APPLICATION-DRIVEN METRICS OF ONLINE ACTION RECOGNITION

Patrícia de Andrade Kovaleski

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientadores: Eduardo Antônio Barros da Silva
Leonardo de Oliveira Nunes

Rio de Janeiro
Abril de 2020

APPLICATION-DRIVEN METRICS OF ONLINE ACTION RECOGNITION

Patrícia de Andrade Kovaleski

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Orientadores: Eduardo Antônio Barros da Silva
Leonardo de Oliveira Nunes

Aprovada por: Prof. Eduardo Antônio Barros da Silva
Eng. Leonardo de Oliveira Nunes
Prof. Lisandro Lovisolo
Prof. José Gabriel Rodríguez Carneiro Gomes

RIO DE JANEIRO, RJ – BRASIL
ABRIL DE 2020

Kovaleski, Patrícia de Andrade

Application-driven Metrics of Online Action Recognition/Patrícia de Andrade Kovaleski. – Rio de Janeiro: UFRJ/COPPE, 2020.

XII, 73 p.: il.; 29, 7cm.

Orientadores: Eduardo Antônio Barros da Silva
Leonardo de Oliveira Nunes

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2020.

Referências Bibliográficas: p. 64 – 73.

1. Deep learning. 2. Convolutional neural network. 3. Computer vision. 4. Action recognition. I. Silva, Eduardo Antônio Barros da *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

Agradecimentos

Agradeço primeiramente à minha família, que forneceu todo o apoio necessário para que eu chegasse até aqui. Agradeço aos meus pais pela paciência e carinho, à minha irmã pelo incentivo e exemplo e ao meu cunhado pelo amparo e momentos de descontração. Também sou grata aos meus amigos Carla, Victor Hugo, Roberto e Igor, que acompanharam e colaboraram imensamente nesta jornada. Sem eles, o caminho teria sido muito mais difícil.

Ao meu time do Laboratório de Tecnologia Avançada (ATL) da Microsoft, agradeço pela oportunidade, confiança e infraestrutura oferecidas. Os recursos computacionais disponibilizados foram fundamentais para o desenvolvimento desta pesquisa, assim como todas as discussões e conhecimentos compartilhados.

Por fim, agradeço aos meus orientadores Eduardo da Silva e Leonardo Nunes, pelo incentivo, confiança e disposição durante esses anos. Também sou grata aos demais professores e funcionários da Universidade Federal do Rio de Janeiro, em especial aos integrantes do Laboratório de Sinais, Multimídia e Telecomunicações, pelo trabalho e dedicação.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

MÉTRICAS ORIENTADAS A APLICAÇÕES DE RECONHECIMENTO DE AÇÕES ONLINE

Patrícia de Andrade Kovaleski

Abril/2020

Orientadores: Eduardo Antônio Barros da Silva
Leonardo de Oliveira Nunes

Programa: Engenharia Elétrica

Este trabalho busca entender a lacuna existente entre pesquisa e aplicações ao avaliar a performance de alguns métodos de classificação de ações em um cenário de streaming de vídeo, essencial a diversas aplicações. É apresentada uma visão geral sobre a área de reconhecimento de ações, suas principais tarefas, métodos e métricas. A partir da situação atual da área e tendo em vista que diversas aplicações de classificação de ações necessitam ser executadas sobre um streaming de vídeo, dois questionamentos são levantados: a) Como é o desempenho dos modelos de classificação de ações em um ambiente de streaming? b) Quais são as melhores métricas para se avaliar um ambiente de streaming? Buscando respondê-los, avaliamos a performance de alguns métodos de classificação de ações em um cenário de streaming de vídeo. As principais métricas e protocolos de avaliação utilizados na literatura foram aplicados e adaptados para o caso de streaming quando necessário. Os resultados obtidos não capturaram de forma satisfatória o desempenho dos métodos, apontando a necessidade de métricas novas e mais tangíveis. Portanto, novas formas de avaliação foram estudadas e propostas. Por fim, foi oferecida uma visão geral dos desafios enfrentados atualmente na área de reconhecimento de ações e algumas sugestões para aqueles que desejem utilizá-la em soluções atuais.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

APPLICATION-DRIVEN METRICS OF ONLINE ACTION RECOGNITION

Patrícia de Andrade Kovaleski

April/2020

Advisors: Eduardo Antônio Barros da Silva
Leonardo de Oliveira Nunes

Department: Electrical Engineering

This work tries to understand the gap between research and applications by evaluating the performance of some action classification methods in a video streaming scenario, which is essential for several applications. An overview of the action recognition area, its main tasks, methods and metrics are presented. Based on the current situation of the area and considering that several action classification applications need to run over a video streaming, two questions are raised: a) How is the performance of classification models in a streaming environment? b) What are the best metrics to evaluate a streaming environment? To answer them, we evaluated the performance of some action classification methods in a video streaming scenario. The main metrics and evaluation protocols used in the literature were applied and adapted to the case of streaming when necessary. The results obtained fail to satisfactorily capture the actual performance of the methods, pointing out the necessity of new and more tangible metrics. Therefore, new forms of assessment are studied and proposed. Finally, we provide an overview of the challenges currently faced in the area of action recognition and some insight for those who wish to use its methods in current solutions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computer Vision is an interdisciplinary field of science that aims to give computers the ability to understand videos and images as humans do. Identifying a dog in a photo, for example, can be easily performed by a child, since humans can interpret the colors and shapes present in the image naturally, translating them into information. However, this is an extremely complex task for a machine, that only receives a sequence of numbers whose meaning depends on the instructions to be executed next.

Some of the challenges that the field tries to solve are detection and classification of objects or faces in images [1–3], generation of three-dimensional information from two-dimensional data – as for 3D scene reconstruction from 2D views images [4–6] and 3D facial geometry reconstruction from 2D face images [7] – detection of moving objects in videos [8, 9], and image restoration [10, 11]. The use of such resources enables a series of applications whose relevance has increasingly attracted the interest of researchers and companies [12–16], leading to the belief that this will be a growing area in the coming years. Some of these applications involve, for example, quality control in manufacturing systems, event detection in videos for security systems, medical image analysis to assist in the diagnosis of patients and navigation of autonomous vehicles.

The history of Computer Vision starts in the 60s as a research interest of universities within the area of Artificial Intelligence. In the 70s there was some progress in the recognition of certain images [17, 18] as the understanding of the human vision system grew. Upon the effective arrival of neural networks in the 80s, a geometric approach with no connection to Artificial Intelligence and with an intense mathematical basis was taken, leading to a major advance in research. From that point on, Computer Vision was recognized as a separate field from Artificial Intelligence, gaining more prominence.

The advance of neural networks, especially deep learning approaches, allowed great progress in the field, making it surpass the human capacity in some tasks, as

labeling objects [15]. In the beginning, those approaches worked at limited capacity due to the high demand for data and computational power. Some of the key factors for the growth of Computer Vision are the enormous amount of data we generate today in videos and photos, and the development of improved hardware and algorithms. As the computational power required to analyze the vast amount of visual data increased and became accessible to a larger part of the academic community, it became possible to improve Computer Vision methods by training larger models.

To accompany the growth of the number of videos being recorded and shared currently, it is essential to efficiently understand and analyze video data, more specifically the human actions contained in it. Being one of the main challenges in Computer Vision, action recognition tries to identify the actions occurring in a video given a predetermined set of options. Figure 1.1 shows two examples of outputs from action recognition systems; in the first image, the actions are being individually detected and classified, while in the second the whole picture is being classified as an action. These tasks, despite being easily performed by a human, are quite complex for a machine. Among the various challenges, we can mention occlusions, viewpoint variation, camera movement, lighting changes and cluttered background.

Action recognition has attracted the interest of the industry and academic community due to its wide range of applications such as automated surveillance [20], video indexing [21], elderly care [22] and behavior analysis [23]. In recent years, there has been a considerable effort and progress in research in the area and several methods are being explored and developed to reach the performance needed for those practical applications. However, only a few of those methods consider the limitations and challenges that would be present when executing them in real situations, as in the applications before mentioned. For instance, many applications required real-time execution with low computational cost, causal algorithms for streaming processing and the ability to deal with low-quality and missing images.

In this work, we try to understand the performance gap between research and

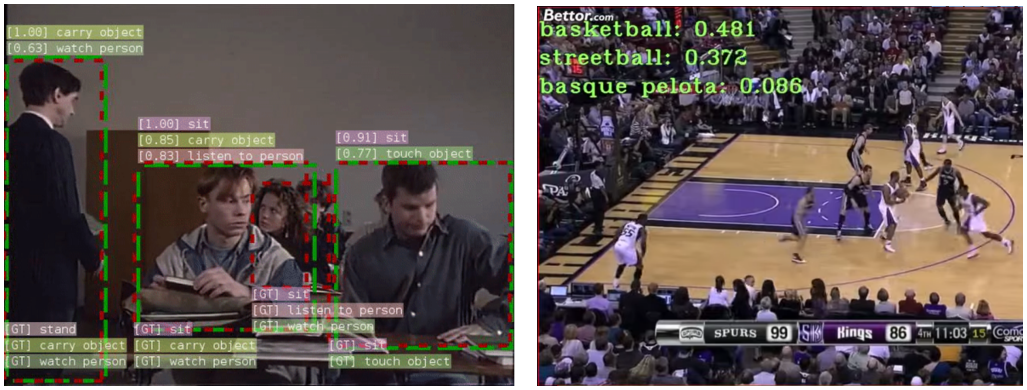


Figure 1.1: Output frames from action recognition systems that perform localization and classification of actions in video. *Source:* [12, 19]

practical applications. For that purpose, we evaluate the performance of some action classification methods in a video streaming scenario, which is essential for several applications. In this configuration, some metrics and protocols used in the literature fail to satisfactorily capture the actual performance of the proposed methods. Therefore, new forms of assessment are studied and proposed. We also provide an overview of the challenges currently faced in the area of action recognition and some insight for those who wish to use its methods in current solutions.

In Chapter 2 an overview of the action recognition field is presented, detailing the different tasks addressed and exploring the specific datasets and methods developed for each one. The main metrics applied in the area are presented in Chapter 3, together with the experiments performed to assess the selected action classification algorithm in a streaming scenario. New metrics and evaluation protocols to efficiently assess the algorithm in the streaming scenario are explored in Chapter 4. Finally, Chapter 5 concludes our work, discussing the results obtained and possible future work.

Chapter 2

Action Recognition

Action recognition has been a highly researched topic in the academic community in the last years [13, 24–26]. As our knowledge about this area grows, some more specific tasks within it are identified. These tasks mitigate the complexity of action recognition by dividing it in smaller and more defined problems. Nowadays, the main problems being tackled in this area are:

- **Action classification (Sec. 2.2.1):** consists in correctly classifying an action occurring in a video.
- **Temporal action localization (Sec. 2.2.2):** requires detecting the position of the action in time, aiming to identify when the action has started and when it has finished.
- **Spatio-temporal action localization (Sec. 2.2.3):** requires detecting the position of the action in space-time, aiming to identify when the action has started and when it has finished, and further detect its spatial position in the frame.
- **Online action detection (Sec. 2.2.4):** consists in detecting the start of an action in a video stream as soon as it happens.

Each of these problems requires different types of annotated data and approaches. The increasing interest in action recognition has led to the massive creation of new datasets and methods, making it difficult to keep up with the latest releases and to fairly compare them. Thus, in the following sections, a wide and updated review of the area is provided, comparing its main features, results and approaches. Considering our focus on action recognition applications, the online action detection problem seems to deal with the approaches most related to our goal. However, among the mentioned problems, the online action one is the least studied. In addition, most of the online action solutions are influenced by other, more well-established, classification and localization approaches. Therefore, for the sake of appropriately framing

the main subject of this thesis, it is important to give a broad overview of the action recognition literature.

In Section 2.1 some of the most utilized datasets in this field are introduced and compared. The datasets vary in annotation level, size, origin and type of action contained. Later, in Section 2.2, the main approaches for each of these problems are presented and discussed to better understand the evolution and the current state of action recognition research. Having in mind the requirement of most of action recognition applications for a causal solution, a special focus on causal methods is given. Finally, after reviewing the main datasets and approaches developed for each task, and considering our application-driven goal, we identify some concerns about the current online action recognition literature and define our next steps in Section 2.3.

2.1 Datasets

In this section, some of the main action recognition datasets are presented and compared. Each of the datasets has its own particularities and are more suitable for different types of tasks. To ease the comparison between them, Tables 2.1 and 2.2 expose some of their main characteristics. Additional information about these datasets are exposed in the following sections, dividing them by the task they focus on.

Table 2.1: Comparison of action recognition datasets. The label '$x+$' indicates that each video has at least x actions.

Dataset (year)	**Action classes**	**Total videos**	**Actions per video**	**Avg. dur.**	**Trimmed actions**
HMDB51 (2011)	51	7K	1	-	Yes
UCF101 (2012)	101	13K	1	7.2s	Yes
Sports-1M (2014)	487	1.1M	1	5.6min	-
Kinetics-400 (2017)	400	300K	1	10s	Yes
Kinetics-600 (2018)	600	500K	1	10s	Yes
Smth-Smth V1 (2017)	174	108K	1	4s	Yes
Smth-Smth V2 (2018)	174	220K	1	4s	Yes
THUMOS14 (2014)	20	18K	1.1	3.9min	No
ActivityNet v1.2 (2015)	100	9.6K	1.5	7.5min	No
ActivityNet v1.3 (2016)	200	20K	1.5	7.5min	No
Charades (2016)	157	10K	6.8	30.1s	No
TVSeries (2016)	30	27	1+	32min	No
MultiTHUMOS (2017)	65	413	10.5	3.9min	No
UCF-Sports (2008)	10	150	1	6.4s	Yes
J-HMDB (2013)	21	928	1	1s	Yes
AVA (2018)	80	437	2+	15min	No

Table 2.2: Comparison of the annotation types of action recognition datasets.

Dataset	**Origin**	**Type**	**Temporal annot.**	**Spatial annot.**
HMDB51	*YouTube*/Movies	Human actions	No	No
UCF101	*YouTube*	Sports	No[1]	No[1]
Sports-1M	*YouTube*	Sports	No	No
Kinetics	*YouTube*	Human actions	No	No
Something Something	Homes	Human-object interaction	No	No
ActivityNet	*YouTube*	Human actions	Yes	No
THUMOS14	*YouTube*	Sports	Yes	No
Charades	Homes	Daily actions	Yes	No
TVSeries	TV Series	Daily actions	Yes	No
MultiTHUMOS	*YouTube*	Sports	Yes	No
UCF-Sports	TV Sport channels	Sports	Yes	Yes
J-HMDB	*YouTube*/Movies	Human actions	Yes	Yes
AVA	*YouTube*	Atomic actions	Yes	Yes

2.1.1 Classification datasets

Datasets that focus in action classification contain videos trimmed around the action, having only one action per video, as exposed in Table 2.1. The annotation for this type of data consists in a class label for each video, since there is only one action occurring from the beginning to the end of it. In this section, the following datasets are described: HMDB51 [27], UCF101 [28], Sports1M [19], Kinetics [29] and Something-something [30].

Some of the first datasets to be used in this task were UCF101 [28] and HMDB51 [27]. Recently, datasets such as Sports1M [19], Something-something [30] and Kinetics [29] have focused on large-scale video classification, which has considerably improved the state of the art results in this task. Although UCF101 and HMDB51 are considered small for today's scenario, they are still widely used for comparison purposes.

HMDB51

The Human Motion DataBase (HMDB51) [27] contains 51 distinct action categories, each with at least 101 clips extracted from different sources, such as movies, public datasets and *YouTube*. In addition to action category labels, each clip was annotated with meta information to allow a more precise assessment of the limitation of current computer vision systems. The meta information contains the following fields: visible body parts, camera motion, camera view point, number of people involved in the

[1]There are spatio-temporal annotation for 24 classes.

action and clip quality.

UCF101

The UCF101 [28] consists of 101 different human actions, which are mostly outdoor sports, distributed over 13320 clips with 180 frames on average. This was one of the first datasets to have a large number of classes over a set of realistic videos, that usually contain camera movement, lighting changes and partial occlusion. Because of this, it has been widely used for validation in action recognition research, together with the HMDB51 dataset. Currently, there are datasets like Kinetics [29] and Sports-1M [19] that are larger and more challenging, however, UCF101 and HMDB51 remain very popular and are widely used to benchmark the performance of different action recognition algorithms [16, 31, 32].

