highlight generation or retrieval based on a viewer‘s favorite players.

Self-Supervised Learning. Recently there has been significant interest in learning deep neural network classifiers without requiring a large amount of manually annotated training examples. *Self-supervised* learning approaches rely on auxiliary tasks for feature learning, leveraging sources of supervision that are usually available “for free” and in large quantities to regularize deep models. Examples of auxiliary tasks include the prediction of ego-motion [49, 50], location and weather [51], spatial context [52, 53], image colorization [54], and temporal coherency [55]. Aytar et al. [27] explored the natural synchronization between vision and sound to learn an acoustic representation from *unlabeled* video. We leverage this work to build audio models for crowd cheering and commentator excitement using few training examples, and use those classifiers to constrain the training data collection for player reaction recognition. More interestingly, we exploit the detection of TV graphics as a free supervisory signal to learn feature representations for player recognition from unlabeled video.

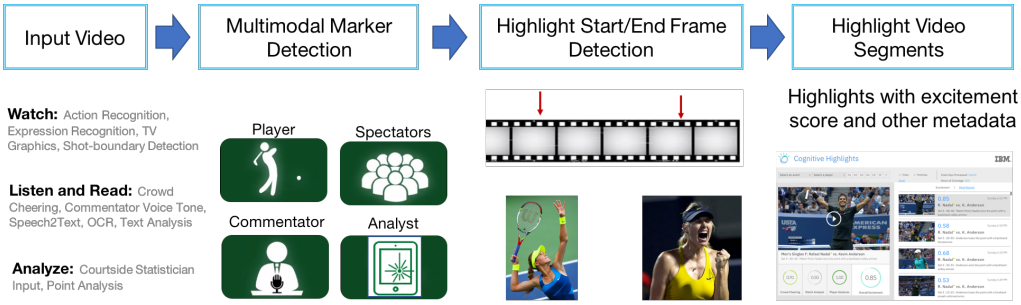


Figure 3.2: Our approach consists of applying multimodal (video, audio, text) marker detectors to measure the excitement levels of the player, spectators, and commentator in video segment proposals. The start/end frames of key shot highlights are accurately identified based on these markers, along with the detection of TV graphics (when available as in golf) and visual shot boundaries, or information from court-side statisticians. The output highlight segments are associated with an overall excitement score, as well as additional metadata about the video segment such as the player name, hole number and shot information in golf, or match point information in tennis.

3.3 Technical Approach

Our framework is illustrated in Fig. 3.2. Given an input video feed, we extract in parallel multimodal markers of potential interest: player action of celebration and facial expression (detected by visual classifiers), crowd cheer (with an audio classifier), commentator excitement (detected by a combination of an audio classifier and a salient keywords extractor applied after a speech-to-text component), and game analyst input information when available (text based metadata). The start and end of a potential highlight clip are determined via analyst input when it is available. In the absence of such input, the start of a highlight is determined by identifying graphic content overlaid to the video feed signifying the start of a shot. Similarly, the end of a highlight segment is identified with visual shot boundary detection, applied in a window of few seconds after the occurrence of the last excitement marker. Additionally, by applying an OCR engine to the graphic, we can recognize the name of the player involved as well as additional metadata such as the hole number, nature of the shot, etc. Finally we compute a combined excitement score for the segment proposal based on a combination of the individual markers. In the following we describe each component in detail.

3.3.1 Audio-based Markers Detection

Crowd Cheer Detection

Crowd cheering is perhaps the most veritable form of approval of a player's shot within the context of any sport. Another important audio marker is excitement in the commentators' tone while describing a shot. Together, those two audio markers play a key role in determining the position and excitement level of a potential highlight clip. We leverage SoundNet [27] to construct audio-based classifiers for crowd-cheering and commentator tone excitement. SoundNet uses a deep 1D convolutional neural network architecture to learn representations of many environmental sounds of objects and scenes, however not including crowd cheering or clapping, nor excitement tone in a person's voice. Therefore a domain adaption step is needed for our purposes. Instead of fine-tuning two SoundNet models, one for each specific task, we chose to employ the same SoundNet deep features as basis to train a linear SVM model for each of the two markers. We opted for this solution to limit the amount of annotation effort, while still building efficient and effective models. At this scale, the literature is not conclusive on whether fine-tuning a deep network is better than learning another model (such as SVM) on top of deep features [56]. Following [27], we extract features from the *conv-5* layer in SoundNet to represent six-second audio segments. The dimensionality of the feature is 17,152. We then learn linear SVM models atop the deep features.

We adopt an iterative refinement bootstrapping methodology to construct our audio based classifiers. We learn an initial classifier with relatively few audio snippets (28 positives and 57 negatives in the first round of training) and then perform a few rounds of bootstrapping on a distinct set. This procedure is repeated to improve the accuracy at each iteration. Cheer samples from 2016 Masters replay videos as well as examples of cheer obtained from YouTube were used as positive training data. For negative examples, we used audio tracks containing regular speech, music, and other kinds of non-cheer sounds found in Masters replays. In total our final training set consisted of 156 positive and 193 negative samples (6 seconds each). The leave-one-out cross validation accuracy on the training set was 99.4%.

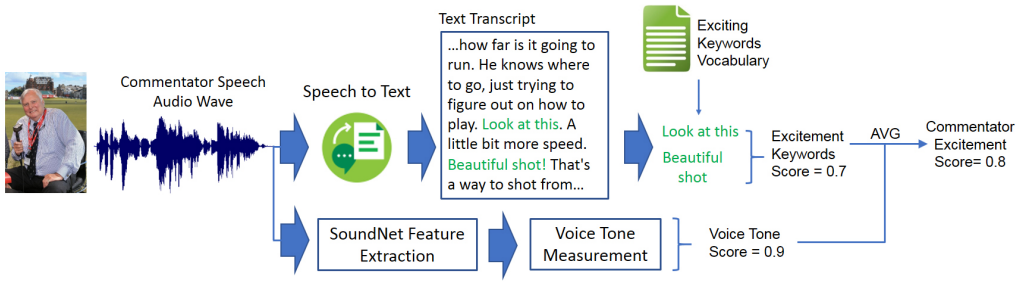


Figure 3.3: Commentator excitement score computation based on (i) audio tone analysis and (ii) speech to text analysis.

Commentator Excitement Detection

We propose a novel commentator excitement measure based on a combination of voice tone and speech-to-text-analysis.

Tone-based: We employ the same deep Soundnet audio features to model excitement in commentators' tone. As above, we employ a linear SVM classifier for modeling. For negative examples, we use audio tracks containing regular speech, music, regular cheer (without commentator excitement) and other sound clips without an excited commentator. In total, the training set for audio based commentator excitement consisted of 131 positive and 217 negative samples. The leave-one-out cross validation accuracy on the training set was 81.3%.

Text-based: Besides the how (*i.e.*, the tone), the excitement level of a commentator can also be gauged from the what, that is, the expressions used. We created a dictionary of 60 expressions (words and phrases) indicative of excitement (*e.g.*, "great shot," "fantastic") and assign to each of them excitement scores ranging from 0 and 1. We use a speech to text service[2] to obtain a transcript of commentators' speech and create an excitement score as an aggregate of scores of individual expressions in it. Finally, we average the tone-based and text-based scores to obtain the overall level of excitement of the commentator, as exemplified in Fig. 3.3.

[2] `https://www.ibm.com/watson/developercloud/speech-to-text.html`

3.3.2 Visual Marker Detection

Player Reaction

Understanding the reaction of a player is another important cue to determine an interesting moment of a game. To the best of our knowledge, measuring excitement with a player celebratory action recognition model for highlight extraction has not been explored in previous work. We adopt two strategies to reduce the cost of training data collection and annotation for action recognition. First, we use our audio-based classifiers at a low threshold to select a subset of video segments for annotation, as in most cases the player celebration is accompanied by crowd cheer and/or commentator excitement. Second, inspired by [57], we use still images which are much easier to annotate and allow training with less computational resources compared to video-based classifiers. Fig. 3.4 shows examples of images used to train our model. At test time, the classifier is applied at every frame and the scores aggregated for the highlight segment. Classifiers to detect player's celebration are based upon the VGG-16 and the ResNet-50 architectures pretrained on ImageNet. Since ImageNet does not contain categories describing a person celebrating, a fine-tuning procedure for our specific domain is needed. We collect positive training examples from 2016 Masters, Wimbledon, and US Open videos, and also from the web. Negative examples are randomly sampled from the aforementioned videos. Similarly to the audio models, multiple rounds of bootstrapping were used to train the model. Details of the training procedure are described in the Sec. 3.5.1.

Facial Expression

Facial expression carries valuable information that can augment or correct predictions from the player reaction models. For example, a tennis player might be raising his arm to collect a ball instead of celebrating a point, thus confusing the player reaction model. In this case, detecting a neutral facial expression can help rejecting a false positive instance. Training data to build a facial expression classifier was collected by extracting faces from the action celebration training images, using a SSD detector [58]. Examples with occlusion, rear-angle, or partial visibility were ignored. The extracted faces were then categorized into four types of expression: aggressive, tense, smiling, and neutral, as shown in Fig. 3.5.

Figure 3.4: Examples used to train the player action recognition model.

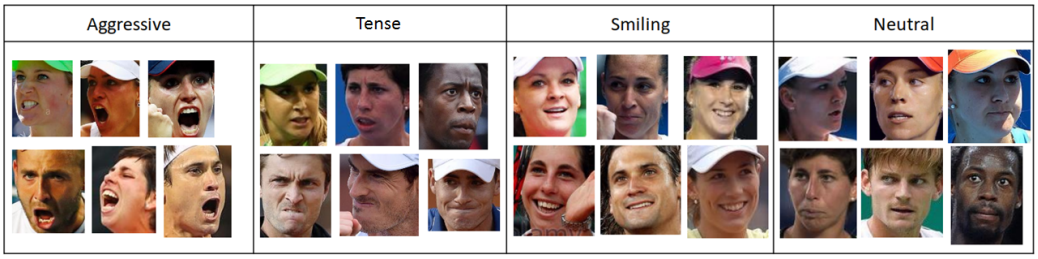


Figure 3.5: Examples of faces used to train the facial expression model.

The first three are associated with celebration, whereas the last one is considered as non-celebratory. The facial expression classifier was trained by fine-tuning a VGG-face [59] model on a manually labeled dataset of tennis players faces.

TV Graphics, OCR, and Shot-boundaries

In professional golf tournament broadcasts, a golf swing is generally preceded by a TV graphics with the name of the player just about to hit the golf ball and other information about the shot. The detection of such markers is straightforward, as they appear in specific locations of the image, and have distinct colors. We check for such colors in the vicinity of pre-defined image locations (which are fixed across all broadcasting channels) to determine the TV graphics bounding box. One could use a more general approach by training a TV graphics detector (for example via Faster-RCNN [60] or SSD [58]), however this was beyond the scope of this work. We apply OCR (using the Tesseract engine [61]) within the detected

region to extract metadata such as the name of the player and the hole number. This information is associated with the detected highlights, allowing personalized queries and highlight generation based on a viewer's favorite players. We then use standard shot-boundary detection based on color histograms as a visual marker to determine the end of a highlight clip.

3.3.3 Game Analytics

In tennis not every point, for how exciting it may be, has equal relevance within a game. For example *match points* and *set points* are more valuable than others, and business rules require them to be included in official highlights packages. During the tournaments, we received live information about the points from statisticians positioned on the side of the court, and compiled it into a single analytics score in the following manner, which was devised following expert advice concerning the significance and difficulty of each item:

- -0.1 for a point won due to unforced error or rally count smaller than 3
- +0.1 for a point won due to positive play, volley winner, smash winner, match point, break point, or rally count greater than 5
- +0.20 for a point won due to forced error, player movement detected, or rally count greater than 10
- +0.25 for a game winning point

where positive play means a point won thanks to a player's active effort, not an opponent's mistake. Player movement signifies that one player moved 25 meters more than the opponent. The sum of values for any given point was then normalized in the range 0 to 1.

3.3.4 Excitement Scores Fusion

For for any given potential highlight clip x, we perform late fusion of the excitement scores $E_n(x)$ produced by each of the N marker classifiers. Specifically, we aggregate (via the max operation) positive scores for each of the markers within the inspected time-window (usually of 15-20 seconds). Each individual score is

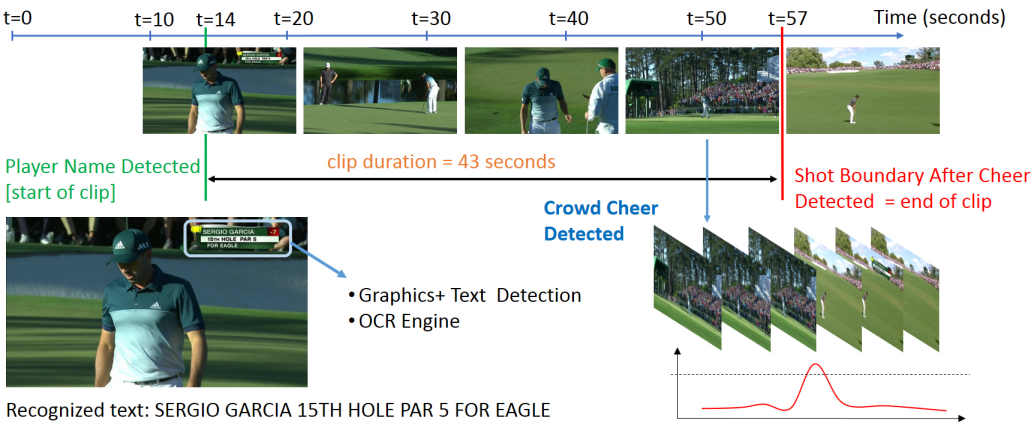


Figure 3.6: Highlight clip start and end frames selection pipeline for the golf video streams.

then registered in the range between 0 and 1 via sigmoid normalization, and the final fusion is computed as a weighted linear sum:

$$F(x) = \sum_{n=1}^{N} w_n E_n(x), \tag{3.1}$$

where n refers to $cheer$, $commentator$, player $action$, and game $analytics$ (when available). The weights w_n for each component are learned via cross-validation on data from the previous year's tournaments. Crowd cheer, commentator excitement (combining audio and text), player reaction and game analytics components weights were set to 0.61, 0.26, 0.13, and 0 respectively for Masters. For Wimbledon and US Open they were learned as 0.6, 0, 0.1, and 0.3 respectively. In Sec. 3.5.2 we compare the benefit of learning the weights versus a Naive-Fusion approach employing equal weighting across components.

3.3.5 Highlight Detection

A highlight is identified as a play or shot that receives a high overall score from the fusion score combining the multimodal markers ones. Besides measuring marker response, it is also important to determine the start and end positions of a highlight. This step is handled differently for the two use cases of golf and tennis, since the inputs to the system were different. We will go through them individually.

Golf: in this case, the input to the system are the live video streams of the 2017 Masters. Fig. 3.6 illustrates then how we incorporate multimodal markers to identify segments as potential highlights and assign excitement scores to them. The system starts by generating *segment proposals* based on the crowd cheering marker. Specifically, crowd cheering detection is performed on a continuous segment of the stream and positive scores are tapped to point to potentially important cheers in audio. Adjacent 6 second segments with positive scores are merged to mark the end of a bout of contiguous crowd cheer. Each distinct cheer marker is then evaluated as a potential candidate for a highlight using presence of a TV graphics marker containing a player name and hole number within a preset duration threshold (set at 80 seconds). The beginning of the highlight is set as five seconds before the appearance of TV graphics marker. In order to determine the end of the clip we perform shot boundary detection in a five-second video segment starting from the end of the cheer marker. If a shot boundary is detected, the end of the segment is set at the shot change point. Segments thus obtained constitute valid highlight segment proposals for the system.

Tennis: As opposed to the golf Masters, Wimbledon and US Open tennis matches are actively annotated by analysts in a live fashion. As a consequence, the start and end times of each play can be accurately determined based on such provided information. Therefore the multimodal marker classification system receives video clips filtered using analyst information as potential highlight candidates and ranks them.

TV graphics detection, shot boundary detection and OCR could be applied to tennis in the same way as they were applied to golf. As our system is motivated by application to real world production needs, we did not investigate the clip detection and cut approach to the tennis videos, since the clips and the player information were already provided to us during the tennis tournaments. However, we believe it would apply seamlessly, as the TV Graphics are easily identifiable and OCR could be employed to identify player names and keep track of the points. The only needed addition would be that of a player serving detection marker, since the start of a tennis point clip corresponds to one player serving. That would require training a specific classifier, which could be done similarly to the player celebratory reaction one.

For both golf and tennis use cases, highlight clips are displayed in the system dashboard as shown in Fig. 3.1 labeled with individual marker scores (normalized between 0 and 1) as well as a combined excitement score that is computed as a

linear combination of the multimodal marker scores.

3.4 Self-Supervised Player Recognition

Automatic player detection and recognition can be a very powerful tool for generating personalized highlights when graphics are not available, as well as to perform analysis outside of the event broadcast itself. It could for example enable to estimate the presence of a player in social media posts by recognizing his face. The task is however quite challenging. First, there is a large variations in pose, illumination, resolution, occlusion (hats, sunglasses) and facial expressions, even for the same player, as visible in Fig. 3.11. Second, inter-player differences are limited, as many players wear extremely similar outfits, in particular hats in golf, which occlude or obscure part of their face. Finally, a robust face recognition model requires large quantities of labeled data in order to achieve high levels of accuracy, which is often difficult to obtain and labor intensive to annotate. We propose to alleviate such limitations by exploiting the information provided by other modalities of the video content, specifically the overlaid graphics containing the players name. This allows us to generate a large set of training examples for each player, which can be used to train a face recognition classifier, or learn powerful face descriptors. In the following we describe the approach employed specifically for the golf tournament data, but it could be easily adapted to other sports.

We start by detecting faces within a temporal window after a graphic with a player name is found, using a Faster-RCNN detector [60]. The assumption is that in the segment after the name of a player is displayed, his face will be visible multiple times in the video feed. Not all detected faces in that time window are going to represent the player of interest. We therefore perform outliers removal, using geometrical and clustering constraints. We assume the distribution of all detected faces to be bi-modal, with the largest cluster containing faces of the player of interest. Faces that are too small are discarded, and faces in a central position of the frame are given preference. Each face region is expanded by 40% and rescaled to 224x224 pixels. Furthermore, only a maximum of one face per frame can belong to a given player. Given all the face candidates for a given player, we perform two-class k-means clustering on top of fc7 features extracted from a VGG Face network [59], and keep only the faces belonging to the largest cluster

while respecting the geometric constraints to be the representative examples of the player's face. This process, working without supervision, allows us to collect a large quantity of training images for each player. We can then train a player face recognition model, which in our case consists of a VGG Face Network fine-tuned by adding a softmax layer with one dimension per player. Fig. 3.11(b) shows an example subset of training faces automatically collected for Sergio Garcia from the 2016 Masters broadcast. The system was able to collect hundreds of images with a large variety of pose and expressions for the same player. Bordered in red are highlighted two noisy examples. While the purity of the training clusters is not perfect, as we will show in the experiments of Sec. 3.5.3 it still allowed to learn a robust classifier with no explicit supervision. This confirms recent results in deep learning modeling, which has been proven to being robust to noise if a large quantity of training data is provided, as demonstrated in recent results for example by Veit et al. on OpenImages [62] or recently achieved top ImageNet performance using noisy data from image tags by Mahajan et al. [63].

3.5 Experiments

We evaluated our system in three real world championships, namely the 2017 Masters, 2017 Wimbledon, and 2017 US Open tournaments. For the 2017 Masters, we analyzed in near real-time the content of four channels broadcasting simultaneously over the course of four consecutive days, for a total of 124 hours of content[3]. Our system produced 741 highlights over all channels and days. The system ran on a Redhat Linux box with two K40 GPUs. We extracted frames directly from the video stream at a rate of 1fps and audio in six seconds segments encoded as 16bit PCM at rate 22,050 kHz. The cheer detector and commentator excitement ran in real time (one second to process one second of content), action detection took 0.05secs per frame, graphics detection with OCR took 0.02secs per frame. Speech-to-text was the only component slower than real time, processing six seconds of content in eight seconds, since we had to upload every audio chunk to an API service. The 2017 Wimbledon and US Open system ran on two Ubuntu nodes with four K80 GPUs each, providing a total of 16 stream services to process candidate highlight clips during the tournaments. Videos were chunked in

[3]Video replays are publicly available at `http://www.masters.com/en_US/watch/index.html`

Table 3.1: Details of the test sets used to evaluate the player celebration action recognition models.

Event	# clips	# frames	# positives	# negatives
2017 Masters	-	1,064	59	1,005
2017 Wimbledon	540	4,777	78	4,699
2017 US Open	510	8,963	52	8,911

10 seconds clips and analyzed in less than 2.5 seconds through our service APIs. Frames and audio extracted from each video were distributed to components for crowd cheering detection, action recognition, expression recognition, and overall aggregation.

In the following we report experiments conducted after the events to quantitatively evaluate the performance of H5, both in terms of overall quality of the produced highlights as well as efficacy of its individual components. All training was performed on content from the 2016 Masters, Wimbledon and US Open tournament videos and from images downloaded from the web, while testing was done on video data from the 2017 tournaments.

3.5.1 Individual Markers

Player Celebration Marker

The player celebration classifier for the 2017 Masters was trained with 574 positive examples and 563 negative examples. The positive examples were sampled from 2016 Masters replay videos and also from the web. The negative examples were randomly sampled from the 2016 Masters videos. We used the VGG-16 model [64], pre-trained on ImageNet as our base model. The Caffe [65] deep learning library was used to fine-tune the model to our data with stochastic gradient descent, learning rate 0.001, momentum 0.9, and weight decay 0.0005. We performed three rounds of hard negative mining on 2016 Masters videos, obtaining 2,906 positive examples and 6,744 negative ones.

Facial Expression Marker

In a similar fashion, player celebration classifiers for the 2017 Wimbledon and US Open were trained using samples from 2016 tournament video frames as well

Table 3.2: Details of the test sets used to evaluate performance of the facial expression recognition models.

Event	# samples	#aggressive	# tense	# smiling	# neutral
2017 Masters	1,285	45	222	346	672
2017 Wimbledon	472	18	56	38	360
2017 US Open	1,129	25	171	44	889

Table 3.3: Details of the test set used to evaluate performance of the crowd cheering recognition models.

Event	# samples	# positives	# negatives
2017 Masters	405	69	336
2017 Wimbledon	1,073	915	158
2017 US Open	1,564	627	937

as examples from the web including multiple rounds of bootstrapping. The final training set for Wimbledon consisted of 13,263 positive and 33,372 negatives samples, augmented by random cropping and horizontal flipping. Since the US Open setting is quite different from Wimbledon's, we trained new celebration classifiers using a training set consisting of 11,330 positive and 12,516 negative samples. The examples from the web were reused from the Wimbledon training, while new video frames were annotated specifically for the US Open. We explored two deep architectures, VGG-16 and ResNet-50 pre-trained on Masters data, and found the ResNet model to work best.

We evaluated the player celebration models on a set of clips randomly selected from each of the tournaments and manually labeled. Table 3.1 reports the details of each test set. The imbalance of positive and negative examples reflects the actual distribution of data, since occurrences of a player celebrating are relatively rare within a match. Classification accuracies on the 2017 Masters, Wimbledon, and US Open data were 98.4%, 98.12% and 99.33% respectively. Compared to VGG-16, the ResNet-50 models performed better on the 2017 Wimbledon (AUC = 0.91 versus 0.87 for VGG) while they were equivalent for the US Open (both AUC = 0.94). Because of the superior performance of ResNet, we used those models in the fusion phase. In Fig. 3.7a we show the ROC curves of the best models for the all three inspected tournaments. We can observe that recognizing players celebrating was easier in golf than in tennis. In fact, despite a significantly smaller training dataset, the model performs better. This is mostly due to false

Table 3.4: Details of the test set used to evaluate performance of the commentator tone recognition models.

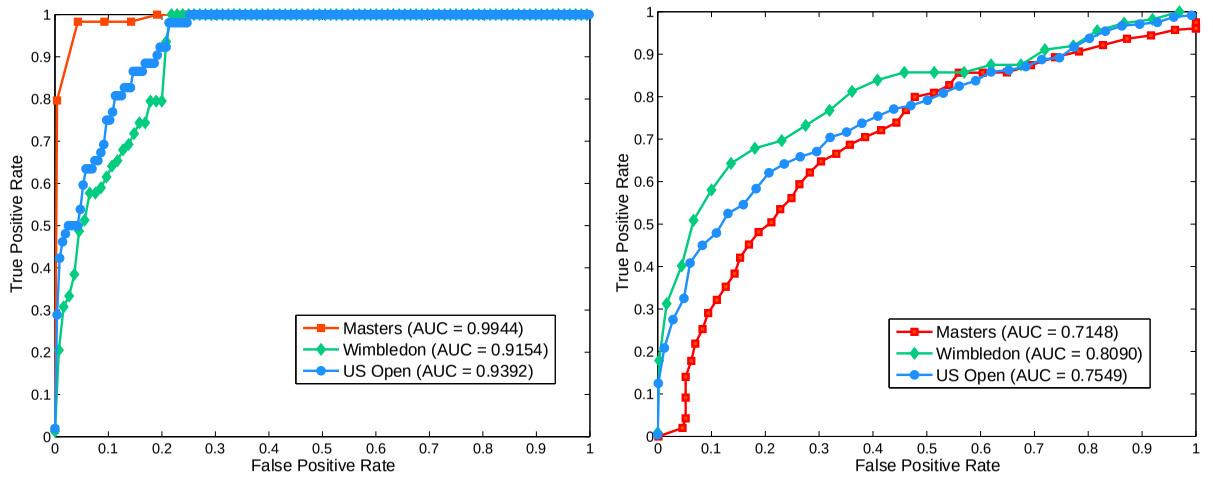
Event	# samples	# positives	# negatives
2017 Masters	240	46	194
2017 Wimbledon	437	85	352
2017 US Open	423	111	312

positives occurring in tennis when players serve, catch a ball in their hand, or pass a towel over their head. The high false positives rate from the the player reaction models was one of the main motivations to introduce a facial expression module.

The facial expression marker was tested on faces extracted from the 2017 Wimbledon and US Open test set videos. While it was not employed during the tournament, we also evaluated this marker on the 2017 Masters data after the event. As shown in the Table 3.2, the expressions on the players faces were at first manually labeled into four categories, which we found to be most representative of the appearance of the players from the 2016 tournaments. During the 2017 Wimbledon however, we found that aggressive, tense and smiling all correlated with players' celebrations. We therefore combined those facial expressions with a linear fusion to generate an overall "excited" score, which was compared against the neutral score representing a lack of celebration. Fig. 3.7b illustrates the ROC curves of the facial expression marker using this binary categorization. The Figure shows that the modules performed reliably enough for both tennis tournaments, with the 2017 Wimbledon's performance being better (AUC of 0.81 for the 2017 Wimbledon versus 0.75 for the 2017 US Open). Recognition accuracies are 82.42% and 82.23%, respectively. The results for golf were worse, with AUC of 0.71 and accuracy of 78.57%. While the performance is not by itself perfect, it resulted in being acceptable since facial expressions were used to refine the results given by the celebration action model.

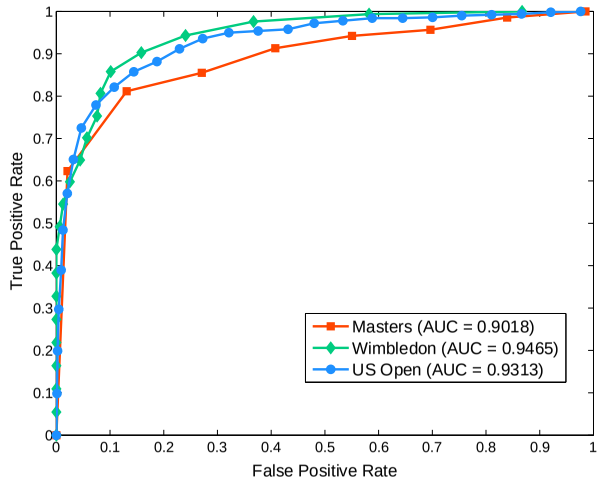
Crowd Cheering Marker

Cheer samples from 2016 Masters and Wimbledon replay videos as well as YouTube were used to train the audio cheer classifier using a linear SVM on top of deep features. For negative examples, we used audio tracks containing regular speech, music, and other non-cheer sounds found in Masters and Wimbledon replays. In total the training set consisted of 453 positive and 454 negative samples (6 sec-

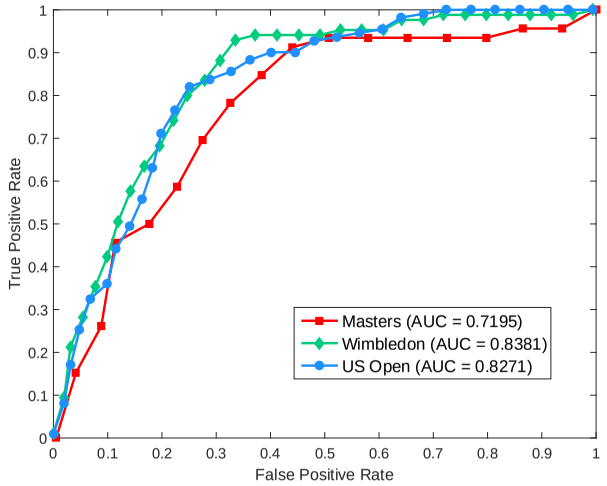


(a) Player Reaction

(b) Facial Expression



(c) Crowd Cheer



(d) Commentator Tone

Figure 3.7: ROC curves for different excitement markers on the test clips from the 2017 Masters, Wimbledon and US Open tournaments.

Table 3.5: OCR performance in terms of words and characters recognition.

Element	Total Number	Precision	Recall
Words	7,663	0.9916	0.9893
Characters	29,016	0.9846	0.9840

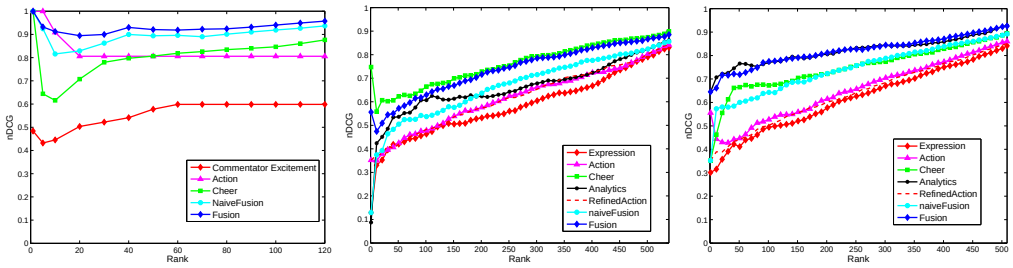
onds each). We manually annotated random sets of six-seconds audio clips from the 2017 Masters, Wimbledon and US Open tournaments videos to evaluate the performance of the model. Table 3.3 reports detailed numbers of test sets and Fig. 3.7c performance of the audio cheer model. The resulting ROC curves are approximately similar, with AUC of 0.9, 0.94 and 0.93 for the 2017 Masters, Wimbledon, and US Open, respectively.

Commentator Tone Marker

In a fashion similar to crowd cheering, we created a training set for the commentator tone excitement marker using 2016 Masters videos and several rounds of bootstrapping. In total, the training set consisted of 131 positive and 217 negative samples. The model was employed only for the golf Masters tournament, as the tennis data we were provided as input did not contain any useful commentary. We evaluated the commentator tone model on randomly sampled audio snippets from the 2017 Masters tournament, and from Youtube videos of past Wimbledon and US Open matches, as summarized in Table 3.4. Each audio clip was sent to AMT for evaluation by 5 different workers, who had to label it as *No speech* (0), *Softly spoken*(1), *Average Excitement*(2), *Loud Excitement*(3). Any clip with an average score of at least 2 was considered as exciting, while the others were considered non-exciting. Fig. 3.7d shows the ROC curves of the model, with an AUC = 0.72, 0.83 and 0.82 for Masters, Wimbledon and US Open, respectively. While the performance of this model is not as good as the cheer classifier, it was reliable enough to be employed in the live system during the Master tournament.

Text OCR Marker

In order to evaluate the text detection and OCR performance, we randomly selected 625 frames from the four channels during the first day of the 2017 Masters tournament. For each of the frames, we manually transcribed the ground truth text



(a) 2017 Masters (b) 2017 Wimbledon (c) 2017 US Open

Figure 3.8: nDCG computed at different ranks for the individual components as well as the Fusion.

and compared it to the outcome of the OCR engine. From the results in Table 3.5 we observe that the system was able to recognize the overlaid text very accurately. Overall, only in 7 frames the name of the player was not properly recognized, while the most common mistake (happened 60 times) was the confusion of the letter *T* with the letter *I* in the ordinal numbers indicating the hole (for example, *15TH HOLE* misspelled as *15YH HOLE*). Precision and Recall are computed as

$$Precision = \frac{N_{cor}}{N_{gt}}, \quad Recall_c = \frac{N_{cor}}{N_r}, \tag{3.2}$$

$$N_{cor} = N_{gt} - ED\left(s_g, s_r\right), \tag{3.3}$$

where N_{cor} is the number of correctly recognized characters, that is, the number N_{gt} of ground truth characters minus the edit distance $ED\left(s_g, s_r\right)$ between the ground truth text s_g and the text output from the system s_r.

3.5.2 Highlights Detection

Evaluating the quality of sports highlights is a challenging task, since a clearly defined ground truth does not exist. Similarly to previous works [46], we approached this problem by comparing the clips automatically generated by our system to two human based references. The first is a human evaluation and ranking of the clips that we produced. The second is the collection of highlights professionally produced by the official Masters curators and published on their Twitter channel.

Table 3.6: Distribution of clips and workers used to evaluate clips rankings for each tournament.

Tournament	# Clips	Workers/Clip	Total # Workers	% Fan Workers
2017 Masters	120	3	3	33%
2017 Wimbledon	540	5	33	58%
2017 US Open	510	5	21	59%

Human Evaluation of Highlights Ranking

In order to determine the quality of the rankings produced by our system, we conducted user studies on Amazon Mechanical Turk. Workers were asked to evaluate the excitement level of several clips randomly sampled from the ones generated and scored by the H5 framework. We asked each participant to assign a score to every clip in a scale from 0 to 5, with 0 meaning a clip without any interesting content and 5 being the most exciting shots. We then averaged the scores of the users for each clip. Table 3.6 summarizes the number of clips and workers employed for each tournament. We also asked each worker if they were fans of the given sport, and on average we found half of them being fans.

Specifically for the 2017 Masters, a score of 1 had the unique meaning of a highlight that is associated with the wrong player, that is, a system mistake. The resulting scores determined that 92.68% of the clips produced by our system were legitimate highlights (scores 2 and above), while 7.32% were mistakes.

We then compared the rankings of the clips according to the scores of each individual component, as well as their fusion, to the ranking obtained through the users votes. The performance of each ranking was computed at different depth k with the normalized discounted cumulative gain (nDCG) metric, which is a standard retrieval measure computed as follows

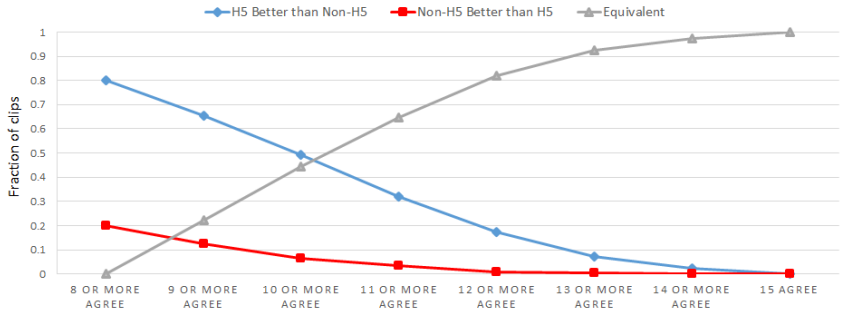
$$nDCG(k) = \frac{1}{Z} \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{log_2(i+1)}, \tag{3.4}$$

where rel_i is the relevance score assigned by the users to clip i and Z is a normalization factor ensuring that the perfect ranking produces a nDCG score of 1. In Fig. 3.8 we present the nDCG at different ranks. Fusion was obtained as a weighted sum of the normalized scores from each component. We tested two different fusion configurations: a Naive-Fusion using equal weights, and the Fusion with weights optimized through cross-validation on a separate training set (which

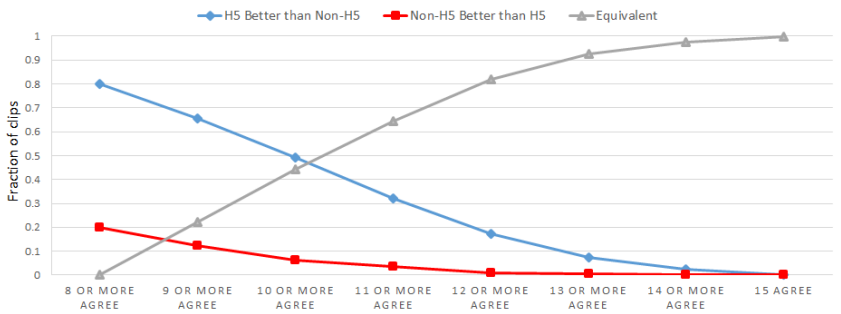
was used by the system during the tournaments), as described in Sec. 3.3.4. In all cases the system's Fusion outperforms the Naive one, confirming the benefit of assigning different weights to different individual components. For both the 2017 Wimbledon and US Open tournaments, the Refined Action obtained by adjusting the player celebration score with the facial expression component (increase if facial expression is "excited," decrease if the expression is neutral) correlated better than the base Action one, confirming the benefit of introducing the facial expression marker. In general, H5 produced rankings which correlated more with human preferences for Golf than for Tennis. For the 2017 Masters (a) we notice that all components but the Commentator Excitement correctly identify the most exciting clip (at rank 1). After that only the Action component assigns the highest scores to the following top 5 clips. When considering 10 top clips or more, the benefit of combining multiple modalities becomes apparent, as the Fusion nDCG curve remains constantly higher than each individual marker. Differently from the 2017 Masters, for the 2017 Wimbledon and US Open the Fusion does not outperform individual components. However it remains fundamental for the system to generalize, as it is interesting to notice how for different tournaments, different components correlated most with human rankings. For Wimbledon (b) Cheer was the best indicator for human excitement, whereas for US Open (c) the game analytics mattered the most. In both cases the Fusion closely follows the performance of the best individual marker.

Tennis A/B Testing

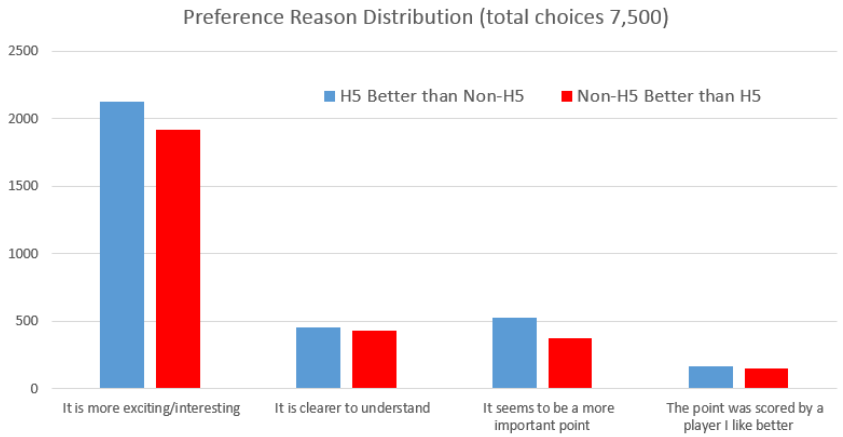
Besides the ranking of clips for Tennis, we also wanted to determine whether the *selection* made by the system about which clips should go into the compiled highlights and which should be instead discarded followed human preferences. Thus we evaluated the clip selection process through another Amazon Mechanical Turk experiment. In this case for each tournament we randomly selected 500 pairs of clips. In each pair both clips belonged to the same game: one clip which had been selected to be part of the highlights, and one clip which had been discarded. We then presented each pair to the workers and asked them to pick which clip in the pair was more exciting and/or interesting. We also asked the workers to motivate their choice among multiple options and to provide some demographic information. Each pair was voted on by 15 workers, and a total of 234 unique users participated in the study. From the results reported in Fig. 3.9 (a) and (b) we can



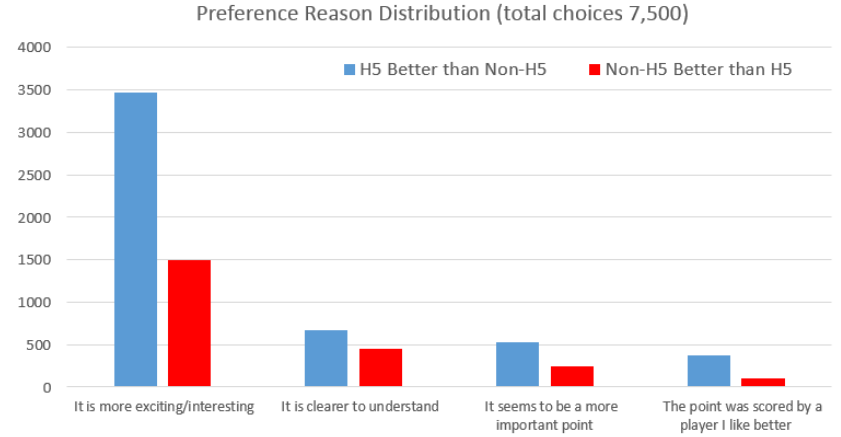
(a) 2017 Wimbledon Preferences



(b) 2017 US Open Preferences



(c) 2017 Wimbledon Reasons



(d) 2017 US Open Reasons

Figure 3.9: Human preferences in AB tests for the 2017 Wimbledon and US Open.

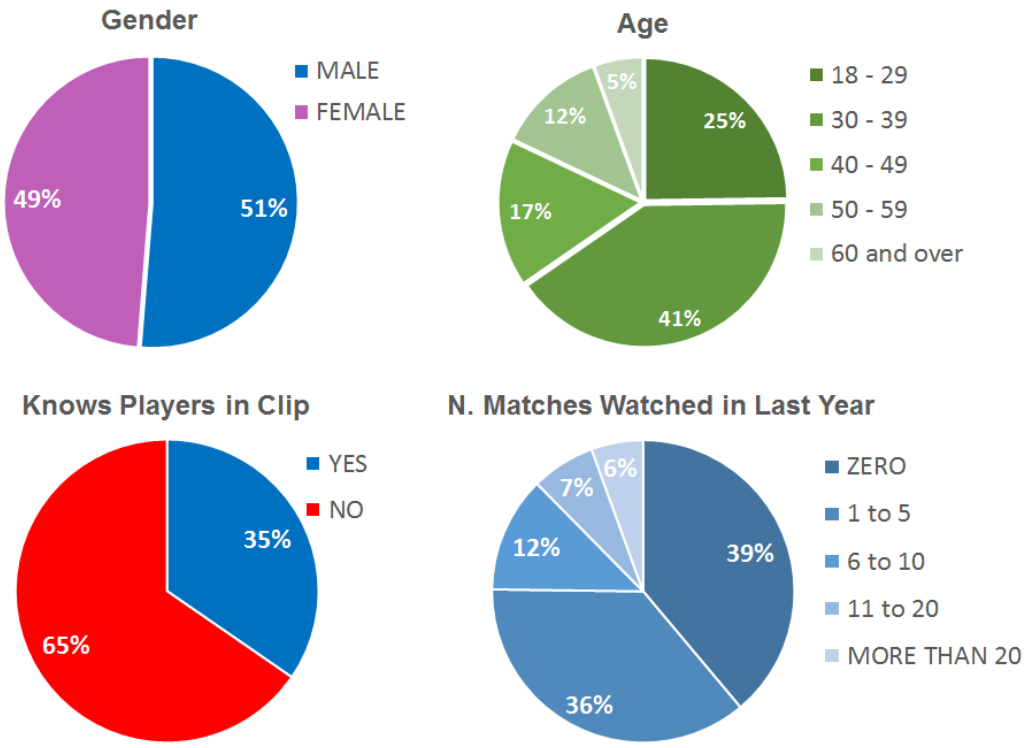


Figure 3.10: A/B Tests users demographics information

observe how for both tournaments the majority of voters picked the clips which were selected by the system to be part of the highlights of a game (blue curves) overwhelmingly over the non highlight worthy ones (red curves). Naturally the fraction of clips on which a larger number of users agrees decreases as we move from 8 (the majority of voters) to to 15 (all the voters), a trend clearly visible in the growth of the grey curves representing an indecision. The distribution of reasons for the choices is highly skewed toward how exiting a clip was, as users paid less attentions to clip clarity (tends to be very similar, as clips belong to the same game), players scoring or significance of a point withing a game. The detailed breakdown is presented in Fig. 3.9 (c) and (d). From the demographic information collected in Fig. 3.10 we can observe a quite even distribution in gender, with a prevalence of young people (18 to 29 years old) who mostly did not know the players in the clips they voted on. This is consistent with the reason *the point was scored by the player I like better* being the least used in Fig. 3.9 (c) and (d). Finally, it seems that the majority of workers were not tennis fans, having watched less than five games in the past year.

Table 3.7: Highlights detection performance. Comparison between the top k ($k = 120, 500$) retrieved clips from H5 and the official 2017 Master's Twitter highlights.