Later, with the release of the THUMOS14 dataset [33], spatio-temporal annotation for 24 sports classes were also provided by JIANG *et al.* [33].

Sports-1M

Sports-1M [19] is a large-scale dataset for sport videos classification. It provides links to 1.133.158 *YouTube* videos annotated with 487 sports labels. Approximately 5% of the videos are annotated with more than one class. The annotations were generated automatically by analyzing the text metadata surrounding the videos, hence it is weakly annotated.

Kinetics

The Kinetics [29] dataset can be seen as the successor of the two standard benchmarks datasets for human action recognition, HMDB51 [27] and UCF101 [28]. It is also usually called "the ImageNet [34] of video recognition" since it provides the data needed to boost the performance of baseline methods as the ImageNet did for the image classification area. It is focused on human actions rather than activities or events, and each *YouTube* video generates a single clip, increasing the variability of the data. There are some cases in which more than one action is happening in a video, however, only one label is provided. For this reason, when evaluating classification performance in this dataset, a top-5 measure is more suitable than a top-1. An explanation about top-1 and top-5 metric is provided in Section 3.1.1.

Something-Something

The Something-Something dataset [30] is a large collection of densely-labeled video clips that show humans performing pre-defined basic actions with everyday objects. Many action datasets focus on high-level, human-centered activities, such as sports

or "getting out of a car". These actions do not encourage a network to learn about motion primitives that can encode object properties and intuitive physics. Thus, the goal of Something-something dataset is to provide fine-grained discrimination tasks, whose solution will require a deeper understanding of the physical world.

The first version of the dataset contains 174 actions labels in 108.499 videos with duration ranging from 2 to 6 seconds. Labels are textual descriptions based on templates, such as "Dropping [something] into [something]" or as specific as "Putting [something] that can't roll onto a slanted surface, so it stays where it is". The slots ("[something]") serve as placeholders for objects. In its second release, the number of videos increased to 220.847 and object classes annotations were provided in addition to the video label when applicable. For example, for a label like "Putting [something] onto [something]" there is also an annotated version like "Putting a cup onto a table".

2.1.2 Temporal localization datasets

For the temporal localization task, the datasets must provide annotation for the start and end time of each action of interest, like in ActivityNet [35], THUMOS14 [33], Charades [36], TVSeries [37] and MultiTHUMOS [38]. These datasets use long untrimmed videos, each containing multiple actions obtained either from YouTube, television series episodes or crowdsourced actors. Some of these datasets, like Charades and MultiTHUMOS, are more densely annotated than others, possessing a high number of actions per videos, as shown in Table 2.1.

ActivityNet

The ActivityNet [35] dataset is composed of daily life, high-level, goal-oriented activities from user-generated videos. The videos are untrimmed and temporally annotated. Its first version, ActivityNet v1.2, contains 9.682 videos in 100 classes, while its newer version, ActivityNet v1.3, which is a superset of v1.2, contains 19.994 videos in 200 classes. The videos are available on *YouTube*, most have a duration between five and ten minutes and contain 1.5 activity instances per video.

THUMOS14

The validation and testing sets of THUMOS14 [33] dataset contain, respectively, 1.010 and 1.574 temporally untrimmed videos with temporal annotations for 20 sports classes. A training set is not provided; instead, the UCF101 [28], a trimmed video dataset, is appointed as the official training set. A background set with over 2.500 relevant videos guaranteed to not include any instance of the 101 actions of UCF101 is provided.

Charades

The Charades [36] dataset aims to provide real and diverse examples of daily-life dynamic scenes. Diversity is ensured by distributing and crowdsourcing the whole process of video creation from script writing to video recording and annotation. The dataset shows activities of 267 people from three continents, and over 15% of the videos have more than one person. Each video is annotated by multiple free-text descriptions, action labels, action intervals and classes of interacted objects. In total, Charades provides 27.847 video descriptions, 66.500 temporally localized intervals for 157 action classes and 41.104 labels for 46 object classes.

TVSeries

The TVSeries dataset is introduced in [37] as a dataset more suitable for the task of online action detection. It is composed of 27 episodes of 6 popular TV series (*Breaking Bad, How I Met Your Mother, Mad Men, Modern Family, Sons of Anarchy* and *24*). There are 30 defined everyday actions and each of them occurs at least 50 times in the dataset. The number of episodes is selected in order to have around 150 minutes of every series, reaching almost 16 hours in total.

The dataset is temporally annotated at the frame level and the videos have metadata labels that give more information on how the action is performed and captured, e.g. whether the action instance is atypical compared to the rest of the action instances in the same class, occluded, or taken from an unusual viewpoint. Figure 2.1 shows examples of metadata annotations. Although being composed of professionally recorded videos, the dataset depicts realistic actions and has large variability, containing multiple actors, different viewpoints, heavy occlusions and cases where the recording only starts after the action has started, or cases when it ends too early.

Figure 2.1: Example frames for some of the metadata annotations. Classes are 'eat', 'smoke', 'stand up', 'drink', 'going up stairway', 'get out of car' and 'use computer'. *Source:* [37].

MultiTHUMOS

The MultiTHUMOS [38] dataset is an extension of the THUMOS14 dataset for dense, detailed, multi-label understanding of human actions in videos. The annotation of 413 videos of THUMOS dataset was extended from 20 action classes with an average of 1.1 actions per video to 65 action classes with an average of 10.5 distinct actions per video. This was done by densely annotating the actions that compose a more high-level action label, such as a sport activity. For instance, a basketball video would be composed of actions like "run", "jump" and "basketball dunk".

2.1.3 Spatio-temporal localization datasets

A few datasets focus on the spatio-temporal localization task, providing bounding boxes annotations at frame level and temporal localization boundaries. Providing this type of annotation is extremely expensive, thus spatio-temporal datasets are rarer and smaller than the previous ones. In this section, were considered UCF Sports [39], JHMDB [40] and AVA [41] datasets.

The UCF Sports and JHMDB datasets are small in size, number of actions and video duration, while the recently released AVA is a large-scale densely annotated dataset that contains long untrimmed videos.

UCF-Sports

The actions of the UCF-Sports [39] dataset were collected from diverse sports videos featured on television channels such as the BBC and ESPN. The selected video sequences are trimmed around the action and bounding boxes annotations are provided for all frames.

J-HMDB

The J-HMDB [40] dataset is a subset of the HMDB51 [27] dataset with additional human joint annotations. Since human silhouettes are annotated for all frames, it is possible to derive ground truth optical flow and segmentation.

AVA

The spatio-temporally localized Atomic Visual Actions (AVA) [41] densely annotates, in space and time, 80 atomic visual actions with multiple labels per person occurring frequently. There are multiple labels for the majority of person bounding boxes. All bounding boxes have one pose label, 28% of bounding boxes have at least 1 person-object interaction label, and 67% of them have at least 1 person-person interaction label.

Atomic action categories consist in more low-level movements, where the most natural descriptions are in terms of intentionality and goal-directed behavior. Forcing a fine time scale limit, the actions are very physical in nature and have clear visual signatures. With these requirements the actions became more defined, being easily localized by annotators with a precision of $\pm 0.5s$ (in contrast to the reported $\pm 1.5s$ precision of THUMOS14 dataset, for instance).

2.2 Literature review

Unlike the case of image analysis, the temporal component present in videos provides important additional information that considerably improves the action recognition task. For certain activities, data such as intensity, duration and direction can be decisive for the correct classification, allowing, for example, to distinguish a person opening a door from a person closing a door. However, the spatial component is also fundamental, especially in more complex actions such as "peeling an apple" or "applying makeup", in which the object being interacted with plays a prominent role. Thus, most current recognition methods make use of spatio-temporal information when classifying an action.

Research in video understanding was significantly driven by advances in image recognition methods, often adapting them to deal with video data. In the beginning, many approaches performed the extraction of hand-designed spatio-temporal data features, such as SIFT-3D (3-Dimensional Scale Invariant Feature Transform) [42], STIPs (Space-Time Interest Points) [43] or HOG3D (Histograms of Oriented 3D-Gradients) [44], that were encoded using representations like bag of visual words or their variants, in order to be used for classification [21, 45, 46]. Among these hand-crafted features, improved Dense Trajectories (iDT) [47] is widely considered the state of the art.

With the return of neural networks to the public interest, there were several attempts to develop a deep architecture for action classification. Some works used as input a stack of static frames, expecting the network to implicitly learn to identify spatio-temporal attributes [48–50]. Others explored the use of predefined spatial-temporal filters in their first layers [51].

In the last decade, Convolutional Neural Networks (CNNs) have achieved a remarkable success in image recognition tasks and attempts to expand them for video data have emerged [20, 52, 53]. KARPATHY *et al.* [52] have compared the use of various convolutional architectures for action recognition using a stack of consecutive RGB frames as input. Almost no difference was found between the results obtained for single or multiple frames input, indicating that the features learned did not capture the movements well.

Based on these findings, SIMONYAN and ZISSERMAN [53] have proposed a two-stream architecture in which spatial and temporal information are explored separately by two distinct convolutional networks that have their individual results combined at the end, as illustrated in Figure 2.2. The spatial convolutional network receives single video frames as input and performs action recognition based on still images, acting as an image classification network. The temporal convolutional network, on the other hand, receives as input multiple sequential frames with motion information. By selecting a sequential set of these frames, the motion along a small duration of the video is captured, making the identification of the action easier. The motion is described as the dense optical flow vectors [54] between two frames. This method became the state of the art at the time and many recent works [31, 32, 55, 56] are based on this architecture.

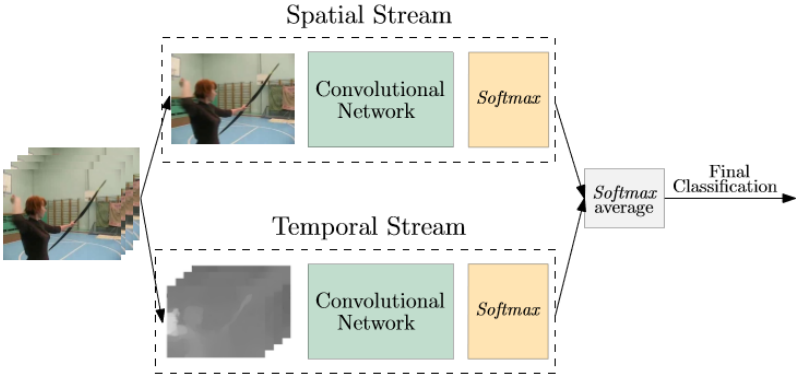


Figure 2.2: Two-Stream architecture for video classification.

Another important approach explored by many works [20, 57, 58] is the use of 3D convolutions to extract, simultaneously, features from the spatial and the temporal dimensions. Human actions in videos are three dimensional signals, thus, applying a 3D convolutional network seems like a natural and efficient evolution of the traditional 2D images approaches. However, by adding a dimension, the number of parameters of the 3D networks increases considerably, making them harder to train and computationally intensive. With the recent creation of large-scale labeled video datasets such as Kinetics [29], Charades [36] and Sports-1M [19], this limitation was partially removed, putting 3D networks back in the spotlights.

In the last years, most advances in the action recognition area were due to deep learning methods. Analyzing the main proposed approaches, it is possible to divide them into basically two categories: those based in a two-stream architecture, with 2D convolutional networks, and those using 3D convolutional networks. Although achieving similar results, these categories are structurally different, and there is no consensus in the community about the superiority of one over the other. Therefore, both architectures are being researched, exploring them individually or together in a mixed architecture.

Besides those three main architectures — two-stream, 3D convolutions and a combination of both — a fourth case that is worth mentioning consists of the approaches based on Recurrent Neural Networks (RNN) [59–61]. Usually, these types of methods combine convolutional networks to model frame-level spatial appearance with recurrent networks to model temporal dynamics. The most common recurrent unit is the Long Short-Term Memory (LSTM). A recurrent network composed of LSTM units is often called LSTM network or just LSTM.

2.2.1 Action Classification

Driven by the success of image classification methods, the action classification problem was the first target of video recognition research. It consists in correctly classifying, given a predetermined set of options, an action occurring in a video. The high performance of image classification networks makes it appealing to try to reuse them for video. Many approaches explore ways to evolve them to include a new dimension that carries the temporal information.

Two-stream methods

WANG *et al.* [55] have expanded the two-stream architecture to be used with very deep convolutional networks (e.g. VGGnet [62] and GoogLeNet [14]), achieving better results. Later, an architecture based in ResNets [15] with additional connections between streams is proposed by FEICHTENHOFER *et al.* [63], further enhancing the two-stream results. Another great improvement was introduced by WANG *et al.* [31], a novel framework for action recognition based on the idea of long-range temporal relation, called Temporal Segment Network (TSN). To capture this type of information, a TSN applies the two-stream for several distinct segments of the input video, averaging their results in the end. This idea was able to considerably boost the results of the classic two-stream method, becoming the new baseline architecture.

Other works have explored alternatives for the temporal stream input data since the computation of dense optical flow fields is impractical in real-time situations. In [31], the use of the difference of consecutive RGB frames as a replacement for the optical flow was proposed. The RGB difference between consecutive frames can be considered a noisy estimate of the flow, indicating the regions where significant change is happening. A comparison between these different types of input data is shown in Figure 2.3. ZHANG *et al.* [56] have introduced the use of enhanced motion vectors, that can be obtained directly without extra calculation. The enhanced motion vector network is the result of transferring the knowledge learned with an optical flow network (teacher) to a motion vector network (student) during training

and initialization. Both of these methods sacrifice some recognition performance in exchange of a huge reduction in the execution time, achieving over 20x of speed up.

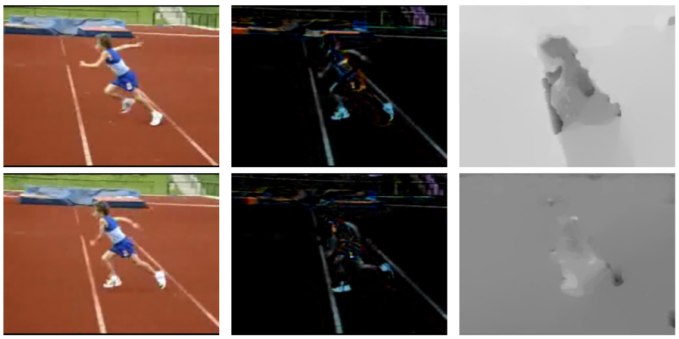


Figure 2.3: Comparison between the three different types of input data: RGB (left), RGB difference (center) and optical flow vectors (right). *Source:* [31].

A more recent approach, proposed by ZHU *et al.* [64], is to train a convolutional network to generate motion information from a set of consecutive frames. This network, called MotionNet, is concatenated with a temporal stream network that learns to project the motion information to action labels. This approach is end-to-end optimized, allowing the MotionNet to extract better motion representations than traditional optical flow for action recognition while still achieving over 5x of speed up. As a consequence, the accuracy drop seen in other approaches [31, 56] when replacing the temporal stream input data is avoided, since the motion estimation is not limited by the quality of the traditional optical flow.

WU *et al.* [25] have proposed to train a deep network directly on the compressed video. Since videos are usually provided compressed, decompressing it is an inconvenience. Moreover, videos have a high temporal redundancy that can be reduced by up to two orders of magnitude by compression. Using the TSN [31] architecture, a deep convolutional network processes the I-frames, which carry the RGB information, while a much smaller and simpler convolutional network processes the P-frames, which carry the motion information through motion vectors and residuals. This trade-off between speed and accuracy in the spatial and temporal streams is possible because most of the information is stored in I-frames and we only need to learn the update for P-frames. In the end, this method, called CoViAR, can be more practical, because it executes over the compressed data, is faster and achieves comparable results.

3D convolution methods

Believing that 3D convolutions are well-suited for spatio-temporal feature learning, TRAN *et al.* [57] have empirically tried to identify a good architecture for 3D networks. They found experimentally that $3 \times 3 \times 3$ convolution kernels for all layers

work best among the few explored architectures, thus defining a 3D network architecture called C3D. The C3D network is able to extract spatio-temporal features and achieves comparable results, but could not surpass those obtained by two-stream architectures. To improve the results, it was proposed to combine the C3D extracted features with the improved Dense Trajectories (iDT) [47] descriptor, achieving a higher result in video level (non-causal) classification. This shows that although some temporal information was modeled by the C3D, it is still non-optimal since the results were considerably improved when combined with iDT, a motion-based hand-crafted descriptor.