Depth	120	500
Precision	0.54	0.35
Recall	0.4	0.9
Matching Highlights Preference	0.57	-
Non-Matching Highlights Preference	0.33	-
Equivalent	0.10	-

Comparison with the Official 2017 Masters Highlights

The previous experiments confirmed the quality of the identified highlights as perceived by potential users of the system. We then compared H5 generated clips with highlights professionally created for the 2017 Masters, *Masters Moments*, available at their official Twitter page[4]. There are a total of 116 highlight videos from the final day at the 2017 Masters. Each one covers a player's approach to a certain hole (*e.g.*, Daniel Berger, 13th hole) and usually contains multiple shots taken to complete a particular hole. In contrast each H5 highlight video is about a specific shot at a particular hole for a given player. In order to match the two sets of videos, we considered just the player names and hole numbers and ignored the shot numbers. After eliminating Masters Moments outside of the four channels we covered live during the tournament and for which there is no matching player graphics marker, we obtained 90 Masters Moments.

In Table 3.7, we report Precision and Recall of matching clips over the top 120 highlights produced by the H5 Fusion system. We observe that approximately half of the clips overlap with Masters Moments. This leaves us with three sets of videos: one shared among the two sets (a gold standard of sorts), one unique to Masters Moments and one unique to H5. We observed that by lowering thresholds on our markers detectors, we can incorporate 90% of the Masters Moments by producing more clips. Our system is therefore potentially capable of producing almost all of the professionally produced content. We also wanted to investigate the quality of the clips which were discovered by the H5 system beyond what the official Master's channel produced. Generation of highlights is a subjective task and may not comprehensively cover every player and every shot at the Masters. At the same time, some of the shots included in the official highlights may not

[4] `https://twitter.com/mastersmoments`

necessarily be great ones but strategically important in some ways.

While our previous experiment was aimed at understanding the coverage of our system vis-a-vis the official 2017 Masters highlights, we wondered if a golf aficionado would find the remaining videos still interesting (though not part of official highlights). We therefore aimed an experiment at quantitatively comparing (a) H5 highlight clips that matched Masters Moments and (b) H5 highlight clips that did not match Masters Moments videos. In order to do so we selected the 40 most highly ranked (by H5) videos from lists (a) and (b) respectively and performed a user study using three human participants familiar with golf. Participants were shown pairs of videos with roughly equivalent H5 scores/ranks (one from list (a) and the other from list (b) above) and were asked to label the more interesting video between the two, or report that they were equivalent. Majority voting was used among the users votes to determine the video pick from each pair. From the results reported in Table 3.7 we observe that while the preference of the users lies slightly more for videos in set (a), in almost half of the cases the highlights uniquely and originally produced by the H5 system were deemed equally if not more interesting. This reflects that the system was able to discover content that users find interesting and goes beyond what was officially produced. It is also interesting to notice that our system is agnostic with respect to the actual score action of a given play, that is, a highlight is detected even when the ball does not end up in the hole, but the shot is recognized as valuable by the crowd and/or commentator and players through their reactions to it.

3.5.3 Self-Supervised Player Face Recognition

In order to test our self-supervised player recognition model we randomly selected a set of 10 players who participated to both the 2016 and the 2017 Masters tournaments (shown in Fig. 3.11 (a)). In Table 3.8 we report the statistics of the number of training images that the system was able to automatically obtain in a self-supervised manner. For each player we obtain on average 280 images. Data augmentation in the form of random cropping and scaling was performed to uniform the distribution of examples across players. Since there is no supervision in the training data collection process, some noise in bound to arise. We manually inspected the purity of each training cluster (where one cluster is the set of images representing one player) and found it to be 94.26% on average. Note that

Figure 3.11: Self-supervised player face learning. (a) Examples of the 10 players used in the experiments. (b) Subset of the images automatically selected as training set (2016 Masters) for Sergio Garcia (note the diversity of pose, expression, occlusion, illumination, resolution). (c) Examples of test faces (the 2017 Masters) correctly recognized through self-supervised learning. (d) Examples of False Negatives (in orange) and False Positives (in red).

despite evaluating its presence, we did not correct for the training noise, since our method is fully self-supervised. The face recognition model was fine-tuned from a VGG-face network with learning rate = 0.001, $\gamma = 0.1$, momentum = 0.9 and weight decay = 0.0005. The net converged after approximately 4K iterations with batch size 32. We evaluated the performance of the model on a set of images randomly sampled from Day 4 of the 2017 Masters and manually annotated with the identity of the 10 investigated players. Applying the classifier directly to the images achieved 66.47% accuracy (note that random guess is 10% in this case since we have 10 classes). We further clustered temporally close frames based on fc7 features and assigned to all faces in a cluster the identity which received the highest number of predictions within the cluster. This process raised the performance to 81.12%. Fig. 3.11 (c) shows examples of correctly labeled test images of Sergio Garcia. Note the large variety of pose, illumination, occlusion and facial expressions. In row (d) we also show some examples of false negatives (bordered in orange) and false positives (in red). The net result of our framework is thus a self-supervised data-collection procedure which allows to gather large quantities

Table 3.8: The 2017 Masters Player face classification performance.

Number of Players	10
Number of Training Images	2,806
Training Clusters Purity	94.26%
Number of Test Images	1,181
Random Guess	10.00%
Classifier Alone Accuracy	66.47%
Classifier + Clustering Accuracy	**81.12%**

of training data without need for any annotation, which can be used to learn robust feature representations and face recognition models.

3.5.4 Discussion

Ablation study results. The combination of multimodal excitement measures is crucial to determine the most exciting moments of a game. Though crowd cheer is an important marker, it alone cannot differentiate a hole-in-one or the final shot of a golf tournament from other equally loud events. In addition, we noticed several edge cases where non-exciting video segments had loud cheering from other holes. Our system correctly attenuates the highlight scores in such cases, due to the lack of player celebration and commentator excitement. In tennis, we observed how the player celebration marker can produce false positives associated with raising one's hands for purposes other than celebrating (for example cleaning one's sweat from the forehead). Our system copes with it by analyzing the player's facial expression in conjunction with his or her actions.

Comparison to the state of the art and extensions. Many state of the art approaches for sports and video analytics are actually complementary to ours. We believe that other sources of excitement measures, such as as replays [39, 40], crowd facial expressions or information from social media feeds [47] could be easily integrated within our framework to further improve it. The live feed nature of the video streams we analyzed during the tournaments, which are the input of our system, made it impossible to rely on production cues such as replays for our purposes at the time. Similarly we did not have access to social media feeds during the events. Integrating such complimentary cues could be a good direction for future work. Also, end-to-end approaches to video description or action recognition (to further capture the plasticity of a move, for example) could be em-

ployed within our framework, although currently the lack of large-scale annotated training data hinders the development of such approaches. Finally, most existing works utilize one or a subset of the components we employ within our framework. For example Baijal et al. [42] and Xiong et al. [44] use audio events (such as crowd cheering) only, Zhang et al. [31] employ closed captions analysis, which can be equated to the commentator text analysis we perform on the output of the speech to text module. As such, the extensive ablation study we performed with the evaluation of the contribution of each individual component in our framework and their combination, as reported in Sec. 3.5.2, can be considered as a proxy for comparison with many existing state of the art methods.

Other uses of self-supervised learning. The same approach used for self-supervised player recognition could also be applied for the detection of other items, for example golf setup (player ready to hit the golf ball), tennis player serving or handshake at the end of a game, using TV graphics or other modalities metadata as a proxy to obtain positive examples without manual supervision. This would generalize our approach to detect the start of an event without relying on TV graphics, and also help fix a few failure cases of consecutive shots for which a single TV graphics is present.

Extension to other sports. While we have demonstrated our approach for golf and tennis, we believe our proposed techniques for modeling the excitement levels of the players, commentator, and spectators are general and can be extended to other sports as well since most of our markers are sport agnostic. However, it should be noted that both tennis and golf are relatively quiet sports, where exciting events are rare. A sport like basketball or soccer has the crowds chanting all the time and it would be challenging to directly employ a completely sport-agnostic system like ours out-of-the-box, without any adaptation. In those instances, specialized knowledge of the sport in question can definitely add value to the highlight selection process. An integration of sport-specific action detection markers (*i.e.*, a basketball dunk, or a soccer goal) might be helpful for the system to work. On the other hand, the system might be directly applicable to similar "quiet" sports such as cricket.

3.6 Conclusion

We presented a novel approach for automatically extracting highlights from sports videos based on multimodal sport-independent excitement measures, including audio analysis from the spectators and the commentator, and visual analysis of the players. Based on that, we developed a first-of-a-kind system for auto-curation of golf and tennis highlight packages, which was demonstrated in three major golf and tennis tournaments in 2017. We also exploited the correlation of different modalities to learn models with reduced cost in training data annotation. As next steps, we plan to generalize our approach to other sports such as soccer and produce more complex storytelling video summaries of the games, while including additional indicators of play importance such as social media feeds or game specific events detection.

CHAPTER 4

LEARNING MOTION IN FEATURE SPACE

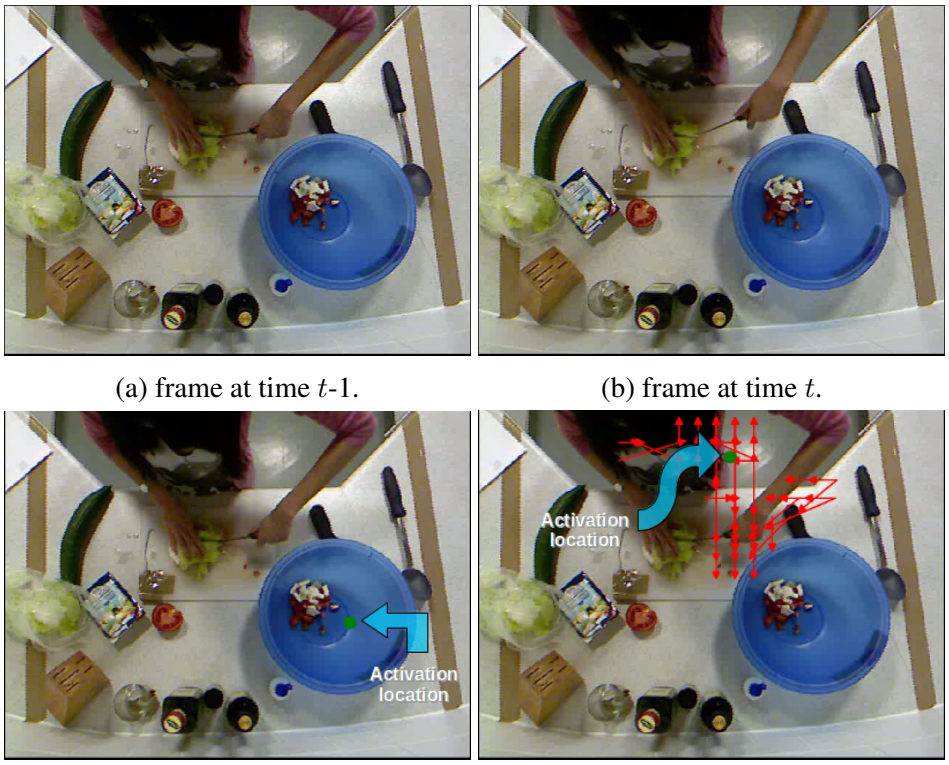
4.1 Introduction

[1] *Action detection*, *a.k.a* action segmentation, addresses the task of classifying every frame of a given video, containing multiple action segments, as one out of a fixed number of defined categories, including a category for unknown actions. This is contrary to the simpler task of *action recognition*, wherein a given video is pre-segmented and guaranteed to be one of the provided action classes [67].

Fine-grained actions are a special class of actions which can only be differentiated by subtle differences in motion patterns. Such actions are characterized by high inter-class similarity [68, 69], *i.e*., it is difficult, even for humans, to distinguish two different actions just from observing individual frames. Unlike generic action detection, which can largely rely on "what" is in a video frame to perform detection, fine-grained action detection requires additional reason about "how" the objects move across several video frames. In this work, we consider the fine-grained action detection setting.

The pipeline of fine-grained action detection generally consists of two steps: (1) spatiotemporal feature extraction and (2) long-temporal modeling. The **first step** models spatial and short-term temporal information by looking at a few consecutive frames. Traditional approaches tackle this problem by decoupling spatial and temporal information in different feature extractors and then combining the two streams with a fusion module. Optical flow is commonly used for such short-term temporal modeling [70, 71, 72, 69, 73]. However, optical flow is usually computationally expensive and may suffer from noise introduced by data compression [5, 4]. Other approaches use Improved Dense Trajectory (IDT) or Motion History Image (MHI) as an alternative to optical flow [74, 4, 75]. Recently, there

[1] Chapter 4 has been published in *The IEEE International Conference on Computer Vision (ICCV)* 2019 [66], Copyright IEEE.



(a) frame at time t-1.

(b) frame at time t.

(c) no motion vectors found on the background region.

(d) motion vectors found on the moving region.

(e) the person at time t-1 (blue) and t (green).

(f) visualization of motion in feature space.

Figure 4.1: Visualization of difference of adaptive receptive fields for action *cutting lettuce* in 50 Salads dataset: (a) and (b) are two consecutive frames; (c) and (d) are motion vectors at background and moving regions (green dots indicate activation locations and red arrows indicate motion vectors); (e) is the manually defined mask of the person at time $t-1$ and t; and (f) is the energy of motion field in feature space, computed by aggregating motion vectors in all deformable convolution layers.

have been efforts to model motion in video using variants of 3D convolutions [76, 77, 78]. In such cases, motion modeling is somewhat limited by receptive fields of standard convolutional filters [79, 80, 81].

The **second step** models long-term dependency of extracted spatiotemporal features over the whole video, *e.g.*, bi-directional LSTM [69], spatial-temporal CNN (ST-CNN) with segmentation models [4], temporal convolutional networks (TCN) [5], and temporal deformable residual networks (TDRN) [82]. Recent works that focused on modeling long-term dependency have usually relied on existing features [5, 4, 82]. In this work, we create efficient short-term spatiotemporal features which are very effective in modeling fine-grained motion.

Instead of modeling temporal information with optical flow, we learn temporal information in the *feature space*. This is accomplished by utilizing our proposed *locally-consistent deformable convolution (LCDC)*, which is an extension of the standard deformable convolution [83]. At a high-level, we model motion by evaluating the local movements in adaptive receptive fields over time (as illustrated in Fig. 4.1). Adaptive receptive fields can focus on important parts [83] in a frame, thus using them helps focus on movements of interesting regions. On the other hand, traditional optical flow tracks all possible motion, some of which may not be necessary. Furthermore, we enforce a local coherency constraint over the adaptive receptive fields to achieve temporal consistency.

To demonstrate the effectiveness of our approach, we evaluate on two standard fine-grained action detection datasets: 50 Salads [6] and Georgia Tech Egocentric Activities (GTEA) [7]. We also show that our features, without any optical flow guidance, are robust and outperform features from original networks. Additionally, we perform quantitative evaluation of the learned motion using ablation studies to demonstrate the power of our model in capturing temporal information.

Our main contributions are: *(1) Modeling motion in feature space* using changes in adaptive receptive fields over time, instead of relying on pixel space as in traditional optical flow based methods. To the best of our knowledge, we are the first to extract temporal information from receptive fields. *(2) Introducing local coherency constraint* to enforce consistency in motion. The constraint reduces redundant model parameters, making motion modeling more robust. *(3) Constructing a backbone single-stream network to jointly learn spatiotemporal features*. This backbone network is flexible and can be used in consonance with other long-temporal models. Furthermore, we prove that the network is capable of representing temporal information with a behavior equivalent to optical flow.

(4) Significant reduction of model complexity is achieved without sacrificing performance by using local coherency constraint. This reduction is proportional to the number of deformable convolution layers. Our single-stream approach is computationally more efficient than traditional two-stream networks, as they require expensive optical flow and multi-stream inference.

4.2 Related work

An extensive body of literature exists for features, temporal modeling, and network architectures within the context of action detection. In this section, we will review the most recent and relevant papers related to our approach.

Spatio-temporal features. Spatio-temporal features are crucial in the field of video analysis. Usually, the features consist of spatial cues (extracted from RGB frames) and temporal cues over a *short* period of time. Optical flow [84] is often used to model temporal information. However, it was found to suffer from noise due to video compression and insufficient to capture small motion [5, 4]. It is also generally computationally expensive. Other solutions to model temporal information include Motion History Image (MHI) [74], leveraging the difference of multiple consecutive frames, and Improved Dense Trajectory (IDT) [75], combining HOG [85], HOF [75], and Motion Boundary Histograms (MBH) descriptors [86].

To combine spatial and (short) temporal components, Lea *et al*. [4] stacked an RGB frame with MHI as input to a VGG-like network to produce features (which they refereed to as SpatialCNN features). Simonyan and Zisserman [72] proposed a two-stream network, combining scores from separate appearance (RGB) and motion streams (stacked optical flows). The original approach was improved by more advanced fusion in [70, 71]. A different school of thought models motion using variants of 3D convolutions including C3D proposed in [78]. Inflated 3D (I3D) network, leveraging 3D convolutions within a two-stream setup was proposed in [76]. To cope with egocentric motion captured by head-mounted cameras, Singh *et al*. introduced a third stream (EgoStream) in [73], capturing the relation of hands, head, and eyes motion. [69] further used four streams (two appearance and two motion streams) in Multi-Stream Network (MSN). Each domain (spatial and temporal) has a global view (whole frame) and a local view (cropped by motion tracker).

Long-temporal modeling. While spatiotemporal features are usually extracted over short periods of time, some form of long-temporal modeling is performed to capture long-term dependencies within the entirety of a video containing an action sequence. In [5] Spatio-temporal CNN (ST-CNN) was introduced to combine SpatialCNN features using a 1D convolution that spans over a long period of time. Singh *et al*. learned the long-term dependency from MSN features (four-stream) using bi-directional LSTMs [69]. More recently, [5] proposed two Temporal Convolution Networks (TCN): DilatedTCN and Encoder-Decoder TCN (ED-TCN). These networks fused SpatialCNN features and captured long-temporal patterns by convolving them in the time-domain. A Temporal Deformable Residual Networks (TDRN) was proposed in [82] to model long-temporal information by applying a deformable convolution in the time domain. The TCN model was also further improved with multi stage mechanism in Multi-Stage TCN (MS-TCN) [87].
Network architectures. Pre-trained architectures for image classification, such as VGG, Inception, ResNet [88, 89, 90] are the most important determinants of the performance of the main down-stream vision tasks. Many papers have focused on improving the recognition accuracy by innovating on the network architecture. In standard convolutions, the convolutional response always comes from a local region. Dilated convolutions have been introduced to overcome this problem by changing the shape of receptive fields with some dilation patterns [79, 80, 81]. In 2017, Dai *et al*. [83] introduced deformable convolutional networks with adaptive receptive fields. The method is more flexible since the receptive fields depend on input and can approximate an arbitrary object's shape. We leverage on the advances of [83], specifically the adaptive receptive fields from the model to capture motion in the *feature space*. We further add a local coherency constraint on receptive fields in order to ensure that the motion fields are consistent. This constraint also plays a major role in reducing model complexity.

4.3 Locally-Consistent Deformable Convolution Networks

Our architecture builds upon deformable convolutional networks with an underlying ResNet CNN. While a deformable convolutional network has been shown to succeed in the task of object detection and semantic segmentation, it is not directly designed for fine-grained action detection. However, we observe that de-

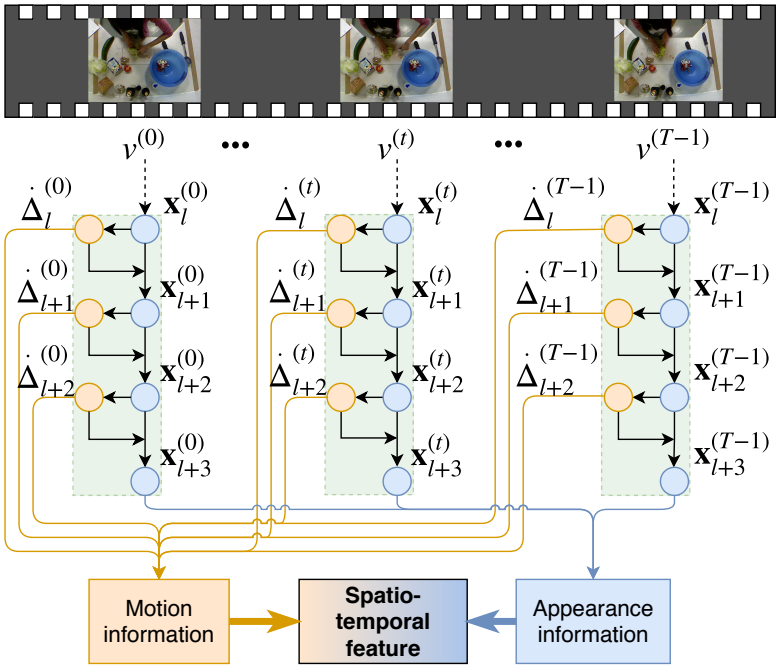


Figure 4.2: Network architecture of the proposed LCDC across multiple frames $v^{(t)}$. Appearance information comes from the last layer while motion information is extracted directly from deformation $\dot{\Delta}$ in the feature space instead of from a separate optical flow stream. Weights are shared across frames over time.

formable convolution layers have a byproduct, the *adaptive receptive field*, which can capture motion very naturally.

At a high level, an adaptive receptive field in a deformable convolution layer can be viewed as an aggregation of important pixels, as the network has the flexibility to change where each convolution samples from. In a way, the adaptive receptive fields are performing some form of key-points detection. Therefore, our hypothesis is that, if the key-points are consistent across frames, we can model motion by taking the difference in the adaptive receptive fields across time. As a deformable convolution can be trained end-to-end, our network can learn to model motion at hidden layers of the network. Combining this with spatial features leads to a powerful spatiotemporal feature.

We illustrate the intuition of our method in Fig. 4.1. The motion here is computed using difference in adaptive receptive fields on multiple *feature spaces* instead of pixel space as in optical flow. Two consecutive frames of action *cutting lettuce* from 50 Salads dataset are shown in Fig. 4.1a and Fig. 4.1b. Fig. 4.1e shows masks of the person to illustrate how the action takes place. We also show the motion vectors corresponding to different regions in Fig. 4.1c and Fig. 4.1d.