DIBA *et al.* [58] have introduced a new temporal layer, named Temporal Transition Layer (TTL), that consists of multiple 3D Convolution kernels with different temporal depths (see Fig. 2.4). This approach aims to capture short, mid, and long term dynamics, improving the capacity of temporal feature learning. Using the TTL, a video convolutional network, named Temporal 3D (T3D), is proposed. This network is an extended version of the DenseNet [65] with 3D filters and pooling kernels. An architecture to transfer learning from a pre-trained 2D network to a 3D network is also proposed, providing a stable weight initialization that avoids the computational cost of training from scratch.

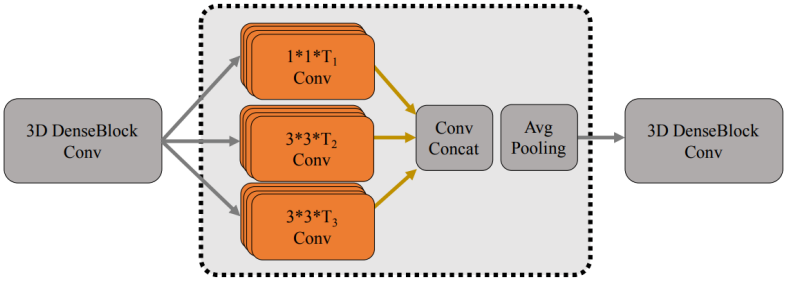


Figure 2.4: Illustration of the 3D Temporal Transition Layer (TTL). *Source:* [58].

Even with transfer learning, the high training complexity and the huge memory cost of 3D convolutions still hinder current 3D approaches. Moreover, most 3D architectures are composed of a stack of 3D convolutions, which are difficult to optimize and have the generation of deeper feature maps for high-level tasks limited by the high memory and computational costs. With these limitations in mind, ZHOU *et al.* [26] introduce 2D convolutions to 3D convolution modules and form a new 3D unit, called MiCT. An illustration of the proposed unit is shown in Figure 2.5. Since 2D convolutions can be constructed deeply, the MiCT unit empowers feature learning and reduces training complexity. By stacking MiCT units together the MiCT Network is created. This network surpasses the results obtained by methods with only RGB frames as inputs.

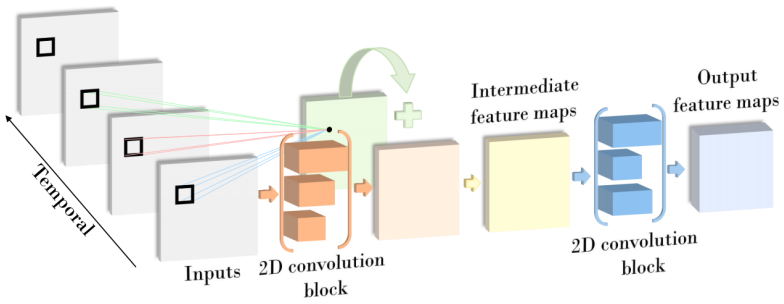


Figure 2.5: Illustration of a MiCT unit. Feature maps generated by the 3D convolutional module (green) are added to the ones produced by the residual 2D convolutional module (orange) on sampled 2D inputs. The combined feature maps are then fed into the concatenated 2D convolutional module (blue) to obtain the final feature maps. *Source:* [26].

Mixed methods

With the two-stream architecture as a base, FEICHTENHOFER *et al.* [32] have explored several ways to merge the results obtained by each stream. A new spatio-temporal architecture is proposed, with a novel convolutional fusion layer between the streams and a novel temporal fusion layer that incorporates 3D convolution and 3D pooling. These modifications highly improve the results of the traditional two-stream [53] without increasing the number of parameters significantly.

CARREIRA and ZISSERMAN [16] have introduced a new mixed architecture called Two-Stream Inflated 3D convolutional network (I3D), that is a base two-stream network with all of its filters and pooling kernels extended to 3D. A comparison is made between some of the main action recognition state of the art methods in the recently released Kinetics [29] dataset, which has two orders of magnitude more data than the previous benchmark datasets [27, 28]. Among the evaluated architectures are the two-stream with two different fusion methods [32, 53], the C3D network [57], a convolutional network with a recurrent layer (LSTM) on top [59] and the I3D network. A graphical visualization of the mentioned architectures is shown in Figure 2.6. The first observation is that pre-training on a larger video dataset, such as Kinetics, highly improves the results of all evaluated architectures, showing the benefit of transfer learning between video datasets. Secondly, the I3D network achieves the best results, becoming an important reference in the area.

Having the I3D [16] as the state of the art, XIE *et al.* [66] explore some variations of this architecture to achieve a more effective and efficient model. Given that 3D networks are much more expensive than 2D networks and tend to overfit, it is wondered if 3D convolutions are necessary in all of the network layers or if some of them can be replaced by 2D convolutions. The best result for speed and accuracy is achieved when replacing the 3D convolutions at the bottom of the network, forming

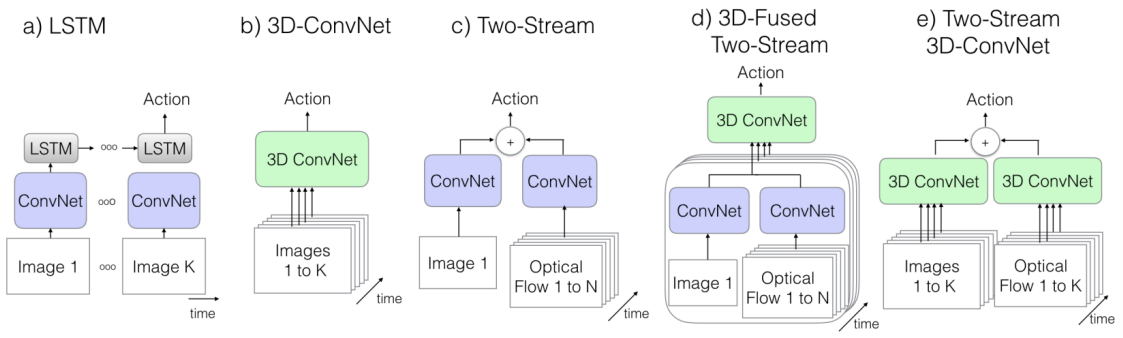


Figure 2.6: Video architectures evaluated in [16] *Source:* [16].

a "top-heavy" architecture, suggesting that learning high-level temporal features is more useful. This architecture also has a smaller number of parameters and is faster, since 3D convolutions are only applied in the top-layers. Another explored idea is to replace 3D convolutions by separable convolutions, in which the convolution is first done spatially — using 2D convolutions —, and then temporally — using 1D convolutions. A new model is then proposed, called Separable 3D network (S3D), that consists of an I3D with separable convolutions using the "top-heavy" architecture. The S3D network is slightly more accurate, has a smaller model size and achieves a large speed up in comparison with the I3D. Finally, they further improve the accuracy of S3D by using feature gating, that acts as a "self-attention" mechanism, upweighting dimensions considered important and downweighting the irrelevant ones. By applying the gating module in the S3D is obtained the S3D-G network. Note that each of these modifications can be applied to other architectures, possibly improving their performances on action recognition tasks.

Separable convolutions were also proposed in prior papers, such as [67, 68], who refer to it as Pseudo-3D (P3D) and R(2+1)D convolutions, respectively. Both approaches explore this idea having the 3D ResNet as baseline. QIU *et al.* [67] have investigated different ways to modify the basic residual unit to contain a 2D and a 1D convolution, inserting them in a cascade or parallel architecture. To enhance structural diversity, different versions of P3D blocks are combined to form the Pseudo-3D Residual Network (P3D ResNet). TRAN *et al.* [68] have conducted a similar study to the one performed by XIE *et al.* [66], evaluating the performance when some 3D convolutions are replaced by 2D convolutions in the 3D ResNet architecture. Differently from [66], no considerable gap was found between the results of those models whose first layers or last layers were replaced by 2D convolutions. The authors then propose the R(2+1)D convolution, which is closely related to P3D blocks [67]. However, the R(2+1)D network has a simpler and more homogeneous architecture than P3D ResNet. The main difference between these two works and the S3D is, besides the backbone model and datasets evaluated, the combination of a "top-heavy"

architecture and the feature gating proposed by [66].

When extracting spatio-temporal features using 3D convolutions, a strong assumption is made that the features at the same spatial location can be aggregated across the frames. This assumption is correct for the cases in which the action being performed does not move spatially, which is not true in most cases. Thinking of that, ZHAO *et al.* [69] have proposed the trajectory convolution, a new operation for integrating spatial-related features along the temporal dimension. Motion, as dense optical flow fields, is used to help define the operation trajectory. By replacing the 1D convolutions of the "R(2+1)D" network with trajectory convolutions, the TrajectoryNet is obtained. The results achieve improvements over the Separable-3D baseline.

Conceptually, applying simultaneously two methods for modeling the temporal information — the temporal stream with optical flow and 3D convolutions — as in I3D, is redundant. However, the fact that the best results are achieved by this architecture means that these methods are, in some way, complementary. In other words, neither of them is capable of properly modeling the temporal information by themselves, needing additional help. Most action classification methods can benefit from the combination of different temporal modeling approaches such as TSN [31], C3D [57] and iDT [47], which makes evident the lack of capability of these models to satisfactorily represent motion information. This fact reinforces the idea that simply evolving image classification methods for video data may not lead to the desired solution, but the development of specific action oriented methods is required.

As a new idea to video recognition, FEICHTENHOFER *et al.* [12] propose the SlowFast network, a two-stream related architecture that is composed of a Slow pathway, operating at lower frame rate, and a Fast pathway, operating at higher frame rate, illustrated in Figure 2.7. The Slow pathway operates at low temporal resolution because it is designed to capture semantic information that can be given by a few sparse frames. In contrast, the Fast pathway is responsible for capturing motion information without focusing on spatial details, operating at high temporal resolution, and has fewer channels. By reducing its channel capacity, the Fast pathway can be very lightweight, resulting in a cheaper architecture since the Slow pathway has fewer frames and the Fast pathway has fewer channels. This approach achieves state of the art accuracy for video action classification and for temporal localization when combined with detection modules [1].

Other methods

WANG *et al.* [70] have proposed a new family of building blocks that apply non-local operations [71] to capture long-range dependencies. Usually, long-range dependencies are captured by repeatedly applying operations that only process a local neigh-

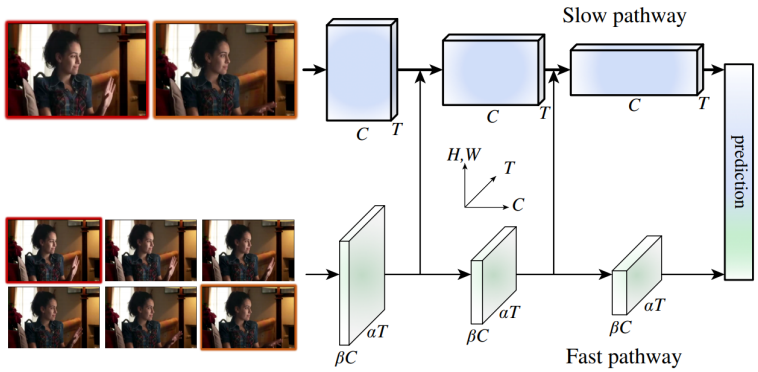


Figure 2.7: Illustration of the SlowFast network. Compared to the Slow pathway, the Fast pathway has temporal resolution $\alpha\times$ higher and only a fraction (β, e.g., 1/8) of channels. *Source:* [12].

borhood, such as convolutions and recurrences. In contrast, non-local operations capture long-range dependencies directly by computing interactions between any two positions, regardless of their positional distance in space or time. This method is more computationally efficient and can be easily combined with other operations. The addition of non-local blocks improved the performance of baselines in tasks of video classification, object detection and segmentation, and pose estimation. Based on the reported results, it is likely that non-local layers will be a common component of future architectures.

Finally, in Table 2.3 the results of the previously mentioned methods are compared.

2.2.2 Temporal Action Localization

Temporally localizing the occurrence of actions in long untrimmed videos is a highly challenging task whose results are still far below the desired to be used in practical applications. This task consists in detecting when an action has started and when it has finished. However, even for humans it is difficult to agree about the temporal boundaries of actions, as reported in [72, 73], indicating how ambiguous this task can be.

To better understand how far we are to solve this problem, ALWASSEL *et al.* [73] have investigated and classified the most relevant error types of temporal action localization. For this purpose, they have introduced a novel diagnostic tool, that analyzes the performance of temporal action detectors in videos. Among the errors categorized, it is shown that the localization error, in which a prediction with the correct label is done but fails to achieve the minimum intersection threshold (Section 3.1.3), has the most impact on the detector's performance. Another source of confusion for detectors is the quantity of context around actions, that can hinder

Table 2.3: Comparison of the reported accuracy results for the action classification approaches mentioned in Section 2.2.1.

	UCF101	**HMDB51**	**Kinetics-400 top1/top5**
Two-Stream [53]	87.0	59.4	-
Very deep two-stream [55]	91.4	-	-
ST-ResNet [63]	93.4	66.4	-
TSN w/ optical flow [31]	93.5	69.4	-
TSN w/ RGB difference	91.0	-	-
EMV [56]	86.4	-	-
MotionNet + I3D [64]	97.1	78.7	-
CoViAR [25]	90.8	60.4	-
C3D [57]	85.2	-	-
C3D + iDT	90.4	-	-
T3D [58]	93.2	63.5	61.2/-
MiCT [57]	88.9	63.8	-
MiCT + flow	94.7	70.5	-
Two-Stream Fusion [32]	92.5	65.4	-
Two-Stream Fusion + iDT	93.5	69.2	-
I3D w/o Kinetics [16]	93.4	66.4	-
I3D w/ Kinetics	**98.0**	**80.9**	74.2/91.3
P3D ResNet [67]	88.6	-	-
P3D ResNet + iDT	93.7	-	-
"R(2+1)D" [68]	97.3	78.7	75.4/91.9
S3D [66]	96.8	75.9	77.2/93.0
TrajectoryNet [69]	-	-	79.8 (avg.)
SlowFast [12]	-	-	**79.0/93.6**
Non-Local [70]	-	-	77.7/93.3

the precise boundaries positioning.

3D convolution methods

The 3D convolutional networks have promising results in video analysis tasks. They can learn spatio-temporal features directly from video frames but lose granularity in time, which is important for precise localization. However, for the temporal localization problem, the temporal length of the output should be the same as the input video while the spatial size should be reduced to 1×1. Therefore, to be able to upsample in time and downsample in space, SHOU *et al.* [74] propose a novel Convolutional-De-Convolutional (CDC) filter, which performs convolution in space (for semantic abstraction) and de-convolution in time (for frame-level resolution) simultaneously. A CDC network is then introduced by adding and replacing some layers of the C3D network [57] for CDC layers.

Inspired by the Faster R-CNN [1] object detection approach, XU *et al.* [75]

introduce the Region Convolutional 3D Network (R-C3D), which encodes the video streams using the C3D network, generates candidate temporal regions containing activities, and finally classifies selected regions into activities classes. The 3D feature maps are shared between the proposal and classification pipeline to save computation time and jointly optimize features for both tasks. As a result, the R-C3D can achieve an execution speed of over 500 fps (frames per second) in GPU. Anchor segments are utilized by the temporal proposal sub-network to predict variable length proposals. The anchors are pre-defined multi-scale windows centered at uniformly distributed temporal locations of the input and serve as reference segments for proposals at each temporal location. A buffer of 768 frames (approximately 30 seconds of video) is utilized as input to R-C3D to cover the length of most activity segments of the evaluated dataset (THUMOS14 [33]).

Mixed methods

CHAO *et al.* [76] have proposed the Temporal Action Localization network (TAL-Net), also inspired by the Faster R-CNN [1] object detection framework. The main limitation of approaches that followed the original Faster R-CNN, as the R-C3D [75], is that their anchor classifiers at each location share the same receptive field. Since the duration of actions can vary from less than seconds to minutes, a fixed receptive field can be too small and not contain all the needed information, or too large, and be dominated by irrelevant information. TAL-Net improves receptive field alignment using a multi-scale architecture in which each anchor scale has an associated network with aligned receptive field. To design temporal networks with a controllable receptive field size the use of dilated temporal convolutions is proposed. A dilated temporal convolution subsamples pixels in the input feature map instead of taking adjacent ones when multiplied with a convolution kernel. TAL-Net also exploits the temporal context of actions by extending the receptive field to cover the contexts regions before and after the action. Moreover, a late fusion scheme is applied for proposal generation and action classification stage.

Other methods

While observing a video, as a detector sees more of one activity, it should become more confident of the presence of the correct action category, improving its accuracy. With this intuition, MA *et al.* [60] introduce two novel ranking losses to be applied in a recurrent neural network (RNN) learning process so that the trained model better captures the progression of activities. As the detector sees more of the activity, the first ranking loss focuses on producing monotonically non-decreasing detection scores for the correct activity category, while the second one focuses on discriminating

between the correct and incorrect categories. The proposed model consists of a convolutional network, that extracts visual features from each video frame, followed by an LSTM, that computes activity detection scores based on the features of the current frame and the hidden states and memory of the LSTM from the previous time step. This solution can also be applied in the problem of online action detection, which is discussed in Section 2.2.4, given its causality and incremental detection.

YEUNG *et al.* [38], besides introducing the MultiTHUMOS dataset, have proposed the MultiLSTM model, a variant of LSTM networks for modeling the dense temporal relations of MultiTHUMOS. The MultiLSTM expands the temporal receptive field of both input and output connections of an LSTM. The extended input provides a direct pathway to previously seen frames, allowing the LSTM to spend its modeling capacity on more complex and longer-term interactions instead of maintaining a summary of the recent frames. A soft-attention mechanism [77] is applied to compute the weighted combination of the input frames. The multiple outputs allow the model to refine its past predictions after seeing more frames. Analogously, the predictions are consolidated with a weighted average.

Table 2.4 summarizes the mAP results of the mentioned methods for temporal action localization at an IoU of 0.5. The definitions of the applied metrics are provided in Section 3.1.3.

Table 2.4: Comparison of the reported mAP results for the temporal action localization approaches mentioned in Section 2.2.2.

	THUMOS14	**ActivityNet v1.3**	**MultiTHUMOS**
LSTM w/o losses	-	31.3	-
LSTM w/ losses [60]	-	36.4	-
CDC [74]	23.3	**45.3**	-
MultiLSTM [38]	41.3	-	**29.7**
R-C3D [75]	28.9	26.8	-
Tal-Net [76]	**42.8**	38.2	-

2.2.3 Spatio-Temporal Action Localization

The spatio-temporal localization problem consists in detecting where, in space and time, an action has occurred. It is the combination of the temporal localization problem mentioned in Section 2.2.2 and the traditional spatial detection problem addressed by image methods. Therefore, an additional challenge in spatio-temporal localization is to accommodate the uncertainty of per-frame spatial localization and temporal consistency.

Two-stream methods

A straightforward approach for spatio-temporal localization is proposed by GKIOXARI and MALIK [78], generating region proposals, submitting them to a two-stream network and applying a linear classifier to decide for a bounding box and an action label. A selective search [79] is applied in each RGB frame to generate region proposals, and using the optical flow data, proposals without motion are discarded. The spatio-temporal features for each bounding box, extracted through the two-stream network, are classified using a linear SVM. The selected bounding boxes for each frame are then linked using the Viterbi algorithm, forming an action tube.

PENG and SCHMID [80] have introduced the multi-region two-stream R-CNN model, an approach to spatio-temporal localization that applies a modified two-stream Faster R-CNN network [1] architecture that evaluates multiple regions of each proposal. Other proposal methods — selective search [79] and EdgeBoxes [81] — were also considered, but the Faster R-CNN achieved better results than the others with higher intersection-over-union (IoU) score (Section 3.1.3). To improve the classification scores, 4 types of regions relevant for action representation are selected from each proposal: the original region, the upper part of the region, the bottom part of the region and the border around the region. Temporal linking is done similarly to [78].

Most spatio-temporal localization approaches apply a detector on the video frames independently, without exploiting their temporal continuity. To surpass this limitation and treat a video as a sequence of frames, KALOGEITON *et al.* [82] have proposed the Action Tubelet detector (ACT-detector), that takes as input a short sequence of a fixed number of frames and outputs tubelets (action tubes slices). The ACT-detector is illustrated in Figure 2.8. The features extracted by the network for each frame in a sequence of K frames are stacked together. The stacked features are the input of two convolutional layers, one for scoring action classes and one for regressing the anchor cuboids of the Single Shot MultiBox Detector (SSD) [83].

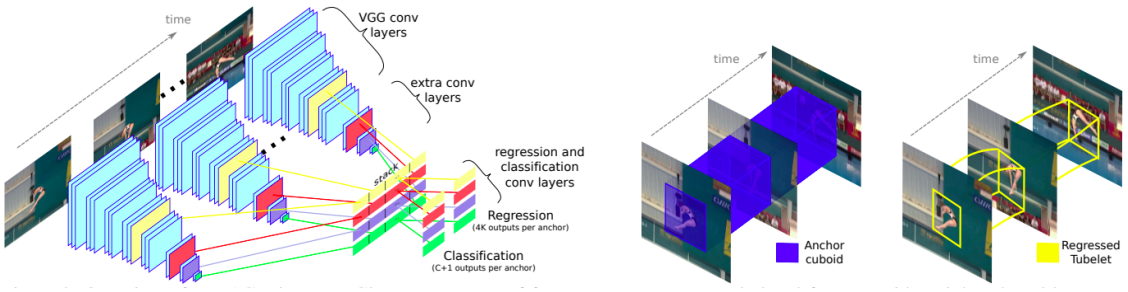


Figure 2.8: Overview of the ACT-detector. *Source:* [82].

Mixed methods

Following [82], GU *et al.* [41] have proposed a new approach that combines Faster R-CNN with an I3D network, in an end-to-end localization and classification method. The I3D network was chosen in order to better model temporal context, which is kept throughout this network. This approach achieves state of the art on spatio-temporal benchmarks. Furthermore, a new spatio-temporal dataset is introduced, AVA, which is considerably more challenging than previous ones, as discussed in Section 2.1.3.

The results of the mentioned methods are listed in Table 2.5 for an easier comparison. There are two main metrics widely used in this task, the frame-mAP and the video-mAP. The frame-mAP consists in the average precision of detections at the frame level while the video-mAP is the average precision of detection at video level. For both metrics, the reported results are at an IoU threshold of 0.5. A more detailed explanation about these metrics is presented in Section 3.1. Note that an extra result is reported, the spatial-temporal adaptation of the S3D classification network [66].

Table 2.5: Comparison of the reported results for the spatio-temporal action localization approaches mentioned in Section 2.2.3.

		UCF-Sports	**J-HMDB**	**AVA**
frame-mAP	GKIOXARI and MALIK [78]	68.1	36.2	-
	PENG and SCHMID [80]	84.5	58.5	-
	KALOGEITON *et al.* [82]	**87.7**	65.7	-
	GU *et al.* [41]	-	73.3	**15.8**
	Faster RCNN + S3D-G [66]	-	**75.2**	-
video-mAP	GKIOXARI and MALIK [78]	75.8	53.3	-
	PENG and SCHMID [80]	**94.7**	73.1	-
	KALOGEITON *et al.* [82]	92.7	73.7	-
	GU *et al.* [41]	-	**78.6**	-

2.2.4 Online Action Detection

The online action detection problem consists in detecting the start of an action in a video stream as soon as it happens. This is a highly important task, especially due to its crucial need for security related applications, such as automated surveillance or autonomous car systems, that require early alert generation in which a response must be immediately returned. However, it is also very challenging since only a small part of the action can be observed, the start of an action has an ambiguous boundary — as commented in Section 2.2.2 — and the positive and negative samples are highly unbalanced. In the online action detection task, one can focus on classification,

temporal localization or spatio-temporal localization.

Many approaches [61, 84] focus on improving a more specific task: given that a single action has started, classify it as soon as possible. This task is usually called early action detection or action anticipation and is evaluated on classification datasets, which have a single trimmed action per video, as discussed in Section 2.1. Since the early detection of an action is intrinsic to the online action detection task, those methods will be presented and discussed in this section.

Aware of the challenges that involve online action detection, GEEST *et al.* [37] introduce a new dataset, TVSeries, focusing on this task, as discussed in Section 2.1.2. An evaluation protocol is also proposed, allowing the comparison of different solutions qualitatively and quantitatively based on their per-frame classification. This protocol is defined and explained in Section 3.1.4. Some baselines were evaluated in this new dataset and it is shown that none of them perform well.

Two-stream methods

SADEGH ALIAKBARIAN *et al.* [61] aim to identify the action with a high prediction accuracy even in the presence of a very small percentage of a video sequence. The proposed approach employs a two-stream inspired feature extractor in a multi-stage LSTM architecture and introduces a novel loss function that encourages the model to predict the correct class as early as possible. The first part of the feature extractor network is shared between the two streams up to a certain layer. The output of this layer is connected to two sub-models, one for context-aware features — that is equivalent to the last part of the feature extractor network — and the other for action-aware ones — that applies the idea of Class Activation Maps (CAMs) [85] to extract features that focus on the action itself. The multi-stage LSTM model first focuses on the context-aware features, which encode global information about the

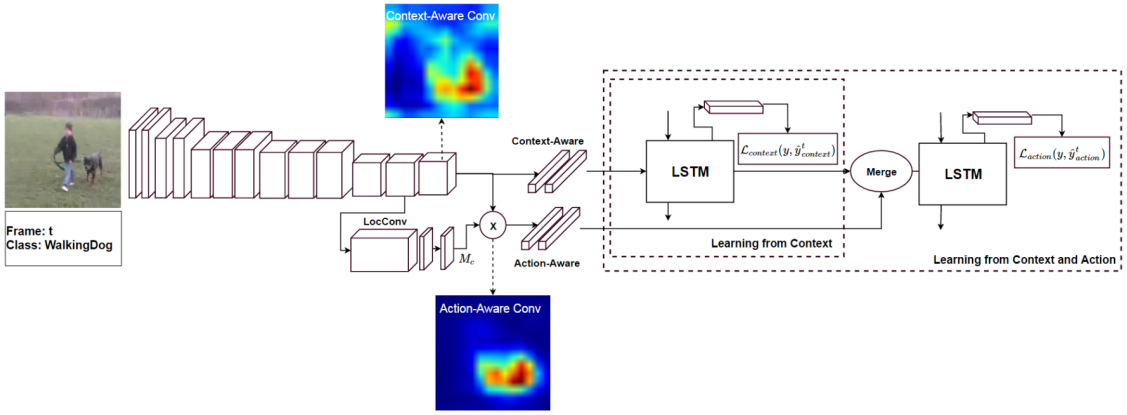


Figure 2.9: Overview of the multi-stage LSTM approach proposed by SADEGH ALIAKBARIAN *et al.* [61]. *Source:* [61].

entire image. It then combines the output of this first stage with the action-aware features to provide a refined class prediction. An overview of this approach is shown in Figure 2.9. As for the loss function, it consists of two terms. The first one penalizes false negatives with the same strength at any point in time, while the second one penalizes false positives with its strength increasing linearly over time until it reaches the same weight as the first one. The second term is based on the intuition that some actions can be highly ambiguous in the first few frames, and therefore false positives should not be penalized too strongly in the early stages. The reported results for this method significantly surpass other baselines in the evaluation performed.

A very challenging task is explored by SINGH *et al.* [84], that aims to perform spatio-temporal localization of actions in a real-time online scenario. The proposed framework starts by predicting detection boxes and the associated action class-specific confidence scores, in a single-stage using the Single Shot MultiBox Detector (SSD) [83]. The prediction is done for RGB and flow frames, following a two-stream based architecture, that are then combined. Finally, multiple action tubes are generated incrementally at frame level with the help of a novel greedy algorithm. Figure 2.10 illustrates the proposed framework. Note that one must choose between Figure 2.10(b) and Figure 2.10(c) and thus determine whether the framework will be less accurate but real-time, or more accurate but not real-time. It is difficult to perform a fair comparison with the most similar approaches as they are still considerably different or do not present results for the same datasets. However, it is safe to say that the proposed method achieved comparable, and in some cases superior, results with a significant drop in the execution time.

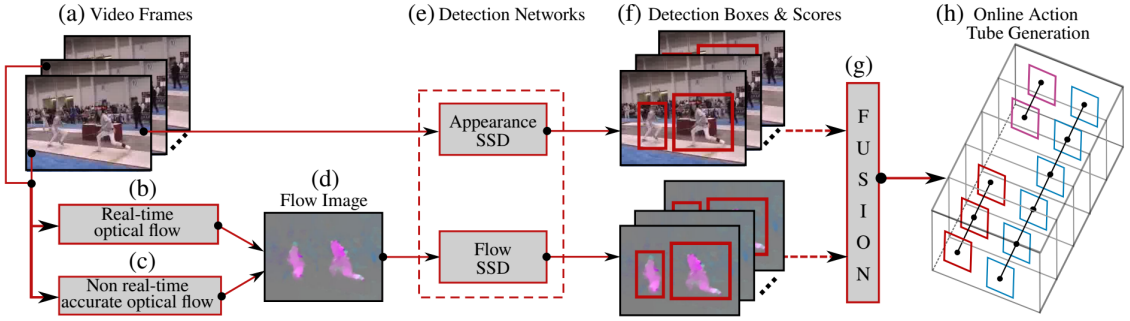


Figure 2.10: Online spatio-temporal action localization framework proposed by SINGH *et al.* [84]. *Source:* [84].

3D convolution methods

SHOU *et al.* [86] have identified three challenges and propose three respective novel solutions for training an online detector of action start (ODAS) for frame-level clas-

sification. The first challenge is how to correctly distinguish the beginning of the action from the background that precedes it, since they can be very similar. To improve the ability to differentiate them, an auxiliary Generative Adversarial Network (GAN) [87] is introduced to automatically generate hard negative samples during training. For the second challenge, a start window and the follow-up window are defined, as shown in Figure 2.11b. Following the state of the art video classification models such as C3D [57], TSN [31] and I3D [16], a short temporal sliding window is accepted as input. In this scenario, there will be a moment in which the start window contains frames from the background and the action, while the follow-up window is completely inside the action. To avoid errors caused by the background frames, it is proposed to model the temporal consistency between the start window and its follow-up window during training. Finally, the third challenge refers to how scarce are the training samples of start windows compared to background windows and windows fully inside the action. To tackle this issue, an adaptive sampling strategy is designed to increase the percentage of start windows during training.

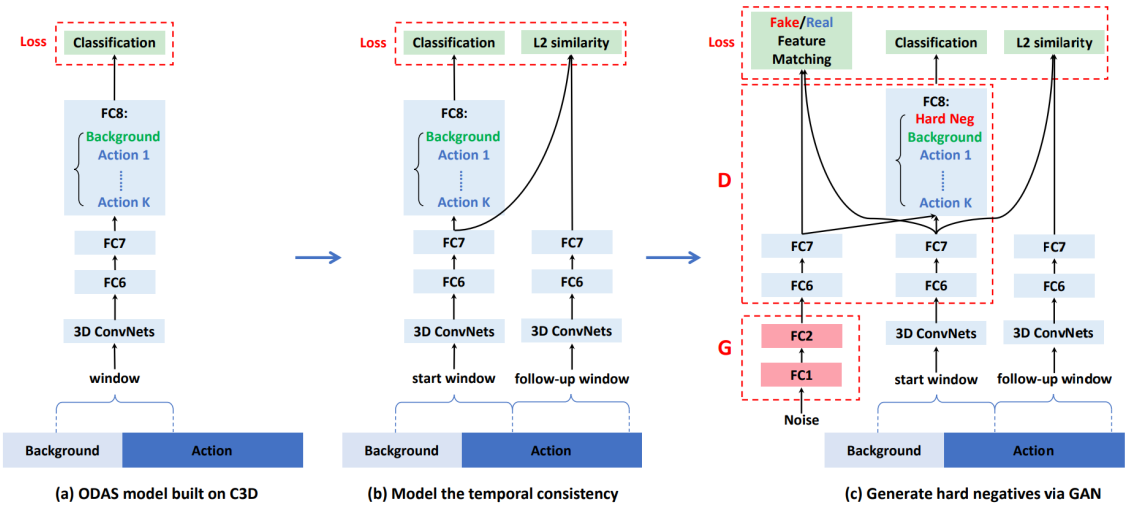


Figure 2.11: The network architectures of ODAS models built on C3D [57] and the proposed training objectives. *Source:* [86].

Besides all the effort of [86], the presented results are still far from acceptable, being even compared with random guess results. However, under the novel evaluation protocol also proposed in [86], ODAS outperforms CDC [74], a per-frame spatio-temporal method, and RED [88], a per-frame action anticipation method. The necessity of a novel evaluation protocol is justified as ODAS focuses on detecting the action start window without labeling per-frame the entire action, as considered in [37, 88]. The proposed protocol is defined and explained in Section 3.1.6.