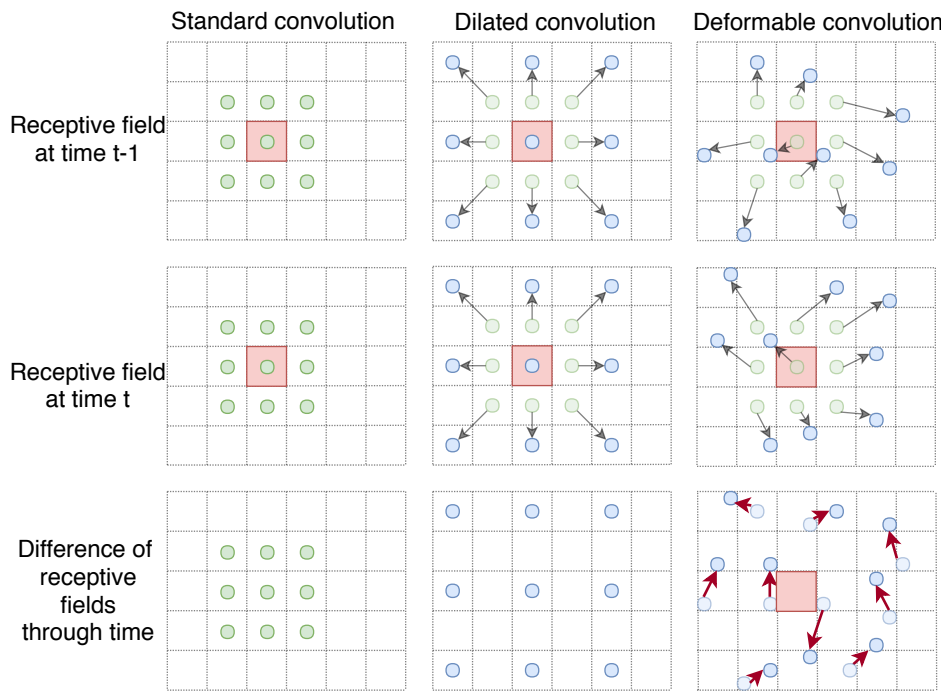


Figure 4.3: Illustration of temporal information modeled by the difference of receptive fields at a single location in 2D. Only deformable convolution can capture temporal information (shown with red arrows). Related to Eq. 4.2 and Eq. 4.3, n is red square, n+k are green dots, $\ddot{\Delta}_{n,k}$ are black arrows, n+k+$\ddot{\Delta}_{n,k}$ are blue dots, and $\ddot{\mathbf{r}}$ are red arrows.

Red arrows are used to describe the motion and green dots are used to show the corresponding activation units. We suppress motion vectors with low values for the sake of visualization. In Fig. 4.1c, the activation unit lies on a background region (cut ingredients inside the bowl) and so there is no motion recorded as the difference between two adaptive receptive fields of background region over time is minimal. However, we can find motion in Fig. 4.1d (the field of red arrows) because the activation unit lies on a moving region, *i.e.*, the arm region. The motion field at all activation units is seen in Fig. 4.1f, where the field's energy corresponds to the length of motion vectors at each location. The motion field is excited around the moving region (the arm) while suppressed in the background. Therefore, this highly suggests that the motion information we extract can be used as an alternative solution to optical flow. A schematic of the proposed network architecture is shown in Fig. 4.2.

4.3.1 Deformable convolution

We first briefly review the deformable convolution layers, before going into a concrete description of the construction of the network architecture. Let $\mathbf{x}$ be the input signal such that $\mathbf{x} \in \mathbb{R}^N$. The standard convolution is defined as:

$$\mathbf{y}[n] = \sum_k \mathbf{w}[-k]\mathbf{x}\left[n+k\right], \tag{4.1}$$

where $\mathbf{w} \in \mathbb{R}^K$ is the convolutional kernel, n and k are the signal and kernel indices (n and k can be treated as multidimensional indices). The deformable convolution proposed in [83] is thus defined as:

$$\mathbf{y}[n] = \sum_k \mathbf{w}[-k]\mathbf{x}\left(n+k+\ddot{\Delta}_{n,k}\right), \tag{4.2}$$

where $\ddot{\Delta} \in \mathbb{R}^{N\times K}$ represents the deformation offsets of deformable convolution. These offsets are learned from another convolution with $\mathbf{x}$ *i.e.*, $\ddot{\Delta}_{n,k} = (\mathbf{h}_k * \mathbf{x})[n]$, where $\mathbf{h}$ is a different kernel. Note that we use parentheses $(\cdot)$ instead of brackets $[\cdot]$ for $\mathbf{x}$ in Eq. 4.2 because the index $n+k+\ddot{\Delta}_{n,k}$ requires interpolation as $\ddot{\Delta}$ is fractional.

4.3.2 Modeling temporal information with adaptive receptive fields

We define the adaptive receptive field of a deformable convolution at time t as $\ddot{\mathbf{F}}^{(t)} \in \mathbb{R}^{N\times K}$ where $\ddot{\mathbf{F}}^{(t)}_{n,k} = n + k + \ddot{\Delta}^{(t)}_{n,k}$. To extract motion information from adaptive receptive fields, we take the difference of the receptive fields through time, which we denote as:

$$\ddot{\mathbf{r}}^{(t)} = \ddot{\mathbf{F}}^{(t)} - \ddot{\mathbf{F}}^{(t-1)} = \ddot{\Delta}^{(t)} - \ddot{\Delta}^{(t-1)}. \tag{4.3}$$

It can be seen that the locations $n+k$ are canceled, going from $t-1$ to t in Eq. 4.3, leaving only the difference of deformation offsets. Given T input feature maps with spatial dimension $H \times W$, we can construct T different $\ddot{\Delta}^{(t)}|_{t=0}^{T-1}$, resulting in $T-1$ motion fields $\ddot{\mathbf{r}}^{(t)}|_{t=0}^{T-2}$ with the same spatial dimension. Therefore, we can model different motion at different positions n and time t.

Fig. 4.3 further illustrates the meaning of $\ddot{\mathbf{r}}^{(t)}$ in 2D for different types of con-

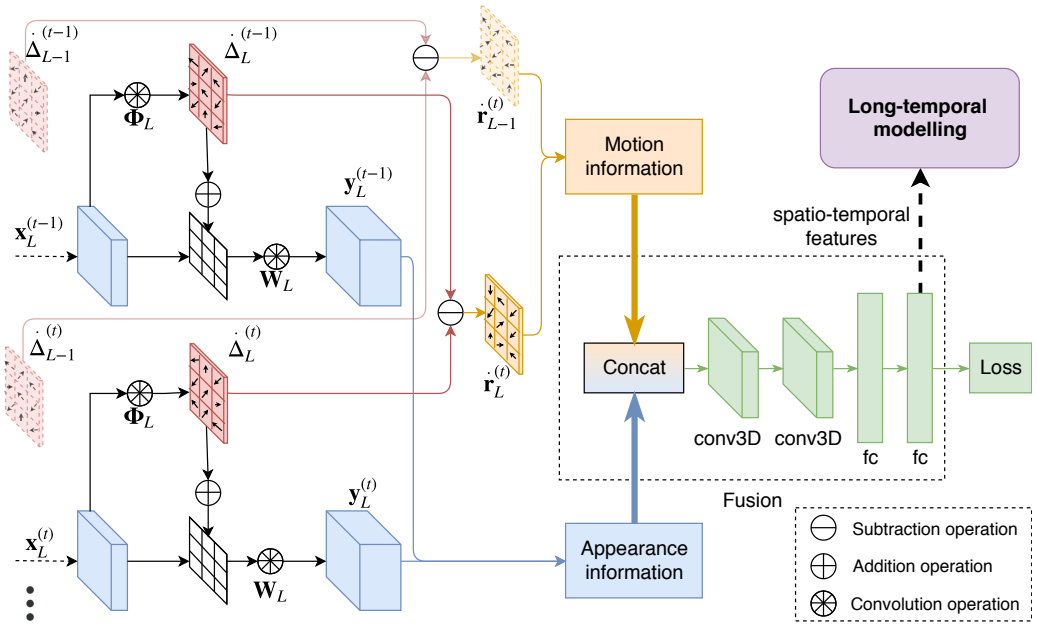


Figure 4.4: A more detailed view of our network architecture with the fusion module. Appearance information comes from output of the last layer while motion information comes from aggregating $\dot{\mathbf{r}}$ from multiple layers. Outputs of the final fc layer can be flexibly used as the features for any long-temporal modeling networks.

volutions. Red square shows the current activation location, green dots show the standard receptive fields, and blue dots show the receptive fields after adding deformation offsets. In the last row, red arrows show the changes of receptive field from time $t-1$ (faded blue dots) to time t (solid blue dots). Readers should note that there are no red arrows for standard convolution and dilated convolution because the offsets are either zero or identical. Red arrows only appear in deformable convolution, which motivates modeling of temporal information.

4.3.3 Locally-consistent deformable convolution

Directly modeling motion using $\ddot{\mathbf{r}}$ is not very effective because there is no guarantee of local consistency in receptive fields in the original deformable convolution formulation. This is because $\ddot{\Delta}_{n,k}$ is defined on both location (n) and kernel (k) indices, which essentially corresponds to $\mathbf{x}[m]$, where $m = n + k$. However, there are multiple ways to decompose m, *i.e.*, $m = n + k = (n - l) + (k + l)$, for any l. Therefore, one single $\mathbf{x}[m]$ is deformed by multiple $\ddot{\Delta}_{n-l,k+l}$, with different l. This produces inconsistency when we model $\ddot{\mathbf{r}}^{(t)}$ in Eq. 4.3, as there can be mul-

tiple motion vectors corresponding to the same location. While local consistency could be learned as a side-effect of the training process, it is still not explicitly formulated in the original deformable convolution formulation.

In order to enforce consistency, we propose a locally-consistent deformable convolution (LCDC):

$$\mathbf{y}[n] = \sum_k \mathbf{w}[-k]\mathbf{x}\left(n + k + \dot{\Delta}_{n+k}\right), \tag{4.4}$$

for $\dot{\Delta} \in \mathbb{R}^N$. LCDC is a special case of deformable convolution where

$$\ddot{\Delta}_{n,k} = \dot{\Delta}_{n+k}, \quad \forall n, k. \tag{4.5}$$

We name this as *local coherency constraint*. The interpretation of LCDC is that instead of deforming the receptive field as in Eq. 4.2, we can deform the input signal instead. Specifically, LCDC in Eq. 4.4 can be rewritten as:

$$\mathbf{y}[n] = \sum_k \mathbf{w}[-k]\tilde{\mathbf{x}}[n + k] = (\tilde{\mathbf{x}} * \mathbf{w})[n], \tag{4.6}$$

where

$$\tilde{\mathbf{x}}[n] = (D_{\dot{\Delta}}\{\mathbf{x}\})[n] = \mathbf{x}\left(n + \dot{\Delta}_n\right) \tag{4.7}$$

is a deformed version of $\mathbf{x}$ and $*$ is the standard convolution ($D_{\dot{\Delta}}\{\cdot\}$ is defined as the deforming operation by offset $\dot{\Delta}$).

Both $\ddot{\Delta}$ and $\dot{\Delta}$ are learned via a convolution layer. Recall that $\ddot{\Delta}_{n,k} = (\mathbf{h}_k * \mathbf{x})[n]$, where $\mathbf{x} \in \mathbb{R}^N$ and $\ddot{\Delta} \in \mathbb{R}^{N \times K}$. $\dot{\Delta}$ is constructed similarly, *i.e.*,

$$\dot{\Delta}_n = (\Phi * \mathbf{x})[n], \tag{4.8}$$

where $\dot{\Delta} \in \mathbb{R}^N$. Since $\ddot{\Delta}$ and $\dot{\Delta}$ share the same spatial dimension N and they can be applied for different time frames, $\dot{\Delta}$ can also model motion at different positions and times.

Furthermore, $\dot{\Delta}$ only needs a kernel Φ, while $\ddot{\Delta}$ requires multiple $\mathbf{h}_k$. Therefore, LCDC is more memory-efficient as we can reduce memory cost K times. Implementation-wise, given input feature map $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$, then

$$\ddot{\Delta} \in \mathbb{R}^{(H \times W) \times (G \times K_h \times K_w \times 2)},$$

where G is the number of deformable groups, K_h and K_w are the height and width of kernels, and 2 indicates that offsets are 2D vectors. However, the dimensionality of LCDC offsets $\dot{\Delta}$ is only $\mathbb{R}^{H\times W\times 2}$. We also drop the number of deformable groups G since we want to model one single type of motion between two time frames. Therefore, the reduction in this case is $G\times K_h\times K_w$ times. The parameter reduction is proportional to the number of deformable convolution layers that are used.

We now show that LCDC can effectively model both appearance and motion information in a single network, as the difference $\dot{\mathbf{r}}^{(t)} = \dot{\Delta}^{(t)} - \dot{\Delta}^{(t-1)}$ has a behavior equivalent to motion information produced by optical flow.

Proposition 1. *Suppose that two inputs* $\mathbf{x}^{(t-1)}$ *and* $\mathbf{x}^{(t)}$ *are related through a motion field,* i.e.*,*

$$\mathbf{x}^{(t)}(s) = \mathbf{x}^{(t-1)}\left(s - o(s)\right), \tag{4.9}$$

where $o(s)$ *is the motion at location* $s \in \mathbb{R}^2$*, and* $\mathbf{x}^{(t)}$ *is assumed to be locally varying. Then the corresponding LCDC outputs with* $\mathbf{w} \neq 0$*:*

$$\begin{aligned}\mathbf{y}^{(t)} &= (D_{\dot{\Delta}^{(t)}}\{\mathbf{x}^{(t)}\}) * \mathbf{w},\\ \mathbf{y}^{(t-1)} &= (D_{\dot{\Delta}^{(t-1)}}\{\mathbf{x}^{(t-1)}\}) * \mathbf{w}\end{aligned}$$

are consistent, i.e.*,* $\mathbf{y}^{(t-1)} = \mathbf{y}^{(t)}$*, if and only if* $\forall n$*,*

$$\dot{\mathbf{r}}_n^{(t)} = \dot{\Delta}_n^{(t)} - \dot{\Delta}_n^{(t-1)} = o\left(n + \dot{\Delta}_n^{(t)}\right). \tag{4.10}$$

Notice that in pixel space, $\mathbf{x}$ *are input images and* $o(s)$ *is the optical flow at* s*. In latent space,* $\mathbf{x}$ *are intermediate feature maps and* $o(s)$ *is the motion of feature.*

Proof. With the connection of LCDC to standard convolution, under the assumption that $\mathbf{w} \neq 0$, we have:

$$\begin{aligned}&\mathbf{y}^{(t)} = \mathbf{y}^{(t-1)}\\ \Leftrightarrow\ &D_{\dot{\Delta}^{(t)}}\{\mathbf{x}^{(t)}\} = D_{\dot{\Delta}^{(t-1)}}\{\mathbf{x}^{(t-1)}\}\\ \Leftrightarrow\ &\mathbf{x}^{(t)}\left(n + \dot{\Delta}_n^{(t)}\right) = \mathbf{x}^{(t-1)}\left(n + \dot{\Delta}_n^{(t-1)}\right), \forall n.\end{aligned}$$

Substituting the LHS in the motion relation in Eq. 4.9, we obtain the following

equivalent conditions $\forall n$:

$$\begin{aligned}\mathbf{x}^{(t-1)}\left(n+\dot{\Delta}_n^{(t)}-o(n+\dot{\Delta}_n^{(t)})\right)&=\mathbf{x}^{(t-1)}\left(n+\dot{\Delta}_n^{(t-1)}\right)\\ \Leftrightarrow \dot{\Delta}_n^{(t)}-o(n+\dot{\Delta}_n^{(t)})&=\dot{\Delta}_n^{(t-1)}\\ \Leftrightarrow o\left(n+\dot{\Delta}_n^{(t)}\right)&=\dot{\Delta}_n^{(t)}-\dot{\Delta}_n^{(t-1)}=\dot{\mathbf{r}}_n^{(t)}.\end{aligned}$$

(since $\mathbf{x}^{(t)}$ is locally varying). □

The above result shows that by enforcing consistent output and sharing weights $\mathbf{w}$ across frames, the learned deformed map $\dot{\Delta}_n^{(t)}$ encodes motion information, as in Eq. 4.10. Hence, we can effectively model both appearance and motion information in a single network with LCDC, instead of using two different streams.

4.3.4 Spatiotemporal features

To create the spatiotemporal feature, we further concatenate across channel dimensions the learned motion information $\dot{\mathbf{r}}^{(t)}$ from multiple layers with appearance features (output of the last layer $\mathbf{y}_L^{(t)}$). We illustrate this process in Fig. 4.4. To model the fusion mechanism, we used two 3D convolutions followed by two fc layers. Each 3D convolution unit was followed by batch normalization, ReLU activation, and 3D max pooling to gradually reduce temporal dimension (while the spatial dimension is retained). Outputs of the final fc layer can be flexibly used as the features for any long-temporal modeling networks, such as ST-CNN [4], Dilated-TCN [5], or ED-TCN [5].

4.4 Experiments

4.4.1 Implementation details

We implemented our approach using ResNet50 with deformable convolutions as backbone (at layers `conv5a`, `conv5b`, and `conv5c` as in [83]). Local coherency constraints were added on all existing deformable convolutions layers. For the fusion module, we used a spatial kernel with size 3 and stride 1; and temporal kernel with size 4 and stride 2. We also used pooling with size 2 and stride 2 in 3D max pooling. Temporal dimension was collapsed by averaging. The network

ended with two fully connected layers. Standard cross-entropy loss with weight regularization was used to optimize the model. After training, LCDC features (last fc layer) were extracted and incorporated into long-temporal models. All data cross-validation splits followed the settings of [5]. Frames were resized to 224x224 and augmented using random cropping and mean removal. Each video snippet contained 16 frames after sampling. For training, we downsampled to 6fps on 50 Salads and 15 fps on GTEA, because of different motion speeds, to make sure one video snippet contained enough information to describe motion. For testing, features were downsampled with the same frame rates as other papers for comparison. We used the common Momentum optimizer [91] (with momentum of 0.9) and followed the standard procedure of hyper-parameter search. Each training routine consisted of 30 epochs; learning rate was initialized as 10^{-4} and decayed every 10 epochs with a decaying rate of 0.96.

4.4.2 Datasets

We evaluate our approach on two standard datasets, namely, 50 Salads dataset and GTEA dataset.

50 Salads Dataset [6]: This dataset contains 50 salad making videos from multiple sensors. We only used RGB videos in our work. Each video lasts from 5-10 minutes, containing multiple action instances. We report results for *mid* (17 action classes) and *eval* granularity level (9 action classes) to be consistent with results reported in [5, 4, 82].

Georgia Tech Egocentric Activities (GTEA) [7]: This dataset contains 28 videos of 7 action classes, performed by 4 subjects. The camera in this dataset is head-mounted, thus introducing more motion instability. Each video is about 1 minute long and has around 19 different actions on average.

4.4.3 Baselines

We compare LCDC with several baselines including (1) methods which do not involve long-temporal modeling where comparison is at spatiotemporal feature level (SpatialCNN) and (2) methods with long-temporal modeling (ST-CNN, DilatedTCN, and ED-TCN).

SpatialCNN [4]: a VGG-like model that learns both spatial and *short-term* tem-

poral information by stacking an RGB frame with the corresponding MHI (the difference between frames over a *short* period of time). MHI is used for both 50 Salads and GTEA datasets instead of optical flow as optical flow was observed to suffer from small motion and data compression noise [5, 4]. SpatialCNN features are also used as *inputs* for ST-CNN, DilatedTCN, ED-TCN, and TDRN.

ST-CNN [4], DilatedTCN [5], and ED-TCN [5]: are long-temporal modeling frameworks. Long-term dependency was modeled using a 1D convolution layer in ST-CNN, stacked dilated convolutions in DilatedTCN, and an encoder-decoder with pooling and up-sampling in ED-TCN. All three frameworks were originally proposed with SpatialCNN features as their input. We incorporated LCDC features into these long-temporal models and compared with the original results.

We obtained the publicly available implementations of ST-CNN, DilatedTCN, and ED-TCN from [92]. On incorporating LCDC features into these models, we observed that training from scratch can become sensitive to random initialization. This is likely because these long-temporal models have a low complexity (*i.e.*, only a few layers) and the input features are not augmented. We ran each long-temporal model (with LCDC features) five times and report means and standard deviations over multiple metrics. For completeness, we have also included original results from TDRN (where the input was SpatialCNN features as well) [82]. However, TDRN's implementation was not publicly available so we were unable to incorporate LCDC with TDRN.

4.4.4 Results

We benchmark our approach using three standard metrics reported in [5, 82]: frame-wise accuracy, segmental edit score, and F1 score with overlapping of 10% (F1@10). Since edit and F1 scores penalize over-segmentation, accuracy metric is more suitable to evaluate the quality of short-term spatiotemporal features (SpatialCNN and LCDC). All mentioned metrics are sufficient to assess the performance of long-temporal models (ST-CNN, DilatedTCN, ED-TCN, and TDRN). We have also specified inputs for spatial and short-term temporal components, as well as the long-temporal model in each setup (Table 4.1 and Table 4.2).

Table 4.1 shows the results on 50 Salads dataset on both granularity levels. Overall performance of LCDC setups, with long-temporal models, outperform their counterparts. We highlight our LCDC + ED-TCN setups as they provided

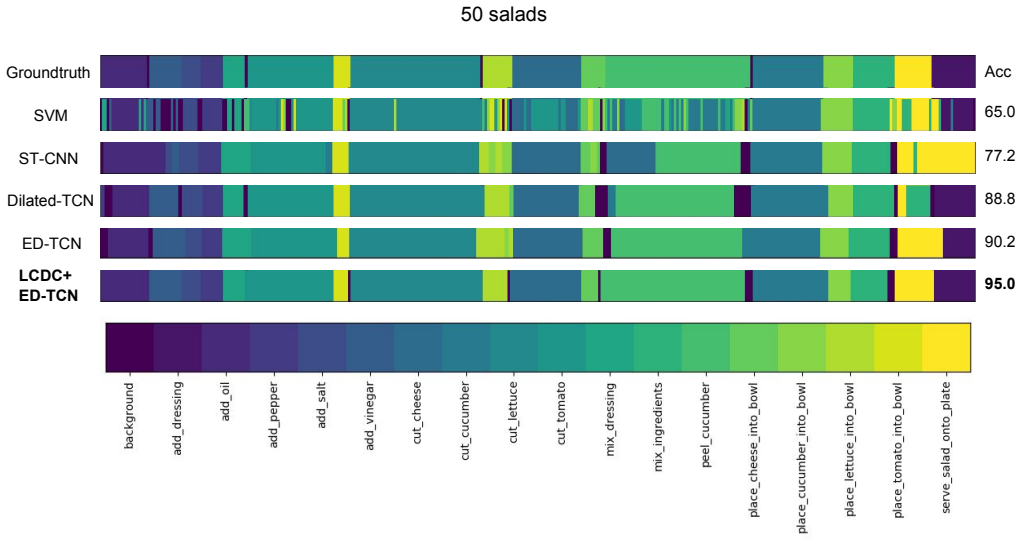
Table 4.1: Results on 50 Salads dataset (*mid* and *eval*-level). Learned deformation is $\dot{\Delta}$ in Eq. 4.8. Means and standard deviations over five runs are reported for LCDC with long-temporal models. Results of baselines are directly reported from their original publications. Please note that since TDRN implementation was not publicly available, LCDC features were not incorporated into TDRN and hence the TDRN results (in parentheses) are not directly comparable with LCDC results.

	Model	Spatial comp	Temporal comp (short)	Long-temporal	F1@10	Edit	Acc
Mid	SpatialCNN [4]	RGB	MHI	-	32.3	24.8	54.9
	(SpatialCNN) + ST-CNN [4]	RGB	MHI	1D-Conv	55.9	45.9	59.4
	(SpatialCNN) + DilatedTCN [5]	RGB	MHI	DilatedTCN	52.2	43.1	59.3
	(SpatialCNN) + ED-TCN [5]	RGB	MHI	ED-TCN	68.0	59.8	64.7
	(SpatialCNN) + TDRN [82]	RGB	MHI	TDRN	(72.9)	(66.0)	(68.1)
	LCDC	RGB	Learned deformation	-	43.99	33.38	67.27
	LCDC + ST-CNN	RGB	Learned deformation	1D-Conv	60.01±0.42	51.35±0.12	68.45±0.15
	LCDC + DilatedTCN	RGB	Learned deformation	DilatedTCN	58.21±0.59	48.54±0.52	69.28±0.25
	LCDC + ED-TCN	RGB	Learned deformation	ED-TCN	**73.75±0.54**	**66.94±1.33**	**72.12±0.41**
Eval	Spatial CNN [4]	RGB	MHI	-	35.0	25.5	68.0
	(SpatialCNN) + ST-CNN [4]	RGB	MHI	1D-Conv	61.7	52.8	71.3
	(SpatialCNN) + DilatedTCN [5]	RGB	MHI	DilatedTCN	55.8	46.9	71.1
	(SpatialCNN) + ED-TCN [5]	RGB	MHI	ED-TCN	76.5	72.2	73.4
	LCDC	RGB	Learned deformation	-	56.56	45.77	77.59
	LCDC + ST-CNN	RGB	Learned deformation	1D-Conv	70.46±0.41	62.71±0.46	77.84±0.26
	LCDC + DilatedTCN	RGB	Learned deformation	DilatedTCN	67.59±0.42	58.97±0.55	78.29±0.29
	LCDC + ED-TCN	RGB	Learned deformation	ED-TCN	**80.22±0.21**	**74.56±0.70**	**78.90±0.25**

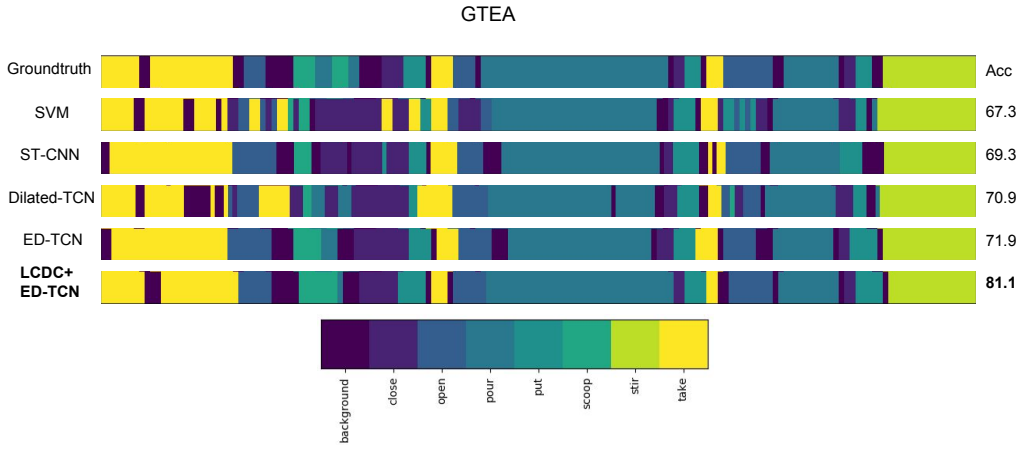
Table 4.2: Results on GTEA dataset. Table format follows the same convention as in Table 4.1.

Model	Spatial comp	Temporal comp (short)	Long-temporal	F1@10	Edit	Acc
SpatialCNN [4]	RGB	MHI	-	41.8	-	54.1
(SpatialCNN) + ST-CNN [4]	RGB	MHI	1D-Conv	58.7	-	60.6
(SpatialCNN) + DilatedTCN [5]	RGB	MHI	DilatedTCN	58.8	-	58.3
(SpatialCNN) + ED-TCN [5]	RGB	MHI	ED-TCN	72.2	-	64.0
(SpatialCNN) + TDRN [82]	RGB	MHI	TDRN	(79.2)	(74.1)	(70.1)
LCDC	RGB	Learned deformation	-	52.42	45.38	55.32
LCDC + ST-CNN	RGB	Learned deformation	1D-Conv	62.23±0.69	55.75±0.94	58.36±0.45
LCDC + DilatedTCN	RGB	Learned deformation	DilatedTCN	62.08±0.85	55.13±0.79	58.07±0.30
LCDC + ED-TCN	RGB	Learned deformation	ED-TCN	**75.39±1.33**	**72.84±0.84**	**65.34±0.54**

Figure 4.5: Comparison of segmentation results across different methods on two test videos (one each for 50 Salads and GTEA dataset). SVM, ST-CNN, DilatedTCN, and ED-TCN are original results with SpatialCNN features. LCDC features are used in conjunction with ED-TCN long-temporal model in the last row. Framewise accuracy is reported for each setup.



(a) 50 Salads dataset (*mid*-level).



(b) GTEA dataset.

Table 4.3: Ablation study on 50 Salads dataset (Split 1, *mid*-level). "Single" and "multi" indicate the amount of input frames for spatial/temporal components.

Model	Spatial comp	Temporal comp (short)	Fusion scheme	Acc	Total params	Deform params
SpatialCNN	RGB (single)	MHI (multi)	Stacked inputs	60.99	-	-
NaiveAppear	RGB (single)	-	-	68.45	38.9M	-
NaiveTempAppear	RGB (multi)	Avg feat frames (multi)	-	71.52	38.9M	-
OptFlowMotion	-	OptFlow (multi)	-	25.67	134.1M	-
TwoStreamNet	RGB (multi)	OptFlow (multi)	Avg scores	71.82	173.0M	-
DC	RGB (multi)	Learned deformation (w/o local coherency) (multi)	3D-Conv	72.25	45.7M	995.5K
LCDC	RGB (multi)	Learned deformation (multi)	3D-Conv	**73.77**	**42.7M**	**27.7K**

the most significant improvement over other baselines. Compared to the original ED-TCN, which used SpatialCNN features, our approach increases by 5.75%, 7.14%, 7.42% on mid-level and 3.72%, 2.36%, 5.5% on eval-level, in terms of F1@10, edit score, and accuracy. Table 4.2 shows the results on GTEA dataset and is organized in a fashion similar to Table 4.1. We achieve the best performance when incorporating LCDC features with ED-TCN framework out of the three baselines. LCDC + ED-TCN also outperforms the original SpatialCNN + ED-TCN on both reported metrics: improving by 3.19% and 1.34%, in terms of F1@10 and accuracy.

We further show segmentation results of test videos from 50 Salads (on *mid*-level granularity) (Fig. 4.5a) and GTEA datasets (Fig. 4.5b). In the figures, the first row is the ground-truth segmentation. The next four rows are results from different long-temporal models using SpatialCNN features: SVM, ST-CNN, DilatedTCN, and ED-TCN. All of these segmentation results are directly retrieved from the provided features in [5], without any further training. The last row shows the segmentation results of our LCDC + ED-TCN. Each row also comes with its respective accuracy on the right. On 50 Salads dataset, Fig. 4.5a shows that LCDC + ED-TCN achieves a 4.8% improvement over original ED-TCN. On GTEA dataset, Fig. 4.5b shows a strong improvement of LCDC over ED-TCN, being 9.2% in terms of accuracy. We also achieve a higher accuracy on the temporal boundaries, *i.e.*, the beginning and the end of an action instance is close to that of ground-truth.

4.4.5 Ablation study

We performed an ablation study (Table 4.3) on Split 1 and *mid*-level granularity of 50 Salads dataset to compare LCDC with SpatialCNN and a two-stream framework. For each setup (each row in the table), we show the inputs for spatial and short-term temporal components, its fusion scheme, frame-wise accuracy, the total number of parameters of the model, and the number of parameters related to deformable convolutions (wherever applicable). Since this experiment focuses on comparing short-term features, accuracy metric is more suitable. We also report whether a component requires single or multiple frames as input.

We evaluate on the following setups:

1. **SpatialCNN:** The features from [4] described in Sec. 4.4.3. Its inputs are stacked RGB frame and MHI.

2. **NaiveAppear:** Frame-wise class prediction using ResNet50 (no temporal information involved in this setup).

3. **NaiveTempAppear:** Appearance stream from conventional two-stream frameworks uses a single frame input and VGG backbone. Therefore, comparing LCDC with the above is not straight-forward. We created an appearance stream with multiple input frames and ResNet50 backbone for better comparison with LCDC. Temporal component was modeled by averaging feature frames (before feeding to two fc layers with ReLU). This model is the same as *NaiveAppear*, except that we have multiple frames per video snippet.

4. **OptFlowMotion:** Motion stream that models temporal component using VGG-16 (with stacked dense optical flows as input). This is similar to the motion component of conventional two-stream networks.

5. **TwoStreamNet:** The two-stream framework obtained by averaging scores from *NaiveTempAppear* and *OptFlowMotion*. We follow the fusion scheme used in conventional two-stream network [72].

6. **DC:** Receptive fields of deformable convolution network (with backbone ResNet50) are used to model motion, but without local coherency constraint.

7. **LCDC:** The proposed LCDC model which additionally enforces local coherency constraint on receptive fields.

Compared to *SpatialCNN*, *NaiveAppear* has a higher accuracy because the *SpatialCNN* features are extracted using VGG-like model while *NaiveAppear* uses ResNet50. The accuracy is further improved by 3.07% by averaging multiple feature frames in *NaiveTempAppear*. Notice that the number of parameters of *NaiveAppear* and *NaiveTempAppear* are the same because the only difference is the number of frames being used as input (averaging requires no parameters). Accuracy from *OptFlowMotion* is lower than other models because the motion in 50Salads is hard to capture using optical flow. This is consistent with the observation in [5, 4] that optical flow is inefficient for the dataset. Combining *OptFlowMotion* with *NaiveTempAppear* in *TwoStreamNet* slightly improves the performance. However, the number of parameters is significantly increased be-

cause of complexity of *OptFlowMotion*. This prevented us from having a larger batch size or training the two streams together.

Both of our DC and LCDC frameworks, which model temporal components as difference of receptive fields, outperform the two-stream approach *TwoStreamNet* with significantly lower model complexities. *DC*, which directly uses adaptive receptive fields from the original deformable convolution, increases the accuracy to 72.25%. LCDC further improves accuracy to 73.77% and with even fewer parameters. This complexity reduction is because LCDC uses fewer parameters for deformation offsets. It means the extra parameters of DC are not necessary to model spatiotemporal features, and thus can be removed. Moreover, if we consider only the parameters related to deformable convolutions, *DC* would require 36x more parameters than *LCDC*. The reduction of 36x matches our derivation in Sec. 4.3.3, where K_h=K_w=3 and G=4. The number of reduced parameters is proportional to the number of deformable convolution layers.

4.5 Conclusion

We introduced locally-consistent deformable convolution (LCDC) and created a single-stream network that can jointly learn spatiotemporal features by exploiting motion in adaptive receptive fields. The framework is significantly more compact and can produce robust spatiotemporal features without using conventional motion extraction methods, *e.g.*, optical flow. LCDC features, when incorporated into several long-temporal networks, outperformed their original implementations. For future work, we plan to unify long-temporal modeling directly into the framework.

CHAPTER 5

ADAPTIVE SPATIOTEMPORAL SAMPLING

5.1 Introduction

Our visual world is highly predictive, making it highly inefficient to process each individual piece of data with the same amount of effort. To cope with it, human perceptual system subconsciously pre-scans the scene to determine important events before actual processing. This mechanism is known as *pre-attentive* processing [93, 94, 95]. The pre-capturing images, although appear to be less clear, help construct a more complete scene perception [96, 97]. Furthermore, the human brain also focuses on certain regions within our *foveal* visual field [96, 98, 99, 100]. These two behaviors are strikingly similar to the objective of our temporal and spatial sampling, respectively.

Sampling has also been one of the most studied problems in various areas of video analysis, such as action recognition and video summarization [101, 102, 103, 104, 105, 30], due to the redundancy between consecutive frames. With the increase in model complexity, it gets progressively expensive to process a single frame. This is even more crucial for resource-limited devices such as AR/VR headsets, like Google Glasses, HoloLens, Occulus VR headsets, Vuzix, *etc*. [106, 107, 108, 109]. However, picking a fixed sampling routine does not guarantee the performance as important information may be under-sampled. Temporally, it is evident that the number of frames required to represent a video vary, depending on the action categories [110]. Therefore, an adaptive sampling rate is preferred as over-sampling results in more computational cost while under-sampling can make performance suffer. Similarly, spatial sampling is also necessary in general computer vision tasks, which is applicable on individual frames of a video sequence. It is preferred to have an adaptive sampling scheme as having a fixed one also leads to similar problems as in time domain.

Motivated by the mechanism of human visual perception, we propose a novel

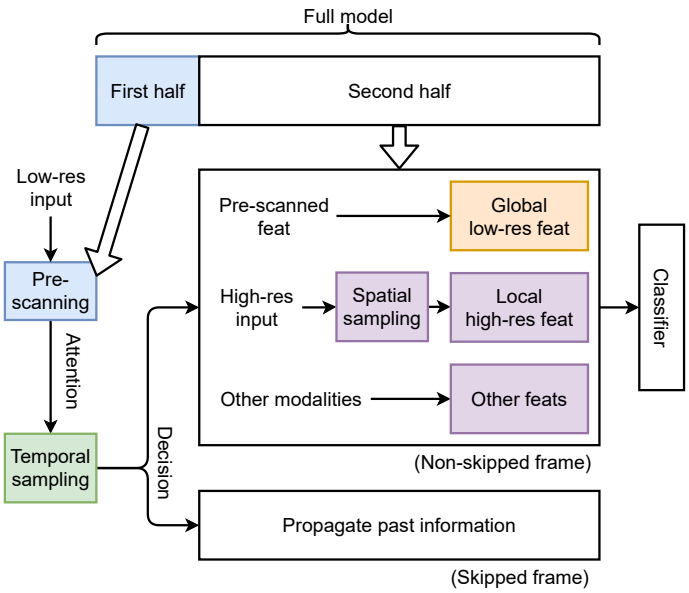


Figure 5.1: Our proposed system has two major components, being temporal and spatial sampling. Based on some pre-scanned features, the temporal sampler decides whether to process the frame fully (top block), or skip to a frame and propagate past information (bottom block). The spatial sampler in turns select RoIs from high-res input to augment the features with low-res inputs. We also include features from other modalities when a frame is not skipped.

adaptive spatiotemporal sampling framework to imitate the human vision. Fig. 5.1 shows an overview of the entire system, with two main components that are built upon the self-attention mechanism: (1) spatial sampler uses the observed attention to sample regions of interest and (2) temporal sampler hallucinates attention in the next frame to model future expectation.

The *spatial sampler* is motivated by human foveal vision. The idea is to only focus on specific regions rather than the whole scene to save computation. It can be seen that lower input sizes significantly reduce the computational complexity. However, it also compromises the performance. To overcome this, we use input at two different resolutions: low-res whole image with size of 112 and high-res image crops with size of 64. The low-res images are processed as a whole for pre-scanning process of temporal sampler and global feature extraction. For the high-res input, we retrieve regions corresponding to the most "important" locations based on the extracted attention and use them to augment the low-res image for the visual recognition task. In our system, we use the low-res image of size 112 and top-k regions of size 64. Fig. 5.2 analyzes the computational complexity in GFLOPS with respect to the spatial dimension of RGB images. We highlight

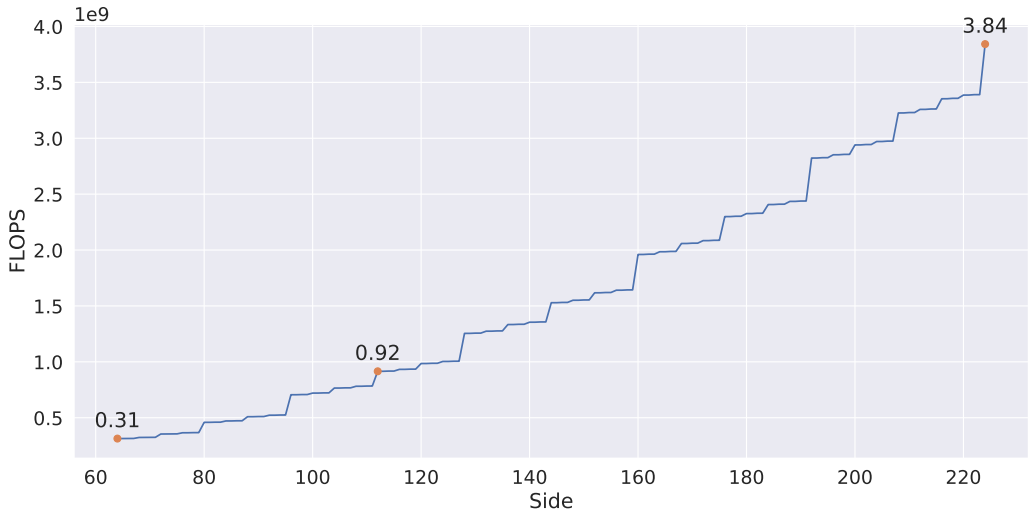


Figure 5.2: Complexity of SAN19 with different input dimensions. The horizontal axis shows the side N of an input with dimension $3 \times N \times N$. We highlight the values where $N = 224, 112, 64$, corresponding to our choices for the size of high-res, low-res, and cropped high-res images.

the GFLOPS with size 224 (high-res baseline), compared with size 112 (low-res input) and size 64 (cropped high-res regions).

The *temporal sampler* follows the concept of pre-attentive processing such that it extracts attention by briefly pre-scanning a low-res input and decides whether to further process the frame if something interesting happens. Since it is possible to predict what would happen in the future [111, 112], we consider an event "interesting" if it is drastically different from what is expected. The idea of using another network for pre-scanning has been discussed in other work [103, 113], however in our proposed approach, we split a backbone network into two halves and use the first one to pre-scan instead of introducing an additional one. Fig. 5.3 shows a complexity analysis on the same model in Fig. 5.2 with low-res input image. Furthermore, we observe that two consecutive frames produce similar attention at some certain intermediate layers, making it possible to pre-scan by forwarding up to such layers. To model future expectation, we hallucinate future attention and compare it with the observed one. When the hallucination matches the actual attention, there is no unexpected event and the model simply uses the previous classification results. Otherwise, the remaining processing routine is carried out to compute new classification.

We demonstrate the effectiveness of our system on the egocentric action recognition task on the large-scale EPIC-KITCHENS 2018 dataset [114]. Our system

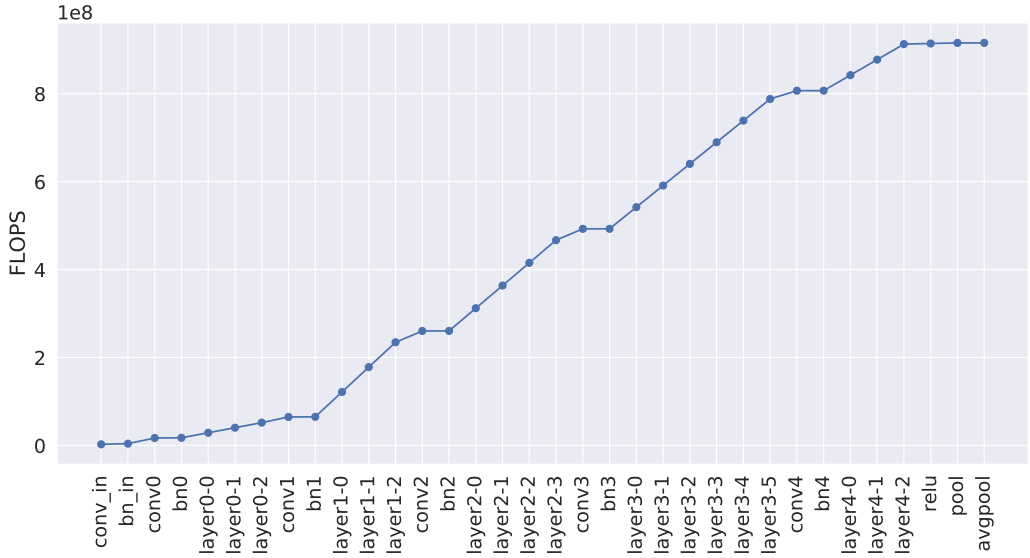


Figure 5.3: Accumulated complexity up to different layers of SAN19, where input size is fixed as 3x112x112, corresponding to the low-res images in Fig. 5.2. The horizontal axis shows the layer names of the model while the vertical axis indicates the accumulated FLOPS up to that layer.

significantly reduces computational complexity compared to the baseline counterpart, with a tolerable loss of accuracy. We also provide qualitative results to reason the sampling results.

Contribution: (1) We present a novel adaptive spatiotemporal sampling scheme for action recognition. It is built upon a temporal sampler that pre-scans the low-res input to decide whether to skip processing by comparing the observed and hallucinated attention and a spatial sampler that selects small high-res RoIs induced by the attention map in pre-scanning process. (2) We showcase the system on first-person videos where our model significantly reduces the computational power with a small loss of accuracy.

5.2 Related Work

Action recognition. With the blooming of deep learning and computer vision, the task of action recognition has evolved from the traditional two-stream networks [115] to more advanced models, *e.g.*, C3D, I3D, ResNet3D, R(2+1)D, TBN, and LSTA [78, 76, 116, 117, 118, 119]. Such standard techniques often demand expensive computation, leading to the challenge of high power consumption [120],

which is crucial for ego-centric action recognition using always-on wearable devices, such as AR/VR glasses. Our adaptive spatiotemporal sampling scheme is designed to address this problem.

Adaptive inference and sampling. Techniques to reduce the complexity of deep networks can be divided into three sub-categories: ignoring layers in deep models, removing input regions, and skipping frames. [121] introduces a stochastic method to drop layers during the training phase. SkipNet [122] and BlockDrop [123] later propose to use reinforcement learning to dynamically drop layers for both training and validation. In spatial domain, RS-Net [124] can decide which resolution to switch to by sharing parameters among different image scales. PatchDrop [125], on the other hand, removes unimportant regions of input images via reinforcement learning. For applications in the area of general video analysis, it is more desirable to rely on time sampling. It has been shown that temporal redundancy results in wasted computation, as some videos only require a single frame to represent [110]. There have been attempts to process videos at multiple frame rates as different actions can happen at different paces [126, 127]. Recently, SC-Sampler [103] and ARNet [113] tackle temporal sampling by using additional simple networks for pre-scanning the features. Our proposed approach addresses all of these areas: using a sub-collection of layers to pre-scan and then sample the inputs, by exploiting self-attention to spatially and temporally determine important information.