Other methods

Action prediction, or action anticipation, aims to detect an action before it happens, and when the anticipation time is 0 it can be viewed as an online action detection problem. GAO *et al.* [88] have proposed a Reinforced Encoder-Decoder (RED) network for action anticipation. The RED architecture is composed of a video representation extractor, an encoder-decoder network and a classification network. Small chunks (6 frames) of the video are processed by a feature extractor — as a two-stream or a VGG16 [62] model — generating visual representations. Then, the encoder-decoder network takes continuous steps of visual representations as input and outputs a sequence of anticipated future representations that will be processed by a classification network. The encoder-decoder network uses an LSTM network as basic cell and reinforcement learning is used to train it on sequence level encouraging the system to make correct predictions as early as possible. State of the art results are achieved when compared to previous encoder-decoder methods [89] that do not apply reinforcement learning.

Each of the above mentioned methods performs different online detection tasks, with different metrics and datasets. Thus, it is not possible to provide a direct comparison between the reported results. SADEGH ALIAKBARIAN *et al.* [61] report a classification accuracy of 55% when observing the first 20% frames on J-HMDB dataset, while SINGH *et al.* [84] report 71.1 mAP for a threshold of 0.5 IoU. SHOU *et al.* [86] report 8.5 mAP on ActivityNet dataset for an offset threshold of 10 seconds. With an anticipation time of 1 second, GAO *et al.* [88] achieve 75.5 cAP (calibrated Average Precision, proposed by [37]) on TVSeries dataset and 37.5 per-frame mAP on THUMOS14 dataset.

2.3 Conclusion and next steps

In this chapter, an overview of the evolution and current state of the action recognition field was presented. The different tasks addressed in this area were detailed, exploring the specific datasets and methods developed for each one. There is much to be improved in each task, but to assess them and compare their results a set of metrics and protocols is required. However, given the variety of sub-tasks performed in online action detection, there is no consensus on the standard metrics to be applied, as discussed at the end of Section 2.2.4. This situation creates uncertainty about the actual performance of the methods evaluated.

Many applications of action recognition require a real-time, continuous execution, in which the model would probably run over a streaming of video. This setting – real-time and streaming execution – is part of the online scenario and current models

are still far from achieving the performance needed for those applications. These are important topics to be researched and developed, but to make progress, we need to be able to evaluate those models effectively and have a good measure of their performance to understand which points need to be improved.

Most online classification approaches do not achieve good results in reasonably easy datasets [61, 86], while classification models are achieving good results in very challenging datasets [12, 70]. It is rare to see a comparison between online and classification approaches in an online setting. This raises a question about what would be the performance of a typical classification model when submitted to an online scenario. Therefore, in this work, we try to understand how a good classification model performs in a desirable online scenario: video streaming.

There are two main points we want to study in this work:

- **How is the performance of classification models in a streaming environment?**

  Classification models achieve good results according to their own metrics. However, many action classification applications require a streaming environment, which is not contemplated by those metrics.

- **What are the best metrics to evaluate a streaming environment?**

  There are a variety of metrics being applied to the different sub-tasks addressed in online action recognition, but their capability to represent a model's performance in a streaming scenario is uncertain, as most of them do not consider the requirements of this case.

Chapter 3

Assessment of current online action recognition methods

In this chapter, we will study the performance of a state of the art classification model in a streaming environment. The performance assessments will be based on the usual metrics applied in the area of action recognition, adapting them, when required, to their streaming "per-frame" version.

In Section 3.1, the main metrics applied in the action recognition literature will be presented and detailed. Later, in Section 3.2, the topics related to the implementation and training of the neural network models will be discussed. The proposed experiments will be explained and their results exposed and discussed in Section 3.3.

3.1 Conventional metrics and protocols

In this section we will present six different action recognition metrics:

- **Top-N accuracy (Sec. 3.1.1):** applied in classification tasks in which only one action occurs throughout the video (the video is trimmed around the action);
- **Mean average precision (mAP) (Sec. 3.1.2):** used on tasks whose samples may have multiple annotations, as multi-label classification and spatial or temporal localization;
- **Intersection over union (IoU) (Sec. 3.1.3):** measures the accuracy of detection algorithms. In action recognition, it is applied in the bounding boxes of spatial localization and in the time intervals of temporal localization;
- **Calibrated average precision (cAP) (Sec. 3.1.4):** weighted per-frame

version of the mean average precision (mAP) used in online action classification tasks;

- **Percentage of frames seen (Sec. 3.1.5):** early action classification metric applied on datasets that have trimmed actions;
- **Point-level action start detection mAP (Sec. 3.1.6):** action start classification measure developed for a specific online action framework (ODAS [86]).

3.1.1 Top-N accuracy

When dealing with classification methods, accuracy is the most common metric applied. It can be defined as the ratio between the number of correct predictions over the total number of samples, as shown in Equation 3.1. Classification accuracy is a good metric only when there is a comparable number of samples in each class, or it can give us the wrong idea of the true state of the model.

$$\text{accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of samples}} \tag{3.1}$$

If we train a model on a dataset with 99% samples of class A and 1% samples of class B, it would easily achieve 99% training accuracy by predicting all samples to class A. But, when testing this model on a test set with 50% samples of class A and 50% samples of class B, the accuracy would drop to 50%. Thus, the main concern of classification datasets is to ensure, as much as possible, an equal division of samples in each class for training and testing sets.

A common practice to deal with an unbalanced dataset is to calculate the accuracy for each class and then average all per-class results to obtain the final accuracy for the model. The per-class accuracy is calculated as the ratio between all correct predictions for that class and all samples annotated as true for that specific class. In the example presented in the last paragraph, when testing a model that always predicts class A on a test set with 99% samples of class A and 1% samples of class B, the mean accuracy per-class would be 50% while the accuracy would be 99%. Accuracy can be seen as the mean per-class accuracy in which the result for each class is weighted by the number of their samples. If the dataset is reasonably balanced, the accuracy is similar to the mean per-class accuracy.

Usually, the result of a classification model can be interpreted as the confidence or the probability that the input sample belongs to each of the classes. This interpretation is possible given the use of an operation that normalizes the output vector of real numbers into a probability distribution. The most common function used for this purpose is softmax, in which the probability for a class i, given an input x is

obtained as:

$$p(\text{class} = i|x) = \frac{e^{y_i}}{\sum_{j=1}^{K} e^{y_j}}, \tag{3.2}$$

where K is the total number of classes. The class with the highest value is probably the best prediction for that sample and choosing the maximum value of your model's predictions is called the **top-1 accuracy**.

In many cases, the top-1 accuracy is not enough to show the model's performance, for instance, when there is more than one class present in an input sample but only one is annotated. Ideally, the model would predict with high confidence all of the classes, but since only one of them is considered truth, we could get a "wrong" prediction if the annotated one does not have the highest value. To avoid this problem, a very usual metric for these cases is the **top-N accuracy**, which measures if the true value of the sample is one of the top N predictions of the model.

3.1.2 Mean average precision

When dealing with a multi-class classification problem we cannot simply choose the class with the highest prediction, as in the top-1 accuracy metric, since multiple classes can be present at the same time. One solution is to use confidence thresholds that decide whether or not each class is active. But choosing the right threshold value is difficult and normally there will be a different value for each class. To deal with these cases, the mean average precision (mAP) metric is applied.

To define mAP some other metrics must be explained first: precision, recall and average precision. In a set of samples that can be either positive (P) or negative (N), there are four types of predictions that a model can make:

- **True positive (TP)**: A sample whose true value is 'positive' and is correctly predicted by the model as 'positive'
- **False positive (FP)**: A sample whose true value is 'negative' and is wrongly predicted by the model as 'positive'
- **True negative (TN)**: A sample whose true value is 'negative' and is correctly predicted by the model as 'negative'
- **False negative (FN)**: A sample whose true value is 'positive' and is wrongly predicted by the model as 'negative'

With those terms, we can define precision and recall as:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.3}$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{TP}{P} \tag{3.4}$$

Precision measures how many of the samples the model said were positive, were annotated as positive. This gives us an idea of how much we can trust our model predictions. Recall measures how many of the positive samples we were able to identify, showing how well the model can find all positive samples. Depending on the problem, one can prioritize precision over recall or the opposite. For instance, when classifying tumor images as benign or malignant, it is better to give a healthy patient false positives than missing a true case and not giving a sick patient the needed treatment. The ideal scenario would be to have both, high precision and recall, but, in most cases, one must do a trade-off between them by selecting the most suitable threshold.

If we set a very low threshold, many predictions will be considered positive, even those with low confidence values. This would result in high recall - as most true positives would likely have high confidence values and be included - and low precision - because it would also include many false positives. In the opposite scenario, setting a very high threshold would select only a few samples as positives: those with the highest confidences. Thus, the few samples selected are likely to be true positives, resulting in high precision and low recall, since many other positive samples would not achieve the threshold and be classified as negatives.

The trade-off between precision and recall can be visualized by plotting a precision-recall (PR) curve. The curve is obtained by varying the thresholds and calculating the precision and recall at each setting. The area under the precision-recall curve is called average precision (AP) and there is an AP value for each class. Finally, the mean average precision is calculated as the mean of the AP of each class, as shown in Equation 3.5:

$$\text{mAP} = \frac{\sum_{c=1}^{\text{N}} \text{AP}(c)}{\text{N}} \tag{3.5}$$

where N is the number of classes and AP(c) is the average precision for a class c.

3.1.3 Intersection over Union

The intersection over union (IoU) metric attempts to measure the ratio between the area of the intersection of the true value with the predicted one and area of their union. This metric is widely used in spatial localization tasks, in which the goal is to correctly select an area of an image, usually represented by a bounding box. Image 3.1 shows a visual representation of how the IoU is calculated for a bounding box case. The predicted area for a sample is considered correct if its IoU value exceeds a defined threshold. Common threshold values vary from 0.3 to 0.7, with 0.5 being the most used value for comparison.

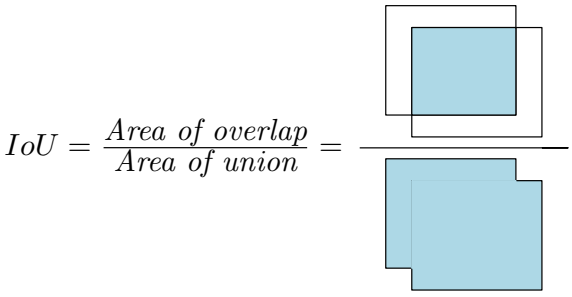


Figure 3.1: Visual representation of the intersection over union (IoU) metric.

In action recognition, the IoU metric is applied in the tasks of temporal localization and spatio-temporal localization, as exposed in Section 2.2. To assert the temporal localization of an action, the true and the predicted time intervals are compared. Thus, the IoU measures the ratio between the intersection and the union of two time intervals.

3.1.4 Calibrated average precision

Calibrated average precision (cAP) is a novel metric introduced by GEEST *et al.* [37] to measure the performance in online recognition tasks, in which a decision must be made at every frame. This metric is inspired by the work of HOIEM *et al.* [90] and can be seen as a per-frame normalized version of the mAP.

A logical idea to deal with online tasks would be to calculate the mAP using the predictions made at every frame, however, this metric is sensitive to the ratio between positive and negatives frames (frames for which the action is occurring or not), as discussed by JENI *et al.* [91]. For each class, the positive frames are the ones for which the action is occurring, while the negative frames are the ones for which the action is not occurring. A video with more negative frames (background data) would tend to have a higher probability to have false positives. This makes the comparison between classes with different positive vs. negative ratios difficult. To enable a fair comparison, GEEST *et al.* introduced the calibrated precision as:

$$\text{cPrec} = \frac{TP}{TP + \frac{FP}{w}} = \frac{w \times TP}{w \times TP + FP} \tag{3.6}$$

where w is the ratio between negative frames and positive frames. The cAP is then calculated similarly to the AP, with the calibrated precision replacing the traditional precision.

3.1.5 Percentage of frames seen

A very common metric when working with early action detection is to calculate the accuracy at different time points of the action. As discussed in Section 2.2.4, in

this task there is only one trimmed action per video and the aim is to classify it as soon as possible. The action will be evaluated at different time points, which can be determined in seconds or percentage of frames passed. There will be an accuracy value for each time point evaluated and the percentage of the frames considered at each point is incremental. Measuring the accuracy for 50% of the frames means that only the first half of the action will be considered, while 100% means that the whole video is considered.

There are no fix percentage values to evaluate the actions. Based on the action length, one can choose different time checkpoints to evaluate it. However, an interesting choice is to measure the accuracy starting at 10% of the total number of frames with a step size of 10%. With this setup it is possible to easily visualize how the accuracy varies throughout the action. It is expected to see the metric improving with additional information.

3.1.6 Point-level action start detection mAP

Point-level action start detection mAP is a metric proposed by SHOU *et al.* to evaluate ODAS [86] performance. ODAS is a framework that aims to detect action starts in realistic, unconstrained videos, as discussed in Section 2.2.4. Detecting the start of an action is a task highly related to early action detection, however, the 'percentage of frames seen' metric cannot be applied in this case since it assumes that the action occurs from the beginning to the end of the video. By definition, action start detection needs to be evaluated in videos with periods with no action (background frames).

ODAS framework generates action start (AS) predictions as new frames are analyzed. Each AS prediction is associated with the time point, predicted class and confidence score. In point-level AS detection mAP, each AS point prediction is counted as correct when its action class is correct and its offset is smaller than the evaluation threshold. The point-level AP is evaluated for each class and the point-level mAP is the average over all classes.

3.2 Experimental setup

In this section, we will define all the implementation details needed to perform the experiments that will assess the current online action recognition methods. The tools used for the implementation of the algorithms are discussed in Section 3.2.1. Later, in Section 3.2.2, we describe the network architectures that will be evaluated in the experiments. Finally, in Section 3.2.3, we present all the information needed to train the networks and obtain the models used in this work.

3.2.1 Tools and frameworks

In this section, the tools and frameworks used to implement and train the neural networks evaluated by this work will be presented. Their main characteristics will be addressed, as the motivations that led to their choice. The functions expected for each tool, their importance and role in the development of the project will also be described.

Pytorch

Pytorch is an open-source machine learning framework that is gaining a lot of space in the community for providing maximum flexibility and speed for deep learning research. It is a Python scientific computing package that explores the power of Graphics Processing Units (GPUs) and was based on the Torch library, a previous machine learning framework that is no longer in active development. Launched in 2016, Pytorch provides two very interesting high-level features:

- Tensor computing (like the NumPy library) with strong GPU acceleration;
- Deep neural networks built on a tape-based autograd system[1].

Many popular deep learning frameworks, like TensorFlow and Caffe, use static computational graphs, which are harder to implement and debug but can achieve some valuable computing optimizations. PyTorch adopts the use of dynamic computational graphs, which offer greater flexibility in building complex architectures without losing much efficiency. This feature allows you to change the graph during runtime, which is useful for solutions that are based on variable data.

Some advantageous features of Pytorch are the support to CPU and GPU execution, distributed training and custom data loaders. It is also Python-based, using its concepts like classes and structures. Hence, you can easily manipulate it or integrate it with your favorite Python libraries.

Pytorch is constantly improving and has a very active and growing community. Due to the ease of use and the efficiency that it offers, Pytorch was chosen as the framework to implement, train and evaluate all the models studied in this work.

Azure Machine Learning

Microsoft Azure is a cloud computing platform created by Microsoft for building, deploying and managing services and applications. Its solutions can be offered at different abstraction levels, as infrastructure as a service (IaaS), platform as a service (PAAS) or software as a service (SAAS). Azure provides a wide range of cloud

[1]Pytorch automatic differentiation module is described in [92].

services, like storage, networking, computing, analytics and machine learning, that can be used from research to production.

Microsoft Azure Machine Learning (AML) is a cloud service targeted to build and deploy models faster and at scale, accelerating the end-to-end machine learning lifecycle. It offers web interfaces and software development kits (SDKs) to manage data, experiments and workflows and support the implementation and development of machine learning solutions. They can be used with open-source Python frameworks, such as PyTorch, TensorFlow and scikit-learn, which are familiar tools to the Machine Learning community. AML also provides great scalability, flexibility and an intuitive interface.

The AML service was used throughout the implementation of this work, from the development environment to the distributed computing set for training. The development environment consisted of a single GPU virtual machine, while the training environment worked as an on-demand GPU cluster for running experiments.

3.2.2 Network architectures

In this work, we choose to explore the non-local block, proposed by WANG *et al.* in [70], that has been briefly discussed in Section 2.2.1. The non-local block works as a building block for capturing long-range dependencies and can be plugged into many computer vision architectures. Any neural network can be turned into its non-local version by adding non-local blocks to its structure. Many papers [12, 93, 94] have already explored their use, showing that non-local blocks are likely to become a common component of future architectures.

Following [70], we use the ResNet I3D architecture for baseline, as it achieves the current state of the art results for action classification. Both architectures, the baseline (ResNet I3D) and its non-local version (non-local ResNet I3D) are described below, while the full architecture and implementation details can be found in [95].

ResNet I3D

ResNet I3D is the 3D version of the residual network (ResNet) [15], obtained by "inflating" all its kernels to perform 3D convolutional operations. A traditional 2D kernel $k \times k$ can be inflated to a 3D kernel $t \times k \times k$ that spans t frames. Each of the t planes can be initialized from the pre-trained weights of the 2D $k \times k$ kernel rescaled by $1/t$. With this initialization, a video composed of a static frame repeated in time produces the same result as the 2D model on the single frame.

The decisions made by WANG *et al.* to inflate the ResNet kernels are the following:

- Inflate the first 1×1 kernel in a residual block to $3 \times 1 \times 1$ (similar to [63]);

- Inflate only one kernel for every 2 residual blocks. Thus, residual blocks in sequence alternate between inflated and not;
- Inflate the first convolutional layer from 7×7 to $5 \times 7 \times 7$.

Table 3.1 shows the I3D baseline obtained from a ResNet50 backbone.

Table 3.1: The baseline ResNet50 I3D model. In each layer we describe the values for the kernel size, batch normalization and stride, respectively.
[1]The first convolution of each residual block can be either $(3 \times 1 \times 1)$ or $(1 \times 1 \times 1)$.
[2]The first convolution of the first residual block has stride $(1 \times 2 \times 2)$.

Layer		Output size
conv_1	$(5 \times 7 \times 7)$, 64, $(2 \times 2 \times 2)$	$(16 \times 112 \times 112)$
pool_1	$(2 \times 3 \times 3)$, max, $(2 \times 2 \times 2)$	$(8 \times 55 \times 55)$
res_2	$\begin{bmatrix} (3 \times 1 \times 1), 64, (1 \times 1 \times 1) \\ (1 \times 3 \times 3), 64, (1 \times 1 \times 1) \\ (1 \times 1 \times 1), 256, (1 \times 1 \times 1) \end{bmatrix} \times 3$	$(8 \times 55 \times 55)$
pool_2	$(2 \times 1 \times 1)$, max, $(2 \times 1 \times 1)$	$(4 \times 55 \times 55)$
res_3	$\begin{bmatrix} (3 \times 1 \times 1)^{\mathbf{1}}, 128, (1 \times 1 \times 1) \\ (1 \times 3 \times 3), 128, (1 \times 1 \times 1)^{\mathbf{2}} \\ (1 \times 1 \times 1), 512, (1 \times 1 \times 1) \end{bmatrix} \times 4$	$(4 \times 28 \times 28)$
res_4	$\begin{bmatrix} (3 \times 1 \times 1)^{\mathbf{1}}, 256, (1 \times 1 \times 1) \\ (1 \times 3 \times 3), 256, (1 \times 1 \times 1)^{\mathbf{2}} \\ (1 \times 1 \times 1), 1024, (1 \times 1 \times 1) \end{bmatrix} \times 6$	$(4 \times 14 \times 14)$
res_5	$\begin{bmatrix} (1 \times 1 \times 1)^{\mathbf{1}}, 512, (1 \times 1 \times 1) \\ (1 \times 3 \times 3), 512, (1 \times 1 \times 1)^{\mathbf{2}} \\ (1 \times 1 \times 1), 2048, (1 \times 1 \times 1) \end{bmatrix} \times 3$	$(4 \times 7 \times 7)$
global average pool, fc		$(1 \times 1 \times 1)$

Non-local ResNet I3D

To obtain a non-local ResNet I3D, one must add non-local blocks to its architecture. By doing so, the network becomes capable of performing non-local operations, which attempt to capture long-range dependencies in space and time. WANG *et al.* [70] defines a generic non-local operation as:

$$\mathbf{y}_i = \frac{1}{C(\mathbf{x})} \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j) g(\mathbf{x}_j) \tag{3.7}$$

where i is the index of the output position and j is the index that enumerates all possible input positions. $C(\mathbf{x})$ is a normalization factor. The pairwise function f

computes a scalar that represents the relationship between j and i, as the importance of j to i. The function g computes a representation of the input signal at the position j in the form:

$$g(\mathbf{x}_j) = W_g \mathbf{x}_j, \tag{3.8}$$

where W_g is a weight matrix to be learned. The non-local behavior is achieved by considering all positions ($\forall j$) in the operation and not just some local neighborhoods, as in the convolutional operation, for example.

There are many possible implementations for the pairwise function f. WANG *et al.* explore the use of Gaussian, embedded Gaussian, dot product and concatenation, with the normalization factor $C(\mathbf{x})$ varying accordingly. All versions have shown similar results and the embedded Gaussian was chosen as the default implementation given its similarity to the self-attention module [96]. Hence, f is defined as:

$$f(\mathbf{x}_i, \mathbf{x}_j) = e^{\theta(\mathbf{x}_i)^T \phi(\mathbf{x}_j)}, \tag{3.9}$$

where $\theta(\mathbf{x}_i) = W_\theta \mathbf{x}_i$ and $\phi(\mathbf{x}_j) = W_\phi \mathbf{x}_j$ are two embeddings and the normalization factor is set to $C(\mathbf{x}) = \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j)$. The coefficients in matrices W_θ and W_ϕ are learned.

Replacing Equations (3.8) and (3.9) in Eq. (3.7) we obtain:

$$\mathbf{y}_i = \frac{1}{\sum_{\forall j} e^{(W_\theta \mathbf{x}_i)^T (W_\phi \mathbf{x}_j)}} \sum_{\forall j} e^{(W_\theta \mathbf{x}_i)^T (W_\phi \mathbf{x}_j)} W_g \mathbf{x}_j, \tag{3.10}$$

that, for a given i becomes the softmax computation along the dimension j. Rewriting Eq. (3.10) using the softmax formula (Eq. (3.2)) we have:

$$\mathbf{y} = \text{softmax}(\mathbf{x}^T W_\theta^T W_\phi \mathbf{x}) g(\mathbf{x}), \tag{3.11}$$

which is the self-attention form in [96].

The non-local operation is wrapped into a non-local block that allows it to be incorporated in many existing architectures. The non-local block is defined as:

$$\mathbf{z}_i = W_z \mathbf{y}_i + \mathbf{x}_i, \tag{3.12}$$

where W_z is a weight matrix, $\mathbf{y}_i$ is the output of the non-local operation and "$+\mathbf{x}_i$" is a residual connection [15]. The residual connection allows the non-local block to be inserted into any pre-trained network without changing its initial behaviour if W_z is initialized as zero. The non-local block structure is illustrated in Figure 3.2.

Finally, to obtain the non-local ResNet50 I3D, 5 non-local blocks are added: 2 in the res_3 layer and 3 on the res_4 layer. On res_3 the non-local blocks are placed

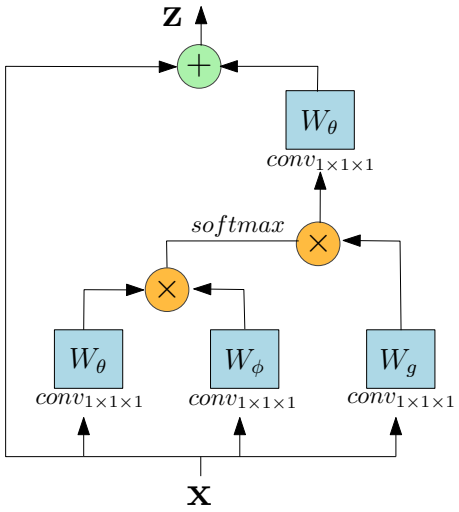


Figure 3.2: The embedded Gaussian version of the non-local block. "$\otimes$" represents a matrix multiplication and "$\oplus$" represents an element-wise sum. *Source:* Adapted from [70].

after the second and forth residual blocks. In res_4, the non-local blocks are placed after the second, forth and sixth residual blocks.

3.2.3 Training procedure

Datasets: To assess the models' performance in action classification and online action tasks two different dataset types were needed: a classification one (with single trimmed actions) and a localization one (with multiple actions). Many available dataset options fit these requirements and some of them were presented in Section 2.1. The ones selected to fill these requirements, respectively, were Kinetics and Charades. They were chosen for being challenging and popular datasets that were used in [70], allowing a direct comparison of the metrics.

Input video clip: The I3D network receives an input in the form $h \times w \times t$, where h and w are the height and width of the frame and t is the number of frames. The value of t defines the temporal distance: how far back in time we can look to get the result for a specific frame. Increasing this value leads to better results, as investigated in [70], since the model can use more information. However, it also increases the input size, which consequently increases the number of parameters and the quantity of memory required to run the model. To avoid this growth, the sequence of consecutive frames is subsampled to the desired number of frames. Considering the same time span, 64 consecutive frames, two settings were used:

- 32×2: Selecting a total of 32 frames with a stride of 2
- 8×8: Selecting a total of 8 frames with a stride of 8

Both settings show similar results, with the 8×8 one being considerably faster.

These frame selection settings were supposed to operate over the extracted frames of the videos from the datasets, that were extracted at a fixed frame per second (fps) rate of 24. For storage efficiency, we have decided to operate directly on the videos, extracting their frames in memory, without saving them. The fps rate of the videos varies from 6 to 120, having an average of 24. To compensate for the fps difference we select a fixed time gap, instead of a fixed number of frames to operate. In 64 consecutive frames of a video with 24 fps, we have, approximately, 2.7 seconds of video. Thus, to maintain the temporal distance of the model, we consider the number of frames that would result in 3 seconds of video based on its fps rate. From this variable quantity of consecutive frames, we select 32 or 8 evenly distributed frames.

Pre-trained weights Ideally, no network training was supposed to be needed since our goal is to evaluate models that already have their performance validated by the community. However, it is hard to find pre-trained weights for some of the most recent networks, as in our case. WANG *et al.* released an online repository [97] with the trained weights on the Kinetics dataset for the ResNet50 I3D and its non-local version. These weights were used as a starting point for training the ResNet models in the Charades dataset.

The repository provided by the authors was develop based on the Caffe2 [98] framework, which is currently a deprecated machine learning framework that became part of Pytorch. The pre-trained weights needed to be converted from Caffe2 to Pytorch to be used in Kinetics evaluation and Charades training. The weight conversion algorithm is a contribution of this work and is publicly available in [95].

Inference: The traditional inference performed on action classification consists in selecting 10 clips evenly spaced in time from a full video and compute their softmax scores individually. The final prediction for the video is the average of all clips. For future references, this evaluation setting will be called "*10-clips video eval*". For each video, its shortest side is rescaled to 256 and fully-convolutional inference [62] is performed. In this inference, the fully-connected layer is replaced by a $1 \times 1 \times 1$ 3D convolutional layer and its weights are adapted to the dimensions of the convolutional kernel. This replacement allows the model to process inputs of variable sizes, since the fixed input size of the fully connected layer limited the model's input dimensions.

During fully-convolutional inference, the whole frame can be processed, so there is no need to crop it to a fixed size. The results using this method are better since they use more information, but the quantity of memory required during inference

also grows with the input size. An alternative to the fully-convolutional inference is to select a fixed size center crop of the frame after rescaling its shortest side to 256. This setting is faster and requires less memory because the input has a smaller fixed size, however, it can achieve worse results since the model will not use the whole frame and might miss some important information.

Training parameters: The training process was the same for all networks, using minibatch stochastic gradient descent [99] with momentum set to 0.9, weight decay (L_2 penalty) of 10^{-5} and minibatch size of 16. The initial learning rate was set to 1.0 and decreased by a factor of 0.1 after learning stagnates for 3 epochs. The training stops after 40 epochs and was done in parallel using 8 Tesla P100 GPUs. A data augmentation scheme is applied during training, in which the frames are randomly scaled so that their smaller size is in the range $[256, 320]$ pixels; then, a random crop of size 224×224 is extracted and randomly flipped horizontally.

3.3 Experimental results

3.3.1 Top-N accuracy (Sec. 3.1.1)

The pre-trained weights of the ResNet50 I3D models on the Kinetics dataset were translated from Caffe2 to Pytorch, as mentioned in the last section. To verify the translated models, they were evaluated using the *10-clips video eval* with and without fully-convolutional inference. The results obtained are shown in Table 3.2. The Pytorch models achieve comparable results to the original Caffe2 ones, validating the weight conversion algorithm.

Table 3.2: Per video Top1/Top5 accuracy results on the Kinetics datasets for the Pytorch ResNet50 I3D models. The results reported by the authors are available in [97].

Model	Input size	Fully conv.	Avg. batch time	Per video Top1/Top5	Reported by authors
resnet50_I3D	32	-	0.43s	71.9/89.9	-
resnet50_I3D	32	Yes	0.61s	73.3/90.7	73.3/90.7
resnet50_I3D_NL	32	-	0.43s	73.4/91.1	-
resnet50_I3D_NL	32	Yes	0.69s	74.4/91.5	74.9/91.6
resnet50_I3D	8	-	0.28s	71.7/90.1	-
resnet50_I3D	8	Yes	0.39s	73.2/90.8	73.4/90.9
resnet50_I3D_NL	8	-	0.28s	73.5/91.0	-
resnet50_I3D_NL	8	Yes	0.49s	74.5/91.5	74.7/91.6

Both settings for the input video clip achieved similar results, with the smaller one, with 8 frames, being considerably faster since it processes fewer data. The input video clip sizes are indicated in the second column of Table 3.2 by the number of frames they possess, 8 and 32. The closeness in the results obtained for these two settings indicates that there may be redundancy in the additional information of the input clip with 32 frames. Another comparison shown in Table 3.2 is between the two evaluation modes: fully convolutional and center crop. Fully convolutional inference achieves higher accuracy, but with a considerable increase in the computational time.

For the streaming environment, each frame of the Kinetics dataset was evaluated separately, in a per-frame evaluation. The Kinetics validation set has approximately 19.7K videos. In the per-video setting (*10-clips video eval*), 197K frames were evaluated, but in the per-frame setting this number increases by a factor of 23, for a total of approximately 4.7M frames. Due to the large amount of data to be processed, we only evaluate the models with input clip size 8 since they achieve similar results to the ones with input clip size 32 (74.4% and 74.5%, respectively) and require more computation time and memory. Table 3.3 shows the results of the Top1/Top5 accuracy considering all frames and compares them to the per-video ones obtained in Table 3.2, showing the percentage points difference between the results.

Table 3.3: Per frame Top1/Top5 accuracy results on the Kinetics datasets for the Pytorch ResNet50 I3D models with input clip size 8.

Model	Fully conv	Per-video Top1/Top5	Per-frame Top1/Top5	Diff. (pp)
resnet50_I3D	-	71.7/90.1	65.1/85.1	6.6/5.0
resnet50_I3D	Yes	73.2/90.8	67.4/86.6	5.8/4.2
resnet50_I3D_NL	-	73.5/91.0	66.9/86.2	6.6/4.8
resnet50_I3D_NL	Yes	74.5/91.5	69.1/87.6	5.4/3.9

There is a drop in accuracy when considering all frames independently in the evaluation. In the per-video evaluation, eventual prediction errors were attenuated by averaging the results of the 10 video frames before deciding on a final label. In the per-frame case, we cannot wait until the end of the video to do an average, so the prediction errors became more prominent. The Top1 metric was more affected, losing around 6.1pp percentage points while the Top5 one has lost around 4.5pp, as shown in the last column of Table 3.3.

3.3.2 Percent of frames seen (Sec. 3.1.5)

In the Kinetics dataset we can also evaluate the 'percentage of frames seen' metric. Following common practice, we evaluate the accuracy from 10% to 100% of the total number of frames with a step size of 10%. The final accuracy for each time point is obtained by averaging the softmax scores of all passed frames. Figure 3.3 shows the evolution of the Top1/Top5 accuracy over the percentage of frames seen of actions. As the per-frame metric of last section, we only evaluate the models with input clip size 8.