Self-attention. In computer vision, gradient-based methods are usually used to generate saliency maps, which can determine the regions where a trained model considers "relevant" to the output [128, 129, 130, 131]. More recently, self-attention is introduced in natural language processing community as a way to direct the focus of deep nets [132]. Since the attention allows a model to focus more on important regions, such self-attention mechanism has been attracting great interest from the computer vision community [133, 134, 135]. Our approach uses such attention as a driving mechanism to find the important regions and frames, allowing spatiotemporal sampling adaptively.

5.3 Approach

Consider a video dataset $\mathcal{D} = \{(\mathbf{v}_n, \mathbf{y}_n)\}_{n=1}^{N}$, where $\mathbf{v}_n$ is a video sequence and $\mathbf{y}_n$ is the corresponding groundtruth label. We assume that the video sequences

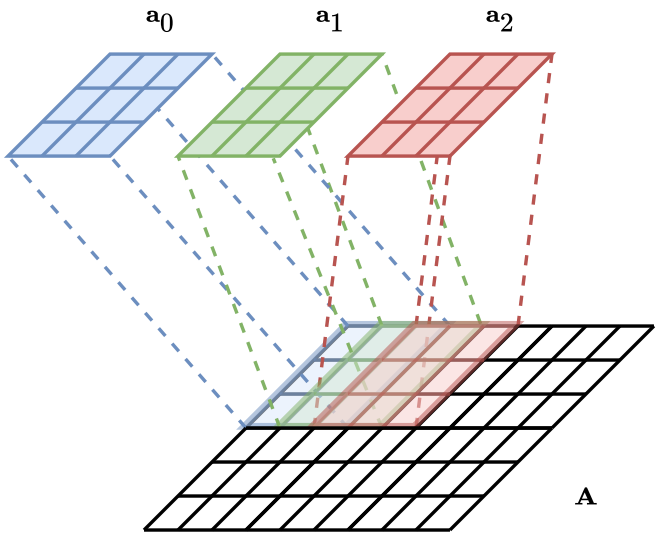


Figure 5.4: Cumulative global attention addresses the sliding effects. The top row is local attention associated with some neighboring footprints. The bottom row shows the global attention aggregated from the local attentions. The sliding effect in this example is similar to how to window kernel moves in convolution (with stride of 1).

have the same length of T frames, *i.e.*, $\mathbf{v}_n = [\mathbf{x}_n^{(1)}, \mathbf{x}_n^{(2)}, \ldots, \mathbf{x}_n^{(T)}]$, where each frame is $\mathbf{x}_n^{(t)} \in \mathbb{R}^{3\times H\times W}, \forall t \in \{1, .., T\}$. Suppose that we have a video classifier $F(\mathbf{v}_n) = \hat{\mathbf{y}}_n$, with some complexity $\mathcal{O}_F$. The goal is to construct another classifier $\tilde{F}$ with less complexity while retaining the accuracy. We address this by introducing the temporal sampler $\mathcal{T}$ and spatial sampler $\mathcal{S}$, *i.e.*, $\hat{\mathbf{y}}_n = \tilde{F}(\mathbf{x}_n; \mathcal{T}, \mathcal{S})$, such that $\mathcal{O}_{\tilde{F}} < \mathcal{O}_F$. At a high level, the spatial sampler chooses the top-k regions based on the most activated areas in the attention map. The temporal sampler decides whether to skip frames, whose attentions are similar to the model's future prediction.

5.3.1 Cumulative global attention

Our cumulative global attention is built upon the pairwise attention formulation of Zhao et al. [133]. This pairwise attention is written as

$$\mathbf{z}_i = \sum_{j\in\mathcal{R}(i)} \alpha(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)) \odot \beta(\phi(\mathbf{x}_j)), \tag{5.1}$$

where $i, j \in \mathbb{R}^2$ are the spatial indices, $\phi(\mathbf{x})$ is the input feature map of a layer, and $\mathbf{z}$ is the output. The compatibility function $\alpha(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j))$ is locally defined over the footprint $\mathcal{R}(i)$ before combining with the projection $\beta(\phi(\mathbf{x}_j))$. We then denote the *local attention* at i as

$$\mathbf{a}_i = \left[\alpha(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j))\right], \quad \forall j \in \mathcal{R}(i). \tag{5.2}$$

Learning to generate such attentions is difficult because we also need to model the underlying relationship of neighboring footprints. However, it is simpler to generate a global attention map where the footprints are already encoded. Thus, we use the *cumulative global attention*, defined as

$$\mathbf{A} = \sum_i \mathbf{a}_i \otimes \mathbb{1}\{\mathcal{R}(i)\}, \tag{5.3}$$

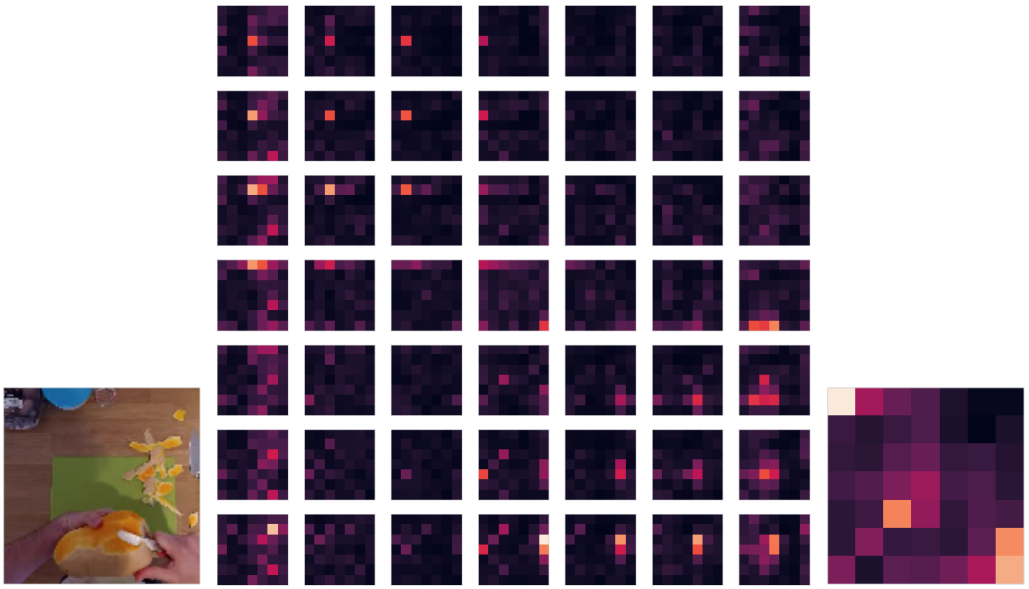
where $\mathbb{1}\{\mathcal{R}(i)\}$ is the indicator function that removes locations outside of footprint $\mathcal{R}(i)$ and $\otimes$ is the multiplication of $\mathbf{a}_i$ with the corresponding footprint. Notice that $\mathbf{a}_i$ has the same spatial dimension as $\mathcal{R}(i)$, while $\mathbf{A}$ has the same spatial dimension as the input feature map $\phi(\mathbf{x})$.

We know that each local attention $\mathbf{a}_i$ is defined with respect to the footprint $\mathcal{R}_i$, as shown in Eq. 5.2. The locations of neighboring footprints here are similar result in overlapping regions similar to the concept of moving the kernel window in convolution. Such overlapping causes the sliding effect, which is observed in neighboring $\mathbf{a}_i$ in Fig. 5.5. Therefore, we can construct a global view of the attention by aggregating the local attention similarly to convolution to remove such sliding effect, as seen in Fig. 5.4. This method creates duplication over the overlapping regions, *i.e.*,

$$\mathbf{A}^* = \mathbf{A} \odot \mathcal{M}, \tag{5.4}$$

where $\mathbf{A}$ is the cumulative global attention defined in Eq. 5.3, $\mathbf{A}^*$ is the clean global attention without overlapping, and $\mathcal{M}$ is the mask counting such duplication. However, we see that the counter mask $\mathcal{M}$ is a constant defined by the size of $\mathbf{A}$ and $\mathbf{a}$. Therefore, using $\mathbf{A}$ instead of $\mathbf{A}^*$ does not affect the quality of our hallucinator.

Fig. 5.5 shows an example of the cumulative global attention (Fig. 5.5b), aggregated from the local attentions across multiple footprints (Fig. 5.5c), given the same input (Fig. 5.5a). We see that the neighboring $\mathbf{a}_i$ have some sliding effect



(a) Input image (b) Local attentions $\mathbf{a}_i$ from different footprint $\mathcal{R}_i$ (c) Cumulative global attention $\mathbf{A}$

Figure 5.5: Local attention and cumulative global attention from the same input image and at the same layer. Hotter color indicates more salient regions. We average across the channel dimension for visualization.

because of how we move the footprints, similar to convolution. Such effect is already encoded in $\mathbf{A}$, making it easier to learn. Specifically, the activation of $\mathbf{A}$ reflects the "important regions" in the input images, being the hands and the bowl (top-left corner). It motivates to use such global attention maps to find ROIs in our spatial sampler. For simplicity, unless stated otherwise we use "attention" to denote the cumulative global attention in the remaining sections.

5.3.2 Spatial sampler

The goal of the spatial sampler is to provide both global and local views of the input image, where the global view has the low-res view of the image and the local view has original high-res image crops. Formally, given input $\mathbf{x} \in \mathbb{R}^{3\times H\times W}$, we define the corresponding low and high-res inputs $\mathbf{x}_l \in \mathbb{R}^{3\times \frac{H}{d}\times \frac{W}{d}}$ and $\mathbf{x}_{h,k} \in$

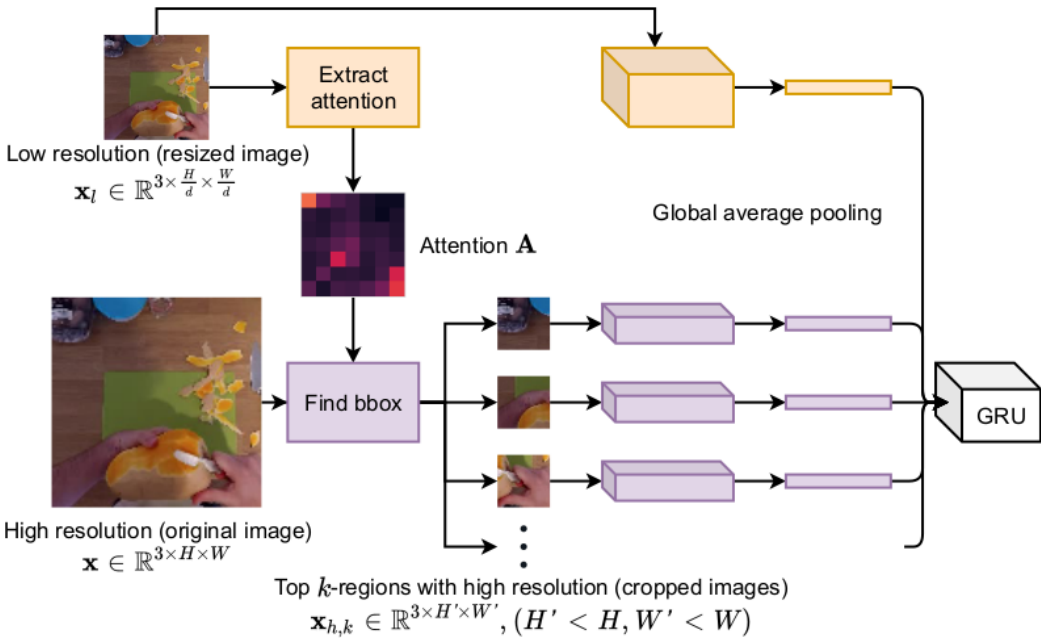


Figure 5.6: Spatial sampler uses attention from low-res image to sample the top-k regions from the (original) high-res input. $\mathbf{x}_l$ gives a global view, while $\mathbf{x}_{h,k}$ provides local views at important regions of the original image $\mathbf{x}$. The global average pooling at the end removes spatial dimension of the features, which are combined and fed to the GRU classifier.

$\mathbb{R}^{3\times H'\times W'}$ as

$$\mathbf{x}_l[:, i] = \mathbf{x}[:, d * i], \tag{5.5}$$

$$\mathbf{x}_{h,k} = \mathcal{S}(\mathbf{x}; \mathbf{A}, k) = \mathbf{x}[:, t_k : b_k, l_k : r_k], \tag{5.6}$$

where $i \in \mathbb{R}^2$ is the 2D spatial index, $d \in \mathbb{R}$ is the down-sampling factor, and $\mathcal{S}$ is the spatial sampler that computes the top-left (t_k, l_k) and bottom-right (b_k, r_k) corners of the bounding box corresponding to the top k-th region. Those regions are constrained to have the same size, *i.e.*, $H' = b_k - t_k$ and $W' = r_k - l_k$. Giving attention $\mathbf{A}$, we find all connected regions and pick the k regions with highest summation. We then linearly project those regions back to pixel space, based on the scaling of spatial dimension between the input image $\mathbf{x}$ and the attention $\mathbf{A}$.

Fig. 5.6 shows the details of the spatial sampler. We extract the attention from the low-res image $\mathbf{x}_l$ and use it to sample the top-k regions in the original image $\mathbf{x}$. This results in $\mathbf{x}_{h,k}$ with lower spatial dimension, while retaining the original resolution of $\mathbf{x}$. As we use the same backbone network to process images of different resolution, we add a global average pooling layer at the end of the feature

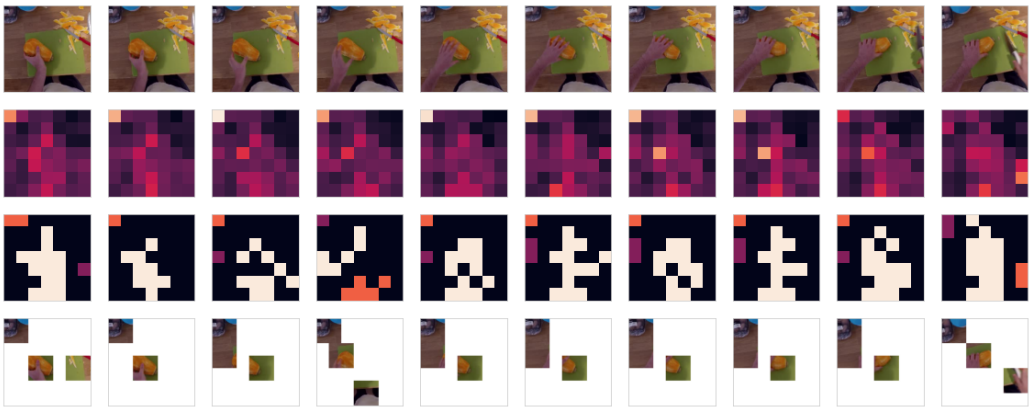


Figure 5.7: Detail of the spatial sampler. The first row is the input RGB frames and the second row is the corresponding attention maps. The third row shows the masks of the top-3 regions, where each color denotes a different region. The bottom row illustrates the sampled regions, corresponding to centroids of each mask.

extractor to remove the spatial dimension. The features are then concatenated and fed to the classifier. We constraint the scaling factor d and the bounding box size H', W' such that the complexity of using $\mathbf{x}_l$ and $\mathbf{x}_{h,k}$'s is less than that of $\mathbf{x}$. Here, we choose $d = 2$ and $H' = W' = 64$, based on our complexity analysis in Fig. 5.2.

Giving an attention from a input image, the sampler suppresses the low values to retrieve the regions with high saliency. However, this could result in multiple fragmented regions. To avoid this, we apply non-maximum supression (NMS) to clean up the candidate regions. We then select the top-k regions based on the its score, defined as sum of all pixels inside the region. Such score scales according to both the values and the size of a region. The results of this procedure is illustrated in the third row of Fig. 5.7. The centroids of such regions (in attention plane) are projected by scaling to find the corresponding centroids in image plane. The sampler then find the bounding boxes surrounding such centroids to generate the regions of interest, as shown in the bottom row of Fig. 5.7. Furthermore, to maintain consistent trajectories of the bounding boxes across frames, we reorder them by using Dijkstra's algorithm. Specifically, we find the box in frame $t + 1$ that is the closest to the query box in frame t, where the distance between two arbitrary boxes is defined as L_2 distance of the respective corners plus their size difference.

In Fig. 5.8, we illustrate some example results of our spatial sampler, extracting the top three regions in a few frames of a video sequence. The colors here

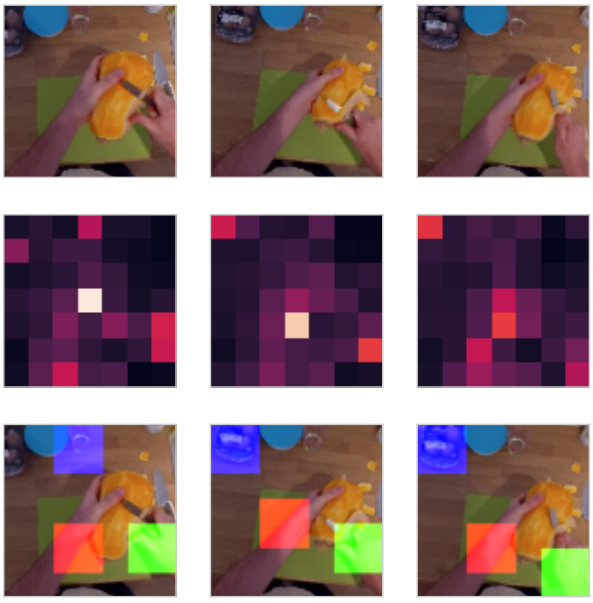


Figure 5.8: Sampled regions from the top-3 spatial sampler. From top to bottom: (1) input frames, (2) attention, and (3) bounding boxes in pixel space. Red, green, and blue colors denote the top 1, 2, and 3 accordingly.

denote the order of the bounding boxes, based on the most activated regions in the attention. It is observed that the sampled regions are not varying rapidly when the activation are similar. This usually happens when the actions are occurring slowly, suggesting that we can predict future attentions in such cases.

5.3.3 Hallucinator

Our hallucinator H is grounded on the notation of temporal consistency, *i.e.*, the attentions of consecutive frames $\mathbf{A}^{(t)}$ and $\mathbf{A}^{(t+1)}$ are similar if the action is slow enough. Intuitively, the attention can reflect the important regions of input images and is not expected to change drastically between consecutive frames. We use H to predict future attention, from which the model can decide to skip future frames if the prediction matches the observed pre-scanned features. The hallucinator H is written as

$$\tilde{\mathbf{A}}^{(t+1)} = H(\mathbf{A}^{(t)}) \quad \text{s.t.} \quad \tilde{\mathbf{A}}^{(t+1)} \approx \mathbf{A}^{(t+1)}, \tag{5.7}$$

where $\tilde{\mathbf{A}}^{(t+1)}$ is the hallucination (predicted future attention). To quantify the similarity between $\tilde{\mathbf{A}}^{(t+1)}$ and $\mathbf{A}^{(t+1)}$, we use the structural similarity index measure (SSIM) [136] as this metrics can compare the structure of input tensors. We train

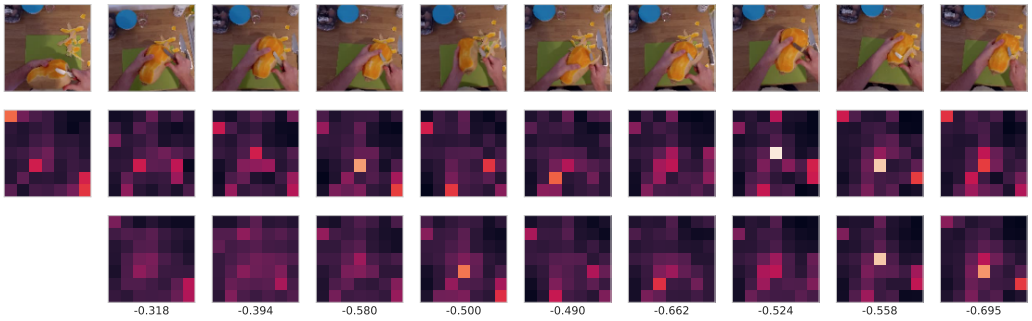


Figure 5.9: Attention and corresponding hallucination of a video sequence. From top to bottom: (1) input frames, (2) attention, and (3) hallucination. Negative SSIM scores between the attention and hallucination are included at the bottom (0 means most different and -1 means most similar). Activated regions of the attentions and hallucinations match the movements of the hands across frames, showing the temporal consistency property.

the hallucinator by minimizing our belief loss:

$$\mathcal{L}_b = -\frac{1}{T-1}\sum_{t=2}^{T} \text{SSIM}(H(\mathbf{A}^{(t-1)}), \mathbf{A}^{(t)}), \tag{5.8}$$

where the function SSIM() computes the structural similarity between the hallucination $H(\mathbf{A}^{(t-1)})$ and the attention $\mathbf{A}^{(t)}$. We minimize the negative SSIM score since the default SSIM ranges from 0 to 1. Higher SSIM indicates more similarity. We build the hallucinator as a convolutional LSTM [137] with encoder-decoder layers and apply teacher forcing technique [138] for the training routine.

Fig. 5.9 shows an example of the hallucination from a video sequence, where the first row is the input video sequence, the second row is the attention extracted from a layer, and the last row is the hallucination, generated by our hallucinator. There is a missing hallucination at the first frame because we are generating future attention. It is observed that the most activated regions of the attention here are located around the two hands. As the hands move in time, these regions also move with a similar manner, in both the attention and hallucination. It suggests that our hallucinator can predict where the important regions would be in the future. We also provide the negative SSIM scores at the bottom to compare the structural similarity between the attention and hallucination. Note that our objective here is not to generate a perfect hallucination, but only to use it as a guideline for the temporal sampler.