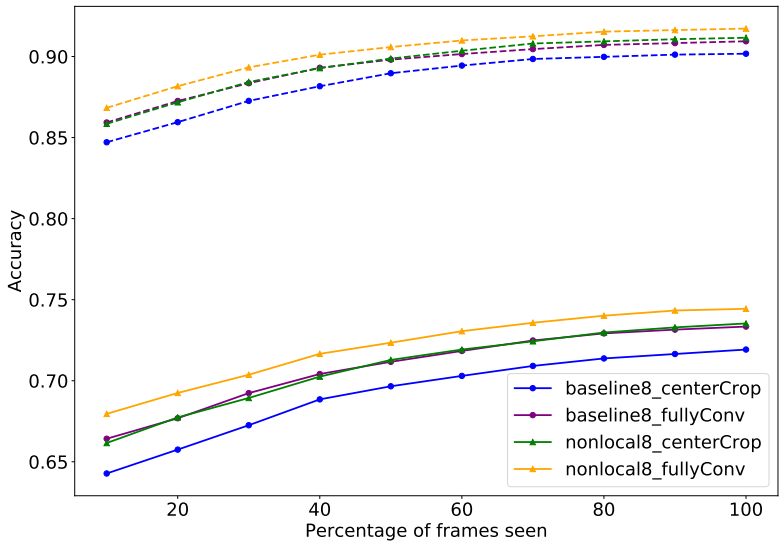


Figure 3.3: Accuracy for different time points in the video, following "percentage of frames seen" metric. Solid and dashed lines represent, respectively, Top1 and Top5 accuracy.

As expected, accuracy increases with the addition of frames to be used in the calculation of the final prediction for the video. Comparing the beginning to the end of the video, there is a relative accuracy drop of, approximately, 10% and 6% in Top1 and Top5 metrics, respectively. This variation in accuracy is one of the challenges that the action start detection task tries to overcome. A desirable result would be to achieve a more horizontal progression line, meaning that the model was able to predict the correct class by only seeing the beginning of the action without losing much accuracy.

It would be interesting to compare the accuracy drop of the classification models evaluated in this work with the drop achieved by some action start detection methods. This comparison would give us an idea of the effectiveness of these methods by comparing them to a classification method that does not focus on improving the accuracy at the beginning of the action. Unfortunately, it not possible to do

a direct confrontation between those approaches as they use different datasets and metrics [24, 84, 100]. The closest example is presented in [61] and calculates the accuracy for the UCF-101 dataset (Sec. 2.1.1) at 1% and 25% of the action, achieving an accuracy drop of 3.5% in the Top-1 results at those two points. In our case, the percentage difference between the accuracy at 10% and 30% is about 4% for Top1 and 3% for Top5.

3.3.3 Mean Average Precision (Sec. 3.1.2)

The mean average precision (mAP) metric was evaluated in the multi-label Charades dataset. The metric was implemented following the official evaluation code ("*Charades_v1_classify.m*") provided by the authors of the Charades Challenge in [101]. Table 3.4 shows the result of the mAP for the "per-video" and "per-frame" cases. The "per-video" results were obtained with the *10-clips video eval*, which is the standard evaluation protocol for this metric and dataset. As in the experiments with the Top-N metric, we perform a "per-frame" evaluation to simulate a streaming environment by considering all frames in the dataset.

Table 3.4: Per frame and per video mAP (%) results on the Charades dataset for the ResNet50 I3D models.

Model	Input size	Fully conv.	Per-video mAP	Per-frame mAP
resnet50_I3D	32	-	31.3	17.5
resnet50_I3D	32	Yes	32.1	18.1
resnet50_I3D_NL	32	-	32.1	18.1
resnet50_I3D_NL	32	Yes	32.5	18.4
resnet50_I3D	8	-	30.1	17.1
resnet50_I3D	8	Yes	30.7	17.8
resnet50_I3D_NL	8	-	31.9	18.3
resnet50_I3D_NL	8	Yes	32.0	18.3

Primarily, the results needed to be validated since there were no official pre-trained weights available at the time this work was produced. As discussed in Section 3.2.3, the models were trained on the Charades dataset using the weights provided by WANG *et al.* for the Kinetics dataset as a starting point. The mAP achieved for the ResNet50 I3D and its non-local version are comparable to the ones reported by WANG and GUPTA [102] for fully convolutional inference with an input size of 32 frames, which are, respectively, 31.8% and 33.5%.

The difference between the per-video and per-frame results in this case is substantial, with a decrease of, approximately, 13.5 percentage points, which means a

relative drop of 43% in the results. The impact of per-frame evaluation on Charades is higher than on Kinetics since this dataset can have multiple classes predicted for each frame, increasing the chances of false positives.

3.3.4 Calibrated Average Precision (Sec. 3.1.4)

The experiments with the calibrated average precision (cAP) metric were the same as the ones performed with mAP, except for the metric applied. As presented in Section 3.1.4, cAP is a metric for per-frame evaluation but we have also performed the per-video evaluation for comparison reasons. The results for the "per-video" and "per-frame" cases are shown in Table 3.5.

Table 3.5: Per frame and per video cAP (%) results on the Charades datasets for the ResNet50 I3D models.

Model	Input size	Fully conv.	Per-video cAP	Per-frame cAP
resnet50_I3D	32	-	83.2	82.4
resnet50_I3D	32	Yes	83.3	82.2
resnet50_I3D_NL	32	-	83.7	82.9
resnet50_I3D_NL	32	Yes	82.8	80.9
resnet50_I3D	8	-	82.6	82.1
resnet50_I3D	8	Yes	82.5	81.6
resnet50_I3D_NL	8	-	83.9	83.2
resnet50_I3D_NL	8	Yes	82.3	80.5

There is a much smaller variation between the results for the per-frame and the per-video cases, which is expected since cAP takes into consideration the number of positive and negative samples evaluated for each class. Moreover, the results remain mostly constant throughout all settings. Surprisingly, fully convolutional evaluation slightly deteriorates cAP, instead of improving it.

Figure 3.4 shows histograms for AP and cAP per class values on the non-local model with an input size of 32 frames and fully convolutional inference. The AP is highly influenced by the ratio between positive and negative samples, having its distribution varying accordingly to the distribution of samples for each class, shown in Figure 3.5, where a few classes have many samples and many classes have few samples. The calibrated AP 'compensates' for the unbalanced number of samples in each class, generating more uniform values of cAP and therefore having a more Gaussian distribution.

The cAP results are substantially higher than the ones achieved for mAP. These

conflicting results give us different ideas about the model's performance, pointing out the necessity of new and more tangible metrics.

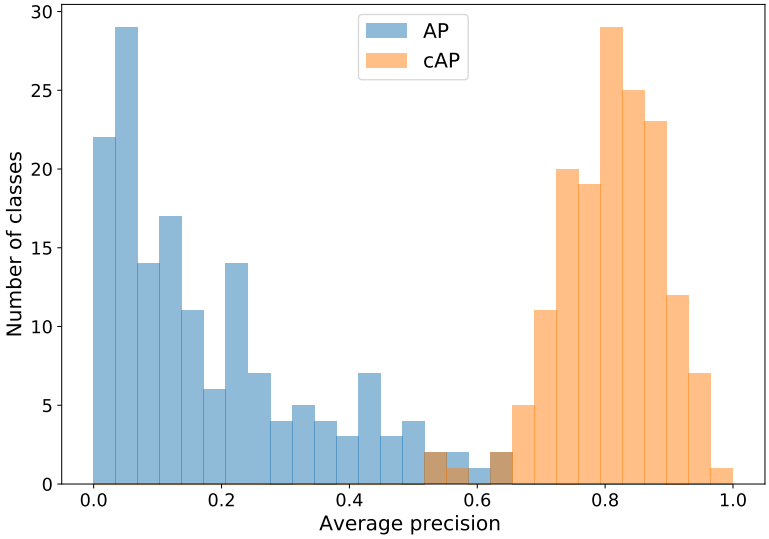


Figure 3.4: Histogram of per class average precision (AP) and calibrated average precision (cAP) for the resnet50_I3D_NL model with input size of 32 frames and fully convolutional inference.

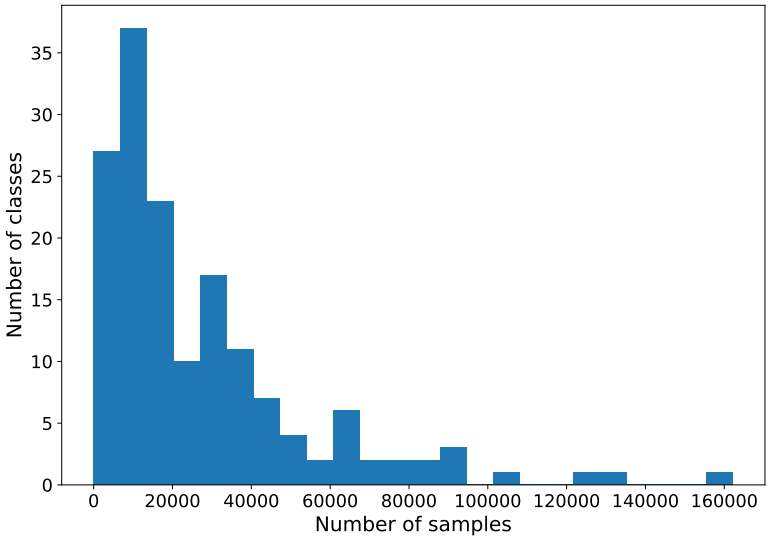


Figure 3.5: Histogram of number of samples (positive frames) per class in the Charades dataset.

Chapter 4

Proposed metrics and protocols

In the last chapter, we presented some of the main metrics applied in the field of action recognition and performed a few experiments to explore their performance in a streaming scenario. Some metrics were already appropriate for a per-frame evaluation, like 'percentage of frames seen' (Sec. 3.3.2) and cAP (Sec. 3.3.4), while others were not usually applied to this kind of situation and needed to be adapted, as Top-N (Sec. 3.3.1) and mAP (Sec. 3.3.3).

Those metrics were adapted to a naïve per-frame version, which achieves considerably lower results when compared to the original per-video ones. The performance drop is expected since we do not apply any mean or filter over the predictions, which could minimize the impact of the false positives. Furthermore, some metrics (mAP and cAP) achieved inconsistent results regarding the performance of the models in a streaming scenario (18.4% mAP and 80.9% cAP, for example). This raises a questioning about how suitable they are for a streaming case or if we are interpreting them correctly.

In this chapter, we investigate those questions by performing new experiments, starting some discussions and even re-training some models. In Section 4.1, we explore different ways to achieve those per-frame results, trying to enhance the frame prediction while maintaining the streaming requirement. Later, in Section 4.2, different ways to measure the performance in an online, streaming scenario are studied, and new metrics are proposed. Finally, in Section 4.3, the interpretation of desirable action classes is discussed, and a new model is trained and evaluated.

4.1 Enhanced per-frame metrics

In our naïve per-frame adaption of the metrics that are traditionally applied per-video, we considered the individual results of all frames in the metrics calculation. The videos of the datasets evaluated have an average frame rate of 24, thus, for each second of video, we obtained around 24 frames and generated 24 individual results.

Returning 24 results per second is probably unnecessary for most online action classification applications. Depending on the requirements of the application, one might need an output for every 0.5 seconds, 1 second or even longer periods.

After those considerations, we could consider the naïve per-frame metric as the 'worst case' scenario, in which we have to return an output for every frame received. If it is possible to wait a few milliseconds (or more), we can apply some sort of aggregation function over the results received until the allowed delay time, achieving an improved final result for that period.

In the following sections, we will explore different ways to improve the per-frame results of our trained models. We believe this is an important study to be made since training deep learning models with video data requires a lot of time and computational resources, making it necessary to take advantage of the available pre-trained weights. It is important to notice that the network architectures for which there are available pre-trained weights might not fit one's application requirements. For example, in our case, the ResNet I3D was not developed to work in a causal, per-frame way. We adapted it by only looking at past frames in the input clip, as shown in Figure 4.1. Limited by the available architectures, we want to explore how to achieve their best performance.

4.1.1 Maximum delay

Figure 4.1 shows a representation of the set of frames considered in the input clip to achieve the result for a frame in our naïve per-frame setting. To simplify, the input clip shown in the image is composed of 5 sequential frames. The input clip for frame

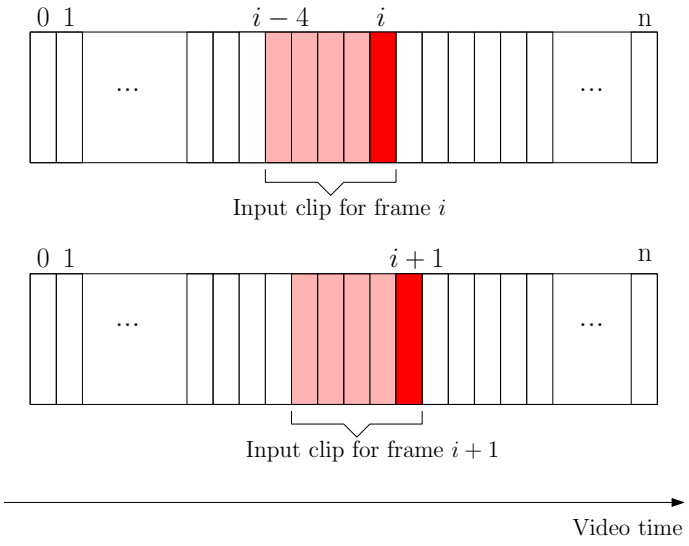


Figure 4.1: Graphic representation of the data considered by an input clip to achieve the individual result of the frame i in a video. In this example, the input clip is composed of 5 sequential frames.

i can only contain past frames, given the causality requirement of a video streaming. If the delay time of our application is zero, we cannot look any further in the future to improve the result for frame i. However, if we can wait enough time to get the result for future frames, we can use them to improve a past frame prediction.

Notice that the data contained in the frames $[i-4,\ i]$ is used to compute the prediction for frame i. Similarly, the data contained in the frames $[i-3,\ i+1]$ is used to achieve the result for the frame $i+1$. Both input clips used the data contained in frame i, among other frames, to get their individual results. Figure 4.2 shows all input clips that would include data from frame i. There are 5 clips (from frame i to frame $i+4$) that carry the information of frame i, and therefore, could be used to improve the result of frame i.

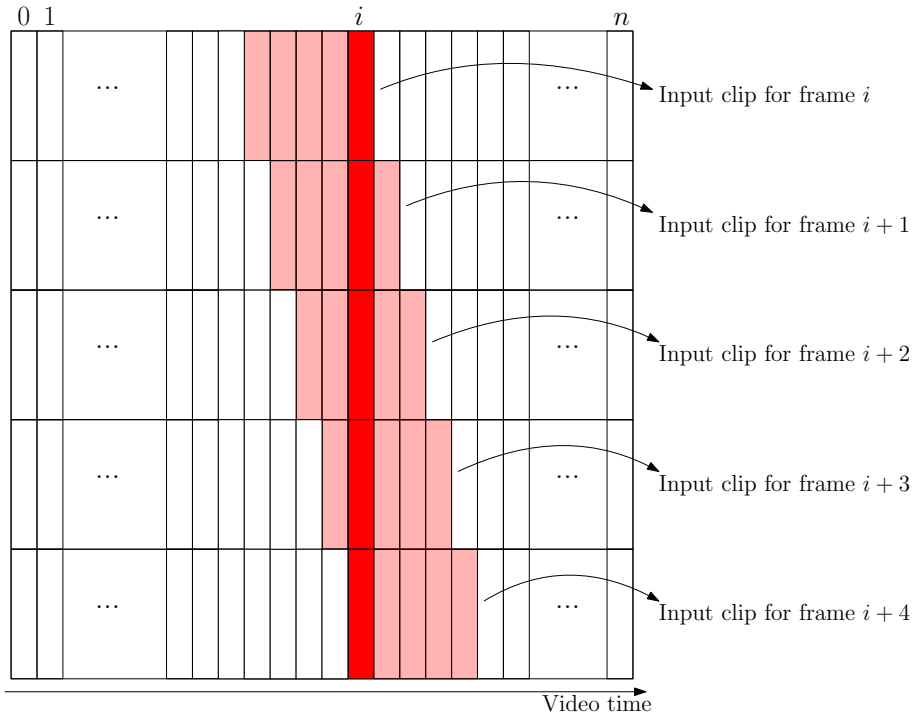


Figure 4.2: Graphic representation of all input clips that would considered the data in the frame i when getting their individual results for a frame.