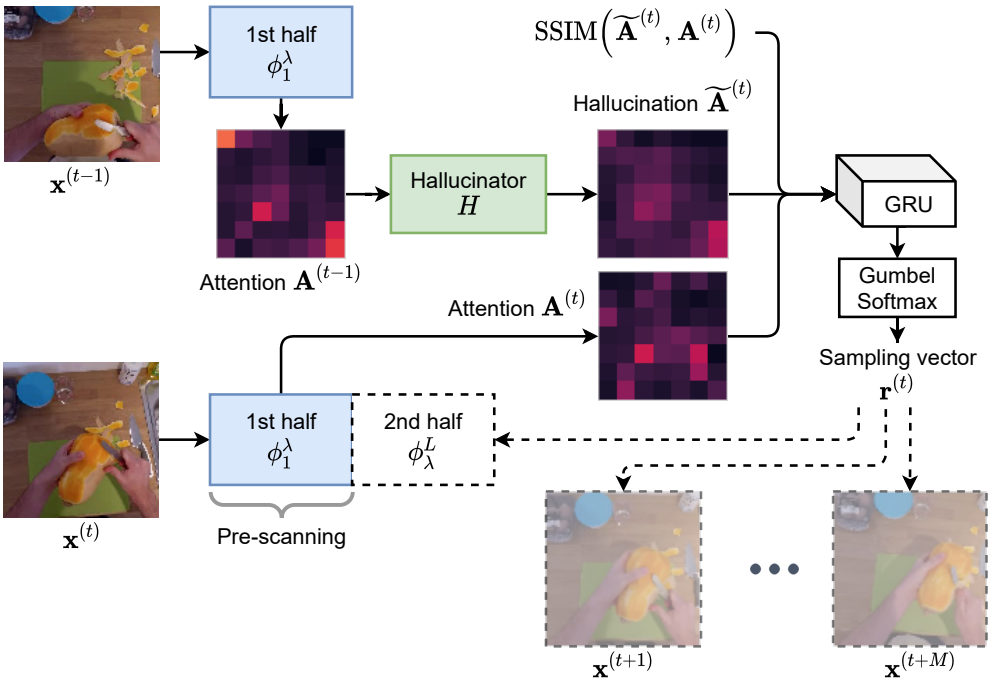


Figure 5.10: Temporal sampler with inputs at $t-1$ and t. Attention from the model's first half at time t, hallucination computed at time $t-1$, and their SSIM score are fed to a GRU to compute the sampling vector $\mathbf{r}^{(t)}$, deciding how many frames to skip (including the second half of the current frame). Model weights are shared across frames.

5.3.4 Temporal sampler

Given a video sequence $\mathbf{v} = [\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(T)}]$, the objective of the temporal sampler is to adaptively select a subset of "important" frames that can still represent $\mathbf{v}$. A frame $\mathbf{x}^{(t)}$ is considered as unimportant if we can reasonably predict its the attention. From Sec. 5.3.1, we know that the attention and hallucination can be retrieved at any arbitrary layer from a model of L layers. Suppose that the attention is extracted at layer $\lambda < L$, it is wasteful to compute the last $L - \lambda$ layers if the temporal sampler decides to skip this frame. In other words, we can forward a frame up to layer λ and choose to run the rest of the model adaptively.

Formally, consider the feature extractor of a deep network of L layers as a composite function, we can split it into two halves:

$$\phi_1^L(\mathbf{x}) = \phi_\lambda^L \circ \phi_1^\lambda(\mathbf{x}), \tag{5.9}$$

where l is the layer where we want to split. We call ϕ_1^λ and ϕ_λ^L the first and second

half, respectively. ϕ_1^λ is used for pre-scanning while ϕ_λ^L can also be augmented with information from other modalities for the classification task later. The temporal sampler $\mathcal{T}$ determines the sampling routine by computing a sampling vector $\mathbf{r} = [\mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(T)}]$, *i.e.*, $\mathcal{T}(\mathbf{v}) = [\mathbf{x}^{(t)} \times \mathbf{r}^{(t)}]_{t=1}^{T}$, with

$$\mathbf{r}^{(t)} \in \{0, 1\}^{M+1} \quad \text{s.t.} \quad \sum_{m=0}^{M} \mathbf{r}^{(t)}[m] = 1, \forall t, \tag{5.10}$$

where M is the maximum number of frames to skip. Fig. 5.10 shows the details of the temporal sampler. At time t, the attention $\mathbf{A}^{(t)}$ is extracted using the first half of the feature extractor ϕ_1^λ. It is flattened and concatenated with the hallucination and the corresponding SSIM score, and then fed to a GRU. The output features are fed to a Gumbel Softmax layer [139] to produce differentiable sampling vector $\mathbf{r}^{(t)}$.

Let $m^* = \text{argmax}_m \, \mathbf{r}^{(t)}[m]$, indicating how many frames we can skip ahead, there are two possible scenarios: $m^* = 0$ and $m^* \in [1, M]$. In the *first* case, we do not skip anything and continue to run the remaining part of the network, thus the complexity is that of the full pipeline $\mathcal{O}_{full}$. In the *second* case, we only pre-scan the current frame, which has already been done, and skip computation on the next $m^* - 1$ frames. The classification results and memory from recurrent models are propagated accordingly. The complexity for these m^* frames is $\mathcal{O}_{pre} = \mathcal{O}_{\phi_1^\lambda} + \mathcal{O}_H + \mathcal{O}_\mathcal{T}$. Note that $\mathcal{O}_{full} = \mathcal{O}_{pre} + \mathcal{O}_{rest}$, where $\mathcal{O}_{rest}$ is the complexity of running the rest of the pipeline, including spatial sampler, other modalities, and classifier. Under such policy, we train the temporal sampler by minimizing the efficiency loss $\mathcal{L}_e$, which is recursively defined as

$$(\mathcal{L}_e, t) \leftarrow \begin{cases} (\mathcal{L}_e + \mathcal{O}_{full}, t + 1), & m^* = 0 \\ (\mathcal{L}_e + \mathcal{O}_{pre}, t + m^*), & \text{otherwise} \end{cases}. \tag{5.11}$$

Without any constraints, it is possible that no frame would be fed to the second half of the pipeline, *i.e.*, $\text{argmax}_m \, \mathbf{r}^{(t)}[m] \neq 0, \forall t$. To avoid such scenario, we include a warm-up step, where the full pipeline is run at the first frame. It also helps initialize memory for recurrent models and ensures that we have classification result for at least one frame. This, however, is not necessary for $\mathcal{L}_e$ since we are only adding a constant.

5.4 Experiments

We evaluate our system on the large-scale video dataset EPIC-KITCHENS 2018 [114], following the training and validation splits of [118]. This is an egocentric dataset with 55 hours of full-HD, 60fps videos, where the actions are recorded in the kitchen environments. Each video sequence is associated with a verb (125 classes) and a noun (331 classes) label. The label of an action is thus defined as a pair of the corresponding verb and noun labels, *e.g.*, [cut, squash] and [open, container].

We use two modalities of inputs in all of our experiments, namely *RGB* and *Spectrogram*, corresponding to the vision and audio domains. Although the optical flow is provided as a part of the dataset, we avoid using it because such source of data is computationally expensive in real-life scenarios. We treat RGB inputs as the guiding modality of the system because our hallucinator and samplers rely on the attention from vision data. The spectrogram inputs act as the additional source of information and are only used when a frame is not skipped by the temporal sampler.

We benchmark our system with top-1 and top-5 accuracy, corresponding to the three domains: action, verb, and noun. To assess the system's efficiency, we further report the models' FLOPS per frame, which is proportional to inference time and power consumption. Since the model complexity is time-variant for the experiments with temporal sampler, we instead provide the accumulated FLOPS over the whole validation set and the average FLOPS per frame, which is the mean complexity with respect to skipped, pre-scanned, and non-skipped frames.

5.4.1 Implementation details

Our system uses SAN19 with pairwise self-attention (equivalent to ResNet50 [133]) as the backbone network to extract features. The Spectrogram (256x256) is constructed from the audio channels using the same processing procedure as in [118]. We choose to extract attention at layer3-0 of the backbone network because it shows good trade-off between complexity and performance in our experiments. This gives us the attention feature map with the dimensionality of 32x7x7. Please refer to the supplement for additional analysis of picking layer 3-0 over others for the attention extraction.

The hallucinator is a conv LSTM with 1 layer and 32 hidden dimensions. It is

equipped with a encoder and a decoder, each is a 2D conv layer with kernel of size 3x3 and 32 channels. The action classifier used with our spatial and temporal sampler is a three-head GRU, corresponding to the global features (low-res RGB and Spectrogram), local features (cropped high-res RGB and Spectrogram), and their concatenation to the master GRU head. The goal of the multi-head architecture is to ensure the network extract prominent features from the cropped regions rather than relying solely on the low-res image. Each head of the GRU classifier and our GRU temporal sampler share the same architecture of 2 layers and 1024 hidden dimension.

The whole system is trained in multiple phases. We first train the two feature extraction modules with FC classifier, corresponding to the low-res and high-res inputs. We sample three frames per video sequence and train the models with the standard cross-entropy loss for 100 epochs, using SGD with momentum of 0.9 [91], with decaying at epochs 30, 60, and 90. The weights of feature extraction modules are frozen and used for other models. The hallucinator is then trained using the belief loss $\mathcal{L}_b$ in Eq. 5.8 with teacher forcing routine [138]. For the spatial sampler with three-head classifier, the predictions on all heads are averaged and the model is trained using the loss $\mathcal{L}_{class} = \sum_{h=1}^{3} \theta_h \mathcal{L}_h$, where $\mathcal{L}_h$ is the cross-entropy loss of a head and θ_h is the corresponding scaling. The temporal sampler is jointly trained with the pretrained three-head classifier and the spatial sampler, using the total loss $\mathcal{L}_{class} + \theta_e \mathcal{L}_e$, where $\mathcal{L}_e$ is the efficiency loss described in Eq. 5.11 with the corresponding scaling θ_e. In the experiments, we sample ten frames per video and train the models for 50 epochs using Adam optimizer [140], with decaying at epoch 20 and 40.

5.4.2 Qualitative results

We demonstrate our qualitative results in Fig. 5.11 to show the outputs of both spatial and temporal samplers. The frames are uniformly sampled from the validation videos. We use "red" and "green' color to highlight whether we run the full inference or simply pre-scan a certain frame. The remaining frames are the skipped ones without any computation. The spatial sampler only runs on non-skipped frames to enrich data, therefore the cropped regions are only available on "red" frames. Our spatial sampler here is set up to retrieve the top three regions based on the attention, however some frames are shown to produce only two

(a) cut squash

(b) put squash

(c) put board

(d) open container

Figure 5.11: Qualitative results of spatiotemporal sampling on video sequences. From top to bottom are: the input video sequence, attention, hallucination generated from past attention, and the non-max suppressed top salient regions from the spatial sampler. The frames with red boundary are the non-skipped ones (running full system) while the frames with green boundary denote pre-scanned ones (running the first half). The frames without any boundary are the skipped ones. We choose to use the first frame to initiate the samplers and always associate it with the full inference.

sampled areas because of overlapping.

Overall, the temporal sampler can adaptively sample the frames. The number of "red" frames is significantly fewer than the original video length and can compactly describe the complete action. In Fig. 5.11a, the action cutting squash is a simple example since it can be easily represented using a single frame. We see that aside from the warming up first frame, the temporal sampler here only pre-scans three frames and skip the rest of them. The action of putting down a squash in Fig. 5.11b is another interesting example, where the first and last frame are selected, corresponding to when the actor is holding the squash in hand and placing it on the chopping board. These two sampled frames concisely represent the action putting down is reality. A similar example is illustrated in Fig. 5.11c, where the actor is putting down a chopping board. Fig. 5.11d depicts a more challenging video sequence of opening a container, as the background is not informative and the container are not opened until the final frame. This results in more pre-scanned frames and they appear to be closer than in other sequences.

5.4.3 Quantitative results

Table 5.1 shows quantitative results of the spatial sampler. The first two rows are TBN and SAN19-baseline with FC classifier, both use high-res RGB (224x224) and Spectrogram (256x256). Since TBN relies Inception backbone, its model complexity is not directly comparable with our experiments, with use SAN19 backbone. However, our baseline model provides comparable accuracy. Since the main objective of the paper is to increase efficiency of a given model, we focus on comparing performance and complexity with the baseline SAN19.

The rest of Table 5.1 shows our results of the spatial sampler with GRU classifier, using the low-res RGB (112x112), cropped high-res RGB (64x64), and the same Spectrogram inputs. We denote $\mathcal{S}_k$ as our spatial sampler with top-k RoIs, where $\mathcal{S}_0$ means no spatial sampling involved. It can be seen that by simply decreasing the image resolution, $\mathcal{S}_0$ can reduce the complexity by 2.84 GFLOPS with a small loss of 2.96% in accuracy. By adding the sampled regions from the spatial sampler $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$, the accuracy is improved in general. We achieve the best performance with $\mathcal{S}_2$ across our spatial sampling experiments. This gets close to the performance of the baseline, with 1.41% accuracy different but still can save 2.16 GFLOPS of computation. Furthermore, the efficiency trade-off of

Table 5.1: Results of baselines and spatial sampler $\mathcal{S}$ on EPIC-KITCHENS validation set. The baseline models TBN [118] and SAN19 takes 224x224 RGB inputs whereas our model uses 112x112 resolution RGB frames. The dimension of the Spectrogram is kept as 256x256. The average GFLOPS per-frame is included to indicate the model complexity. We showcase the performance as top-1 and top-5 per-video accuracy for action, verb, and noun domain. We also include the efficiency trade-off (lower means better), computed as the GFLOPS over top-1 accuracy, to show how much GFLOPS is needed with each accuracy percent on average. Our models with spatial samplers are denoted as $\mathcal{S}_k$, where k is the number of RoIs extracted. $\mathcal{S}_0$ means no spatial sampling. $\mathcal{S}_3$ provides the best accuracy among spatial samplers and still with a lower complexity than the baseline.

Model	Avg GFLOPS	Top-1	Top-5	Verb Top-1	Verb Top-5	Noun Top-1	Noun Top-5	Trade-off
TBN [118] (224)	4.62	28.32	60.30	56.96	86.61	41.03	65.35	0.163
SAN19-baseline (224)	8.64	27.52	57.55	55.84	86.24	39.83	62.84	0.314
$\mathcal{S}_0$	5.80	24.56	53.34	53.50	83.95	34.57	58.84	0.236
$\mathcal{S}_1$	6.16	25.77	53.88	56.09	84.82	36.11	59.38	0.239
$\mathcal{S}_2$	6.48	26.11	54.17	55.51	84.74	35.95	59.72	0.248
$\mathcal{S}_3$	6.80	25.31	54.80	55.09	84.65	35.95	60.22	0.268

$\mathcal{S}_2$ is lower, showing that the model with spatial sampler is more efficient in terms of average GFLOPS per accuracy.

Table 5.2 shows our ablation experiment to verify the contribution of the multi-head classifier (Sec. 5.4.1) and bounding boxes processing (Sec. 5.3.2) on the performance of all three spatial samplers $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$. Note that $\mathcal{S}_0$ means no spatial sampling, so these two techniques are not applicable. The experiment shows that multi-head classifiers can consistently improve the performance in all models with a negligible increase of complexity (around 0.05 GFLOPS on average), as each different head of the classifier is only a small GRU. The bounding boxes processing, consisting of non-maximum suppression and reordering of the boxes over time, further increases the accuracy on $\mathcal{S}_1$ and $\mathcal{S}_2$. It should be noticed that only NMS is applicable on $\mathcal{S}_1$ since there is only one bounding box at each frame, making reordering obsolete. On $\mathcal{S}_3$, applying such processing techniques does not enhance the performance. Giving that the bounding boxes correspond to the most salient regions in attention maps, we believe the locations of less salient regions are not as consistent across frames as the more salient ones. Therefore, having too many bounding boxes can make it more difficult to reorder them, thus results in lower accuracy, as seen in the last two rows of Table 5.2. We conjecture that using more sophisticated tracking techniques may help with the reordering; however, it is outside the scope of this work and will be investigated more in future work. We use both multi-head classifier and bounding boxes processing techniques for consistent comparison across experiments since they provide the best accuracy overall.

Table 5.3 shows our results with both spatial and temporal samplers. For better comparison, we include the results of $\mathcal{S}_0$ from Table 5.1 with its accumulated TFLOPS over the whole validation set. Notice that the number of skipped and pre-scanned frames are both zero for $\mathcal{S}_0$ because there is no temporal sampling in this case. We also observe no skipping frames in $\mathcal{T}_1$ because this model only allows either pre-scanning or running the full pipeline. Each block in the table shows the results for a different temporal sampler $\mathcal{T}_M$, where M determines the sampling range, *i.e.*, the maximum number of frames to skip.

In general, the temporal samplers significantly reduce the average GFLOPS compared to $\mathcal{S}_0$, which is also scalable in terms of total TFLOPS. As the sampling range gets wider from $\mathcal{T}_1$ to $\mathcal{T}_4$, we can save more complexity while the accuracy gets more compromised. Furthermore, the speeding up factor is proportional to the sampling range M. We achieve the best accuracy with $\mathcal{S}_3, \mathcal{T}_1$, 0.5% lower than

Table 5.2: Ablation study of multi-head classifier and bounding boxes processing (non-maximum suppression and reordering over time) on EPIC-KITCHENS validation set. Each block shows our results of different settings for the same spatial sampler $\mathcal{S}_k$ (top-k RoIs). The multi-head classifier approach consistently improves the performance of the original counterpart with an insignificant increase in complexity. The bounding box processing technique further enhances the accuracy for $\mathcal{S}_1$ and $\mathcal{S}_2$. We use the spatial sampler with multi-head and bounding box processing in all other experiments for consistent comparison.

Model	Multi -head	Bbox -processing	Avg GFLOPS	Top-1	Top-5	Verb Top-1	Verb Top-5	Noun Top-1	Noun Top-5	Trade-off
$\mathcal{S}_1$			6.12	24.65	51.84	54.75	83.19	34.61	57.21	0.248
$\mathcal{S}_1$	✓		6.16	25.23	54.05	55.17	84.49	35.49	59.68	0.244
$\mathcal{S}_1$	✓	✓	6.16	**25.77**	53.88	56.09	84.82	36.11	59.38	0.239
$\mathcal{S}_2$			6.43	25.27	52.09	54.67	82.24	35.61	57.92	0.254
$\mathcal{S}_2$	✓		6.48	25.69	54.75	55.46	84.95	35.74	60.18	0.252
$\mathcal{S}_2$	✓	✓	6.48	**26.11**	54.17	55.51	84.74	35.95	59.72	0.248
$\mathcal{S}_3$			6.74	24.19	52.46	54.21	83.57	33.95	57.26	0.279
$\mathcal{S}_3$	✓		6.80	**25.69**	55.21	55.00	84.95	36.24	60.38	0.265
$\mathcal{S}_3$	✓	✓	6.80	25.31	54.80	55.09	84.65	35.95	60.22	0.268

Table 5.3: Results of spatial sampler $\mathcal{S}$ and temporal samplers $\mathcal{T}$ on EPIC-KITCHENS validation set. Each block show the results corresponding to the temporal sampler $\mathcal{T}_M$, where M is the maximum number of frames that $\mathcal{T}$ allows to skip. We highlight the best top-1 accuracy and speeding up factor for each $\mathcal{T}_M$ block. The table includes the percentage of frames being skipped (*skip %*), pre-scanned (*scan %*), and not skipped (*full %*), the accumulated Tera-FLOPS over the whole validation set, and the average computation saving compared to its spatial sampler counterpart. All models have temporal sampling, except for the first row, which is copied from Table 5.1 for comparison. All temporal samplers significantly save the compute compared to the baseline system.

Model	Skip (%)	Scan (%)	Full (%)	Total TFLOPS	Avg GFLOPS	Top-1	Top-5	Verb Top-1	Verb Top-5	Noun Top-1	Noun Top-5	Trade -off	Speed up
$\mathcal{S}_0$	0.00	0.00	100.00	139.14	5.80	24.56	53.34	53.50	83.95	34.57	58.84	0.236	-
$\mathcal{S}_0, \mathcal{T}_1$	0.00	41.96	58.04	86.59	3.61	22.81	52.29	52.00	83.07	32.90	57.92	0.158	1.60x
$\mathcal{S}_1, \mathcal{T}_1$	0.00	49.17	50.83	81.27	3.39	22.98	51.63	53.25	83.07	33.15	57.30	0.147	1.71x
$\mathcal{S}_2, \mathcal{T}_1$	0.00	49.96	50.04	83.96	3.50	23.52	52.09	53.34	82.32	32.90	57.92	0.149	1.76x
$\mathcal{S}_3, \mathcal{T}_1$	0.00	50.00	50.00	87.47	3.66	**24.06**	52.88	54.17	83.70	33.74	57.92	0.152	**1.77x**
$\mathcal{S}_0, \mathcal{T}_2$	14.05	52.34	33.61	53.82	2.24	**22.52**	50.75	51.59	82.61	32.15	56.84	0.100	2.58x
$\mathcal{S}_1, \mathcal{T}_2$	14.35	51.58	34.07	56.91	2.37	21.94	51.50	51.50	83.32	33.24	56.88	0.108	2.44x
$\mathcal{S}_2, \mathcal{T}_2$	12.74	52.18	35.08	61.11	2.55	22.23	51.92	51.92	83.28	32.03	57.59	0.115	2.42x
$\mathcal{S}_3, \mathcal{T}_2$	15.02	52.70	32.28	59.27	2.47	21.23	51.50	48.92	83.28	32.36	56.88	0.116	**2.62x**
$\mathcal{S}_0, \mathcal{T}_3$	25.99	48.36	25.65	42.19	1.76	20.64	49.37	49.21	82.40	30.28	55.00	0.085	**3.29x**
$\mathcal{S}_1, \mathcal{T}_3$	25.46	48.54	25.99	44.63	1.86	21.06	50.29	50.00	82.99	31.86	56.09	0.088	3.11x
$\mathcal{S}_2, \mathcal{T}_3$	25.75	48.60	25.64	46.04	1.92	**21.23**	51.71	49.57	83.28	31.23	57.51	0.090	3.21x
$\mathcal{S}_3, \mathcal{T}_3$	25.18	48.92	25.89	48.41	2.02	20.73	50.67	49.99	83.20	31.19	56.24	0.097	3.21x
$\mathcal{S}_0, \mathcal{T}_4$	34.80	44.65	20.55	34.58	1.44	18.85	48.83	47.16	81.78	29.11	54.67	0.077	**4.01x**
$\mathcal{S}_1, \mathcal{T}_4$	32.06	46.16	21.78	38.12	1.59	18.89	49.12	48.79	81.90	30.15	55.34	0.084	3.64x
$\mathcal{S}_2, \mathcal{T}_4$	35.53	40.03	24.44	43.05	1.80	**19.35**	49.42	48.29	82.03	30.61	55.76	0.093	3.43x
$\mathcal{S}_3, \mathcal{T}_4$	34.63	44.52	20.85	39.66	1.65	18.56	47.50	47.00	81.61	29.36	54.34	0.089	3.92x

$\mathcal{S}_0$ with 1.77x speed-up. On the other hand, $\mathcal{S}_0, \mathcal{T}_4$ has the highest speed-up of 4.01x but loses 5.71% of accuracy. However, this model also provides the lowest trade-off factor between accuracy and speed-up. We see that the effect of using spatial sampler to compensate for the loss of accuracy is the most significant for $\mathcal{T}_1$, where more RoIs consistently result in higher top-1 accuracy. More RoIs also improves the speed-up factor of $\mathcal{T}_1$, compared against the corresponding spatial sampler counterparts. More interestingly, we observe that the percentage of pre-scanning frames has a low variance of 0.14% regardless of the sampling range, while the amount of skipping frames increases when the such range raises. This behavior suggests that our temporal sampler knows how to compare the attention of frame t with the hallucination from frame $t-1$, thus knows which frames only need to be pre-scanned. However, it is more challenging for the sampler to predict how many frames ahead to skip. We believe this happens because our hallucination is only predicted for only one future frame and will explore multi-frame hallucination in future work.

5.5 Conclusion

We introduce an attention-based spatiotemporal sampling scheme to adaptively and contextually sample video sequences for efficient action recognition. Spatial sampler provides a global view at a low-res and local salient views at high-res. Temporal sampler pre-scans using the first few layers of a deep network and decides the sampling strategy by comparing the current attention with the generated hallucination from the past. Our system significantly reduces the complexity of a given deep model with a tolerable loss in accuracy. Future exploration includes exploiting different approaches for reordering bounding boxes in spatial sampler and multi-frame hallucination for temporal sampler.

CHAPTER 6

CONCLUSION

The thesis introduces four methods to improve the efficiency in deep networks that use temporal information from different angles. We first study the application of mixed bandwidth for speech recognition. We then switch to the computer vision domain via the task of auto-curation for sports highlights. After that, we look at the problem from a more abstract level by modeling motion in feature space. Finally, we propose an adaptive spatiotemporal sampling technique by imitating the human's visual perception mechanism.

The first project looks at an application for 1D temporal information, *i.e.*, speech recognition. Giving audio signals in WB and NB, it is more efficient to have a single ASR model that can decode both bandwidths instead of deploying separate models for different bandwidths. We train our BWE model discriminatively, as the groundtruth mapping between bandwidths is unavailable. In our investigated case, the best mixed-band model yields lower average WER than the NB baseline and only slight degradation over the WB baseline.

The second project switches the domain of applications to video sequences. The main objective here is to quantify the excitement level of video snippets to produce highlight videos from the most exciting fragments. Since the abundant amount of data makes it inefficient to process every single frame, we decide to recognize the celebrating action at an extremely low frame rate of 1fps and compensate for the performance by combining with other modalities, *i.e.*, the crowd cheering and facial expression. We demonstrate the system to create highlight videos at a golf tournament and two tennis tournaments.

The third project focuses on modeling by formulating motion in feature space. The traditional approach of using optical flow in pixel space often requires an additional stream of inference. Our method enhances the efficiency of spatiotemporal feature extraction models by inducing the temporal feature directly from the feature space, thus eliminating having a second network for motion information. We evaluate our approach using fine-grained action detection problems, where

time is more sensitive than spatial information. Our experiments show that the proposed spatiotemporal features outperform the standard optical flow approach.

The final project aims to model a spatiotemporal sampling scheme for video sequences. Since consecutive frames contain similar information, it is efficient to select only essential frames and regions to process instead of inferencing every single frame. Our approach is inspired by the concepts of pre-attentive processing and foveal vision from the human visual mechanism. Specifically, we only process unexpected events, which are different from the future prediction, and focus more on the regions with high saliency. From our experiments, the system can significantly reduce the complexity with a slight loss of performance.

This work shows great potential for learning efficient temporal information in deep neural networks. Going from 1D to 2D signals and from application to modeling approaches, we see that clever use of time sequences allows a reduction in model complexity. For future work, we target to exploit the availability of static features in an environment, such as the sound, locations, and other physical attributes of landmark objects. Such information carries valuable prior knowledge that helps to narrow the search space for deep neural networks.

REFERENCES

[1] "IBM Watson is creating highlight reels at the Masters," http://www.zdnet.com/article/ibm-watson-is-creating-highlight-reels-at-the-masters/, [Online; accessed 05-August-2019].

[2] "IBM uses AI to serve up Wimbledon highlights," https://www.cnet.com/news/ibm-wimbledon-highlights-artificial-intelligence/, [Online; accessed 05-August-2019].

[3] "Enjoy those U.S. Open highlights. a computer picked them for you." https://www.nytimes.com/2017/09/05/sports/us-open-highlights.html/, [Online; accessed 05-August-2019].

[4] C. Lea, A. Reiter, R. Vidal, and G. D. Hager, "Segmental spatiotemporal CNNs for fine-grained action segmentation," in *European Conference on Computer Vision (ECCV)*, 2016.

[5] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2017.

[6] S. Stein and S. J. McKenna, "Combining embedded accelerometers with computer vision for recognizing food preparation activities," in *International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, 2013.

[7] A. Fathi, X. Ren, and J. M. Rehg, "Learning to recognize objects in egocentric activities," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2011.

[8] K. C. Mac, X. Cui, W. Zhang, and M. Picheny, "Large-scale mixed-bandwidth deep neural network acoustic modeling for automatic speech recognition," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, Sep 2019.

[9] B. Iser, W. Minker, and G. Schmidt, *Bandwidth extension of speech signals*. Springer, 2008.

[10] N. Prasad and T. Kumar, "Bandwidth extension of speech signals: a comprehensive review," *International Journal of Intelligent Systems Technologies and Applications*, vol. 2, no. 2, pp. 45–52, 2016.

[11] F. Nagel and S. Disch, "A harmonic bandwidth extension method for audio codecs," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2009, pp. 145–148.

[12] H. Pulakka and P. Alku, "Bandwidth extension of telephone speech using a neural network and a filter bank implementation for highband Mel spectrum," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 2170–2183, 2011.

[13] C. Liu, Q.-J. Fu, and S. Narayanan, "Effect of bandwidth extension to telephone speech recognition in cochlear implant users," *The Journal of the Acoustical Society of America*, vol. 26, no. 5, pp. 77–83, 2018.

[14] Z.-H. Ling, Y. Ai, Y. Gu, and L.-R. Dai, "Waveform modeling and generation using hierarchical recurrent neural networks for speech bandwidth extension," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 125, no. 2, pp. 883–894, 2009.

[15] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1991.

[16] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[17] K. Li, Z. Huang, Y. Xu, , and C.-H. Lee, "DNN-based speech bandwidth expansion and its application to adding high-frequency missing features for automatic speech recognition of narrowband speech," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2015, pp. 2578–2582.

[18] M. L. Seltzer and A. Acero, "Training wideband acoustic models using mixed-bandwidth training data for speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing,*, vol. 15, no. 1, pp. 235–245, 2007.

[19] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[20] J. Li, D. Yu, J.-T. Huang, and Y. Gong, "Improving wideband speech recognition using mixed-bandwidth training data in CD-DNN-HMM," in *Spoken Language Technology Workshop (SLT)*, 2012, pp. 131–136.

[21] J. Gao, J. Du, C. Kong, H. Lu, E. Chen, and C.-H. Lee, "An experimental study on joint modeling of mixed-bandwidth data via deep neural networks for robust speech recognition," in *International Joint Conference on Neural Networks (IJCNN)*, 2016, pp. 588–594.

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition." in *International Conference on Learning representation (ICLR)*, 2015.

[23] "Nvidia collective communications library (NCCL)," https://developer.nvidia.com/nccl, [Online; accessed 26-October-2018].

[24] M. Merler, K. C. Mac, D. Joshi, Q. Nguyen, S. Hammer, J. Kent, J. Xiong, M. N. Do, J. R. Smith, and R. S. Feris, "Automatic curation of sports highlights using multimodal excitement features," *IEEE Transactions on Multimedia (TMM)*, 2018.

[25] M. Merler, D. Joshi, Q. B. Nguyen, S. Hammer, J. Kent, J. R. Smith, and R. S. Feris, "Automatic curation of golf highlights using multimodal excitement features," in *Conference on Computer Vision and Pattern Recogition Workshops (CVPRW)*, 2017.

[26] D. Joshi, M. Merler, Q.-B. Nguyen, S. Hammer, J. Kent, J. Smith, and R. Feris, "Ibm high-five: Highlights from intelligent video engine," in *ACM International Conference on Multimedia (ACMMM)*, 2017.

[27] Y. Aytar, C. Vondrick, and A. Torralba, "Soundnet: Learning sound representations from unlabeled video," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2016.

[28] Y.-F. Ma, L. Lu, H.-J. Zhang, and M. Li, "A user attention model for video summarization," in *ACM International Conference on Multimedia (ACMMM)*, 2002.

[29] A. Rav-Acha, Y. Pritch, and S. Peleg, "Making a long video short: Dynamic video synopsis," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2006.

[30] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman, "Video summarization with long short-term memory," in *European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 766–782.

[31] J. Zhang, J. Yu, and D. Tao, "Local deep-feature alignment for unsupervised dimension reduction," *IEEE Transactions on Image Processing (TIP)*, vol. 27, no. 5, pp. 2420–2432, 2018.

[32] M. Sun, A. Farhadi, and S. Seitz, "Ranking domain-specific highlights by analyzing edited videos," in *European Conference on Computer Vision (ECCV)*, 2014.

[33] H. Yang, B. Wang, S. Lin, D. Wipf, M. Guo, and B. Guo, "Unsupervised extraction of video highlights via robust recurrent auto-encoders," in *International Conference on Computer Vision (ICCV)*, 2015.

[34] T. Yao, T. Mei, and Y. Rui, "Highlight detection with pairwise deep ranking for first-person video summarization," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2016.

[35] G. Irie, T. Satou, A. Kojima, T. Yamasaki, and K. Aizawa, "Automatic trailer generation," in *ACM International Conference on Multimedia (ACMMM)*, 2010.

[36] G. Evangelopoulos, A. Zlatintsi, A. Potamianos, P. Maragos, K. Rapantzikos, G. Skoumas, and Y. Avrithis, "Multimodal saliency and fusion for movie summarization based on aural, visual, and textual attention," *IEEE Transactions on Multimedia (TMM)*, vol. 15, no. 7, pp. 1553–1568, 2013.

[37] H. Xu, Y. Zhen, and H. Zha, "Trailer generation via a point process-based visual attractiveness model," in *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2015.

[38] J. R. Smith, D. Joshi, B. Huet, W. Hsu, and J. Cota, "Harnessing a.i. for augmenting creativity: Application to movie trailer creation," in *ACM International Conference on Multimedia (ACMMM)*, 2017.

[39] T. Hasana, H. Boril, A. Sangwan, and J. H. L. Hansen, "Multi-modal highlight generation for sports videos using an information-theoretic excitability measure," *EURASIP Journal on Advances in Signal Processing*, 2013.

[40] A. Javed, K. B. Bajwa, H. Malik, and A. Irtaza, "An efficient framework for automatic highlights generation from sports videos," *IEEE Signal Processing Letters (SPL)*, vol. 23, no. 7, 2016.

[41] Z. Zhao, S. Jiang, Q. Huang, and G. Zhu, "Highlight summarization in sports video based on replay detection," in *International Conference on Multimedia and Expo (ICME)*, 2006.

[42] A. Baijal, J. Cho, W. Lee, and B.-S. Ko, "Sports highlights generation based on acoustic events detection: A rugby case study," in *IEEE International Conference on Consumer Electronics (ICCE)*, 2015.

[43] Z. Xiong, R. Radhakrishnan, A. Divakaran, and T. S. Huang, "Audio events detection based highlights extraction from baseball, golf and soccer games in a unified framework," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2003.

[44] Z. Xiong, R. Radhakrishnan, and A. Divakaran, "Generation of sports highlights using motion activity in combination with a common audio feature extraction framework," in *IEEE International Conference on Image Processing (ICIP)*, 2003.

[45] D. Zhang and S.-F. Chang, "Event detection in baseball video using superimposed caption recognition," in *ACM International Conference on Multimedia (ACMMM)*, 2002.

[46] V. Bettadapura, C. Pantofaru, and I. Essa, "Leveraging contextual cues for generating basketball highlights," in *ACM International Conference on Multimedia (ACMMM)*, 2016.

[47] A. Tang and S. Boring, "# epicplay: Crowd-sourcing sports video highlights," in *Conference on Human Factors in Computing Systems (CHI)*, 2012.

[48] T. Decroos, V. Dzyuba, J. Van Haaren, and J. Davis, "Predicting soccer highlights from spatio-temporal match event streams," in *Association for the Advancement of Artificial Intelligence (AAAI)*, 2017.

[49] P. Agrawal, J. Carreira, and J. Malik, "Learning to see by moving," in *International Conference on Computer Vision (ICCV)*, 2015.

[50] D. Jayaraman and K. Grauman, "Learning image representations tied to ego-motion," in *International Conference on Computer Vision (ICCV)*, 2015.

[51] J. Wang, Y. Cheng, and R. Feris, "Walk and learn: Facial attribute representation learning from egocentric video and contextual data," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2016.

[52] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *European Conference on Computer Vision (ECCV)*, 2016.

[53] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2016.

[54] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *European Conference on Computer Vision (ECCV)*, 2016.

[55] H. Mobahi, R. Collobert, and J. Weston, "Deep learning from temporal coherence in video," in *International Conference on Machine Learning (ICML)*, 2009.

[56] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: An astounding baseline for recognition," in *Conference on Computer Vision and Pattern Recogition Workshops (CVPRW)*, 2014.

[57] S. Ma, S. A. Bargal, J. Zhang, L. Sigal, and S. Sclaroff, "Do less and achieve more: Training cnns for action recognition utilizing action images from the web," *Pattern Recognition (PR)*, 2017.

[58] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *European Conference on Computer Vision (ECCV)*, 2016.

[59] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *British Machine Vision Conference (BMVC)*, 2015.

[60] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2015.

[61] R. Smith, "An overview of the tesseract ocr engine," in *Conference on Document Analysis and Recognition (ICDAR)*, 2007.

[62] A. Veit, N. Alldrin, G. Chechik, I. Krasin, A. Gupta, and S. Belongie, "Learning from noisy large-scale datasets with minimal supervision," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2017.

[63] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten, "Exploring the limits of weakly supervised pretraining," in *European Conference on Computer Vision (ECCV)*, 2018.

[64] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning representation (ICLR)*, 2015.

[65] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *ACM International Conference on Multimedia (ACMMM)*, 2014.

[66] K. C. Mac, D. Joshi, R. A. Yeh, J. Xiong, R. S. Feris, and M. N. Do, "Learning motion in feature space: Locally-consistent deformable convolution networks for fine-grained action detection," in *International Conference on Computer Vision (ICCV)*. IEEE, Oct 2019.

[67] S. M. Kang and R. P. Wildes, "Review of action recognition and detection methods," *arXiv preprint arXiv:1610.06906*, 2016.

[68] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele, "A database for fine grained activity detection of cooking activities," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2012.

[69] B. Singh, T. K. Marks, M. Jones, O. Tuzel, and M. Shao, "A multi-stream bi-directional recurrent neural network for fine-grained action detection," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2016.

[70] C. Feichtenhofer, A. Pinz, and R. Wildes, "Spatiotemporal residual networks for video action recognition," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2016.

[71] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2016.

[72] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2014.

[73] S. Singh, C. Arora, and C. V. Jawahar, "First person action recognition using deep learned descriptors," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2016.

[74] J. W. Davis and A. F. Bobick, "The representation and recognition of action using temporal templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2001.

[75] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *International Conference on Computer Vision (ICCV)*, 2013.

[76] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2017.

[77] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, and W. Ouyang, "T-CNN: Tubelets with convolutional neural networks for object detection from videos," *Transactions on Circuits and Systems for Video Technology (TCSVT)*, 2018.

[78] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in *International Conference on Computer Vision (ICCV)*, 2015.

[79] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian, "A real-time algorithm for signal analysis with the help of the Wavelet transform," in *Wavelets*, 1990.

[80] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *International Conference on Learning representation (ICLR)*, 2016.

[81] F. Yu, V. Koltun, and T. Funkhouser, "Dilated residual networks," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2017.

[82] P. Lei and S. Todorovic, "Temporal deformable residual networks for action segmentation in videos," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2018.

[83] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *International Conference on Computer Vision (ICCV)*, 2017.

[84] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 1981.

[85] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2005.

[86] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *European Conference on Computer Vision (ECCV)*, 2006.

[87] Y. A. Farha and J. Gall, "MS-TCN: Multi-stage temporal convolutional network for action segmentation," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2019.

[88] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2016.

[89] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[90] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2015.

[91] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Networks*, 1999.

[92] C. Lea, "Temporal convolutional networks," https://github.com/colincsl/TemporalConvolutionalNetworks, [Online; accessed 20-March-2019].

[93] L. G. Appelbaum and A. M. Norcia, "Attentive and pre-attentive aspects of figural processing," *Journal of Vision*, pp. 18–18, 2009.

[94] M. Atienza, J. L. Cantero, and C. Escera, "Auditory information processing during human sleep as revealed by event-related brain potentials," *Clinical neurophysiology*, pp. 2031–2045, 2001.

[95] X. Meng and Z. Wang, "A pre-attentive model of biological vision," in *International Conference on Intelligent Computing and Intelligent Systems*, 2009, pp. 154–158.

[96] L. Barghout-Stein, *On differences between peripheral and foveal pattern masking*. University of California, Berkeley, 1999.

[97] S. A. Klein, T. Carney, L. Barghout-Stein, and C. W. Tyler, "Seven models of masking," in *Human Vision and Electronic Imaging II*. International Society for Optics and Photonics, 1997, pp. 13–24.

[98] M. Iwasaki and H. Inomata, "Relation between superficial capillaries and foveal structures in the human retina," *Investigative ophthalmology & visual science*, pp. 1698–1705, 1986.

[99] H. Kolb, "Simple anatomy of the retina," *Webvision: The Organization of the Retina and Visual System*, 2011.

[100] Z. Kourtzi and N. Kanwisher, "Cortical regions involved in perceiving object shape," *Journal of Neuroscience*, pp. 3310–3318, 2000.

[101] B. Gong, W.-L. Chao, K. Grauman, and F. Sha, "Diverse sequential subset selection for supervised video summarization," *Advances in neural information processing systems*, vol. 27, pp. 2069–2077, 2014.

[102] M. Gygli, H. Grabner, and L. Van Gool, "Video summarization by learning submodular mixtures of objectives," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2015, pp. 3090–3098.

[103] B. Korbar, D. Tran, and L. Torresani, "SCSampler: Sampling salient clips from video for efficient action recognition," in *International Conference on Computer Vision (ICCV)*, 2019, pp. 6232–6242.

[104] B. Mahasseni, M. Lam, and S. Todorovic, "Unsupervised video summarization with adversarial lstm networks," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2017, pp. 202–211.

[105] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang, "Real-time action recognition with enhanced motion vector cnns," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2016, pp. 2718–2726.

[106] "Google glass," https://www.google.com/glass/start/, 2021, [Online; accessed 03-March-2021].

[107] "Microsoft hololens," https://www.microsoft.com/en-us/hololens/, 2021, [Online; accessed 03-March-2021].

[108] "Oculus," https://www.oculus.com/, 2021, [Online; accessed 03-March-2021].

[109] "Vuzix," https://www.vuzix.com/, 2021, [Online; accessed 03-March-2021].

[110] D.-A. Huang, V. Ramanathan, D. Mahajan, L. Torresani, M. Paluri, L. Fei-Fei, and J. Carlos Niebles, "What makes a video a video: Analyzing temporal information in video understanding models and datasets," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2018, pp. 7366–7375.

[111] R. Gao, B. Xiong, and K. Grauman, "Im2flow: Motion hallucination from static images for action recognition," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2018, pp. 5937–5947.

[112] J. Guan, Y. Yuan, K. M. Kitani, and N. Rhinehart, "Generative hybrid representations for activity forecasting with no-regret learning," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2020, pp. 173–182.

[113] Y. Meng, C.-C. Lin, R. Panda, P. Sattigeri, L. Karlinsky, A. Oliva, K. Saenko, and R. Feris, "AR-Net: Adaptive frame resolution for efficient action recognition," in *European Conference on Computer Vision (ECCV)*, 2020.

[114] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray, "Scaling egocentric vision: The epic-kitchens dataset," in *European Conference on Computer Vision (ECCV)*, 2018, pp. 720–736.

[115] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2014.

[116] K. Hara, H. Kataoka, and Y. Satoh, "Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?" in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2018, pp. 6546–6555.

[117] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2018, pp. 6450–6459.

[118] E. Kazakos, A. Nagrani, A. Zisserman, and D. Damen, "Epic-fusion: Audio-visual temporal binding for egocentric action recognition," in *International Conference on Computer Vision (ICCV)*, 2019, pp. 5492–5501.

[119] S. Sudhakaran, S. Escalera, and O. Lanz, "LSTA: Long short-term attention for egocentric action recognition," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2019, pp. 9954–9963.

[120] R. Possas, S. Pinto Caceres, and F. Ramos, "Egocentric activity recognition on a budget," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2018, pp. 5967–5976.

[121] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 646–661.

[122] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "Skipnet: Learning dynamic routing in convolutional networks," in *European Conference on Computer Vision (ECCV)*, 2018, pp. 409–424.

[123] Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman, and R. Feris, "Blockdrop: Dynamic inference paths in residual networks," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2018, pp. 8817–8826.

[124] Y. Wang, F. Sun, D. Li, and A. Yao, "Resolution switchable networks for runtime efficient image recognition," *arXiv preprint arXiv:2007.09558*, 2020.

[125] B. Uzkent and S. Ermon, "Learning when and where to zoom with deep reinforcement learning," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2020, pp. 12 345–12 354.

[126] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," in *International Conference on Computer Vision (ICCV)*, 2019, pp. 6202–6211.

[127] F. Xiao, Y. J. Lee, K. Grauman, J. Malik, and C. Feichtenhofer, "Audiovisual slowfast networks for video recognition," *arXiv preprint arXiv:2001.08740*, 2020.

[128] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, "Sanity checks for saliency maps," in *Conference on Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[129] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?" explaining the predictions of any classifier," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016, pp. 1135–1144.

[130] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *International Conference on Computer Vision (ICCV)*, 2017, pp. 618–626.

[131] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 818–833.

[132] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008.

[133] H. Zhao, J. Jia, and V. Koltun, "Exploring self-attention for image recognition," in *Conference on Computer Vision and Pattern Recogition (CVPR)*, 2020, pp. 10 076–10 085.

[134] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens, "Stand-alone self-attention in vision models," in *Conference on Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019, pp. 68–80.

[135] G. Ren, T. Dai, P. Barmpoutis, and T. Stathaki, "Salient object detection combining a self-attention module and a feature pyramid network," *Electronics*, vol. 9, no. 10, p. 1702, 2020.

[136] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers*, 2003, pp. 1398–1402.

[137] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," 2015.

[138] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.

[139] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *International Conference on Learning representation (ICLR)*, 2017.

[140] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning representation (ICLR)*, 2015.