An improved final result for a single frame can be achieved by using all individual results that considered that frame in its input clip. The first input clip to consider the information of frame i is the frame output for frame i, while the last input clip to consider the information of frame i is the frame output for frame $i+t$, where t is the time span covered by an input clip. In our models, the time span of the input clip is 3 seconds, as discussed in Section 3.2.3. Thus, the delay time for a frame result in our case can vary from 0s to 3s. In the next section, we experiment with different delay values and aggregation functions, studying the variation of the results for each setting.

4.1.2 Aggregation functions

Several functions can be used to aggregate the individual results of the frames. In this section, we study the use of four different functions: mean, median, Gaussian, and maximum value. For each function, we vary the delay time to evaluate their overall performance and how they benefit from the extra information of the frames. Figures 4.3 and 4.4 show, respectively, the mAP and cAP results for each combination of delay time and aggregation function for the non-local ResNet50 I3D model with an input size of 32 frames and fully convolutional inference. In this experiment, we do not consider the inference time of the model in our 'delay time'. Table 4.1 shows the execution time of each function to achieve the final result for a frame when the delay time is 3 seconds, which contains the higher number of input clips for frame. Those values were obtained by averaging the total time needed to calculate the enhanced per-frame results of all frames in the dataset.

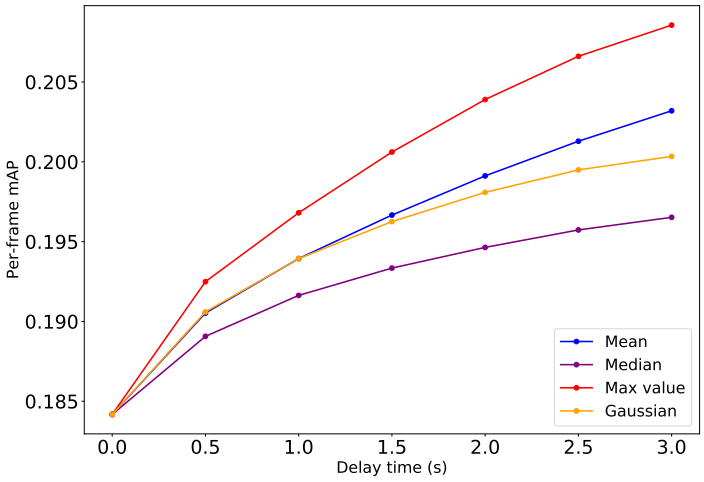


Figure 4.3: Per-frame mean average precision (mAP) results when applying different frame aggregation functions and delay time for the resnet50_I3D_NL model with an input size of 32 frames and fully convolutional inference.

All functions were capable of improving the result for a frame by considering the information added in the delay time. For both metrics, mAP and cAP, applying the maximum over the data led to the best results. This might be because the dataset we are evaluating is multi-label, thus, getting the maximum prediction over the clips helps in not missing a true positive. Within the evaluated functions, the maximum and the mean are the best options to achieve the enhanced per-frame results. The choice of one over the other depends on a possible time performance requirement since maximum achieves the higher results (with mean in the second place) and mean is the faster function (with maximum in the second place), as shown in Table 4.1.

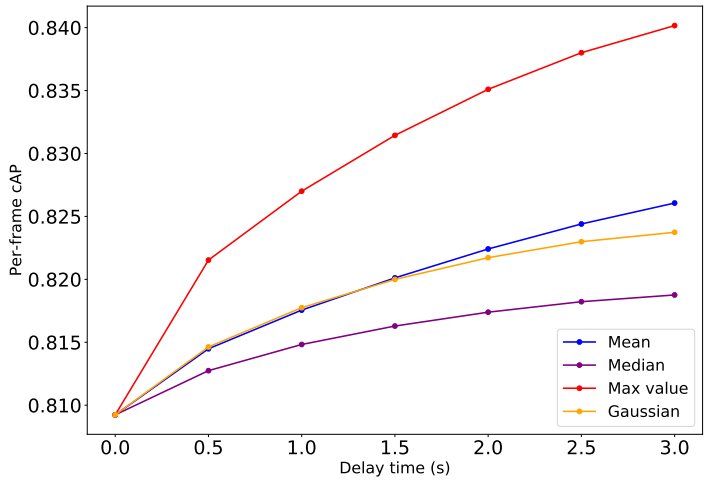


Figure 4.4: Per-frame calibrated average precision (cAP) results when applying different frame aggregation functions and delay time for the resnet50_I3D_NL model with an input size of 32 frames and fully convolutional inference.

Table 4.1: Average time needed for each function to achieve a result per frame when delay time is 3 seconds.

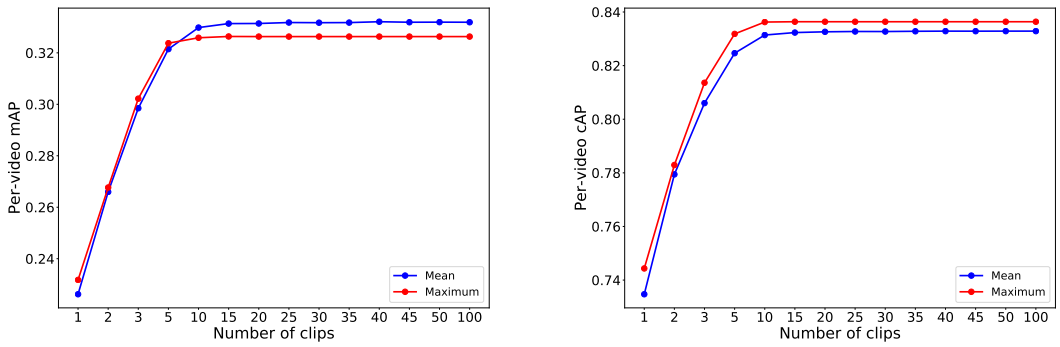
Aggregation function	Time per-frame (ms)
Mean	0.026
Median	0.225
Max value	0.050
Gaussian	0.073

Considering the data added in the delay time of 3 seconds, the mAP increased by, approximately, 2.5 percentage points, which means a relative improvement of 13.6%. Similarly, the cAP metric increased 3.1 percentage points, which means a relative improvement of 3.8%. The 'enhanced per-frame' results can be used together with other metrics, further increasing their results, as we will see in the next section.

4.1.3 Number of clips

There are cases in which the delay time can be over 3 seconds, allowing us to use the "mean of evenly spaced clips" approach. Averaging the results for evenly spaced clips in a video (usually 10 clips) is a standard approach within action recognition. Besides being used in many metrics, as Top-N and mAP, this idea has also been successfully incorporated into some network architectures [31, 32]. Figure 4.5 shows the mAP and cAP per-video results for different numbers of clips on the non-local ResNet50

I3D model with an input size of 32 frames and fully convolutional inference. The results for each clip were obtained using the enhanced per-frame result applying the mean and the maximum value for a delay of 3 seconds, which was introduced in the last section. For clarification, "clip" means the input clip needed to achieve an individual frame result.



(a) Mean average precision (mAP) (b) Calibrated average precision (cAP)

Figure 4.5: Per-video mAP and cAP results when averaging a different number of clips per video for the resnet50_I3D_NL model with an input size of 32 frames and fully convolutional inference.

We varied the number of clips averaged per video from 1 to 100, and looking at the graphs in Figure 4.5, we notice that the metric saturates for a number of clips higher than 15. Therefore, the "10 evenly spaced clips" approach used in many metrics is a good choice to achieve their best per-video results. Besides that, there is a slight advantage in using the mean or the max enhanced per-frame results depending on the metric that will be applied. The cAP prioritizes the number of positive predictions, which is favored by the max function, while the mAP is highly influenced by the number of false positives, which is diminished by the use of the mean function.

4.2 Proposed metrics

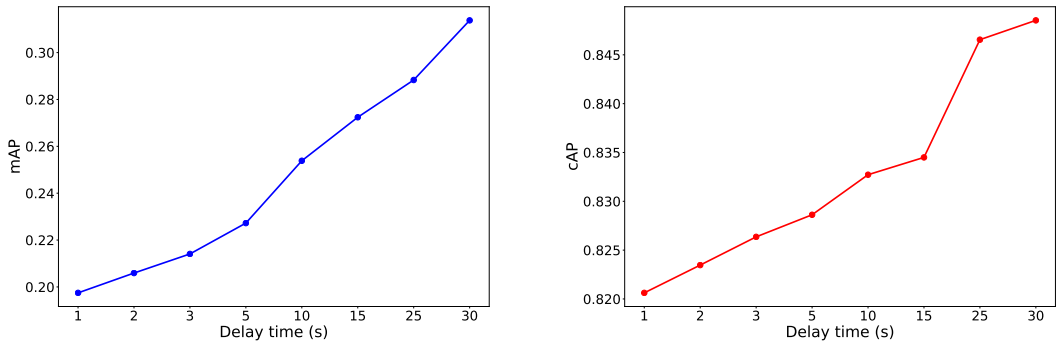
In this section, we will discuss and implement some proposed metrics to action classification in a streaming scenario. Our goal is to elaborate more tangible metrics, with a closer relation to the problems faced in real-world applications.

4.2.1 Metrics per delay

It became clear in the last section that the more we can wait until giving a prediction, the more certain we can be about it. To explore this idea, in this section, we compare

the performance of a model when evaluating it at varied periods of time. In this way, any per-video metric can be seen as a metric obtained for a "video length" delay time, while the per-frame metrics are the ones achieved for a zero delay time. For example, in the Charades dataset, which has an average video time of 30 seconds, the per-video results are the ones achieved when we can wait 30 seconds.

Figure 4.6 shows the mAP and the cAP calculated at different delay times for the non-local ResNet50 I3D model with an input size of 32 frames and fully convolutional inference on the Charades dataset. The result for a delay time was obtained by getting the maximum of 10 evenly spaced clips within the delay period. As expected, the performance increases with the delay time available, until reaching its final "per-video" value. We believe the graph "metric $\times$ delay time" consolidates well our initials per-frame and per-video metrics, giving the information needed for one to choose the best trade-off between performance and delay time depending on its application.



(a) Mean average precision (mAP) (b) Calibrated average precision (cAP)

Figure 4.6: Results for mAP and cAP at different delay times for the resnet50_I3D_-NL model with an input size of 32 frames and fully convolutional inference.

4.2.2 False alarm rate

When dealing with online methods, a tangible measurement for performance is the number of false alerts generated in a period of time. This metric gives a more realistic idea of the model performance since it is possible to relate to its value. For example, if an online, streaming application raises an alert, a human will probably be notified by an e-mail, pop-up or even by a sound signal, requiring his or her attention. Then, the number of false alerts by time represents the unnecessary reactions to the application.

We defined false alarm rate (FAR) as:

$$\text{FAR} = \frac{\text{Number of false predictions}}{\text{Total length evaluated}}. \tag{4.1}$$

To determine which predictions qualify as positives and then calculate the number of false alarms in our evaluation dataset, we need to determine confidence thresholds. As discussed in Section 3.1.2, for a multi-label dataset as Charades, the threshold will determine the trade-off between precision and recall per class, and there is no recipe for choosing the right threshold. We propose two different thresholds setups:

1. Precision = AP: Selecting a threshold so that precision is similar to the average precision;

2. max(precision $\times$ recall): The highest product between precision and recall is, theoretically, a point with a good trade-off between precision and recall, thus a threshold is selected so that this product is maintained.

Table 4.2 shows the results obtained when evaluating the non-local ResNet50 I3D model with an input size of 32 frames and fully convolutional inference on those two different thresholds setups. The FAR values achieved for all thresholds are incredibly high, showing that the model raises dozens of false alarms per second. These results indicate how inappropriate this model can be for online applications in which the generated alarms have a high impact on the real world.

4.2.3 Non-action detection

A more simple question to ask our model is if it is capable of detecting when none of the annotated classes is happening, that is, differentiating between background and any annotated class. This measurement would help us understand, together with FAR, the number of unnecessary reactions to an action classification application. Given a FAR value, we cannot tell if a false alarm was raised when a true alarm was also happening or if it was raised when none of the wanted classes were happening. These two situations can have a different impact on an application performance, with the latter being the worst scenario.

To evaluate this case, we followed the same procedure as in the last section, considering those three delay times and two thresholds to achieve the number of false alerts. Then, for each period of time evaluated, we adapted the prediction vector to a binary value representing 'action' or 'non-action'. If at least one action was positive in the vector, we set it to 'action', if none of the actions were positives, we set it to 'non-action'. With a single class label, we calculated the accuracy for

Table 4.2: False alarm rate (FAR), average number of positive samples and positive predictions per delay period on the resnet50_I3D_NL model with an input size of 32 frames and fully convolutional inference.

Threshold type	Delay time (s)	FAR (Hz)	Avg. # of annotated positives	Avg # of predicted positives	Prec (%)	Recall (%)
prec = AP	1	13.9	3.9	18.1	11.5	52.5
	3	5.9	4.3	21.2	11.7	57.3
	5	4.1	4.6	23.4	11.9	60.1
	10	2.5	5.4	27.2	12.8	64.5
	15	1.8	6.0	29.8	13.6	67.3
	20	1.5	6.3	30.3	14.1	68.4
	25	1.3	6.1	29.1	14.3	68.3
	30	0.4	6.3	32.5	13.5	69.5
max(prec × recall)	1	24.2	3.9	28.5	8.0	58.1
	3	9.6	4.3	31.9	8.5	62.6
	5	6.4	4.6	34.1	8.9	65.1
	10	3.7	5.4	37.9	9.9	69.1
	15	2.6	6.0	40.5	10.6	71.5
	20	2.2	6.3	40.9	11.1	72.5
	25	1.9	6.1	39.6	11.2	72.7
	30	0.6	6.3	42.6	10.8	73.3

each of the classes, since the number of background and action frames is highly unbalanced.

The model predicted 'action' for all samples in all settings assessed. This result is a reflection of the unbalanced data on the Charades dataset, that does not posses many background frames (around 7% of the frames). Therefore, the model did not have enough data to learn how to differentiate background frames from actions frames. This result was also expected given the high number of FAR identified in the last section.

To investigate the reason of so many false alerts we reviewed the action classes of the dataset to verify if there were closely related ones that could be confusing the model. Our findings and next steps are described in the following section.

4.3 Verb classes

The classes of the Charades dataset are arrangements of 33 verbs with 38 objects (including 'None'). For example, the objects [book, laptop] and the verbs [hold, put], generate the classes 'holding a book', 'holding a laptop', 'putting a book somewhere' and 'putting a laptop somewhere'. One of the reasons many action datasets have classes that specify the objects of interaction is because, in some cases, they can help in identifying the action. For example, if the model detects a book in a frame is more plausible that the action in question is 'holding' or 'reading' than 'washing' or 'drinking'. However, this over-specification can negatively impact the performance of action models, which might dilute the accuracy of an action class between all its action-object arranged classes. In the Charades dataset, for example, there is even the case of overlapping classes, like 'someone is smiling' and 'smiling in a mirror', which can confuse the model.

To test if the model is confusing the classes related by verb or object, we adapted the predictions from a trained model to their respective verb-classes or object-classes versions. For example, the 'verb-class' version of the class 'holding a book' is 'hold', while its 'object-class' version is 'book'. Since there are many classes in the original dataset division for that correspond to same object or class, their predictions were aggregated by their mean or maximum value, as exemplified in Equation 4.2:

$$\begin{aligned} \text{pred}_{(\text{hold})} &= \text{mean}\Big(\big[\text{pred}_{(\text{holding a book})}, \text{pred}_{(\text{holding a laptop})}, ...\big]\Big) \\ \text{pred}_{(\text{hold})} &= \max\Big(\big[\text{pred}_{(\text{holding a book})}, \text{pred}_{(\text{holding a laptop})}, ...\big]\Big) \end{aligned} \tag{4.2}$$

where $\text{pred}_{(\text{C})}$ is the prediction for a class C. Table 4.3 shows the mAP and cAP, per-frame and per-video, achieved by adapting the model predictions for their 'verb-class' and 'object-class' versions.

Table 4.3: Mean average precision (mAP) and calibrated average precision (cAP) per-frame and per-video, for the verb-classes and object-classes adapted versions of the results of the resnet50_I3D_NL model with an input size of 32 frames and fully convolutional inference.

Classes type	Agg. function	Per-frame mAP	Per-video mAP	Per-frame cAP	Per-video cAP
Verb	Mean	33.8	52.5	81.4	83.2
	Max	34.1	52.1	81.8	83.0
Object	Mean	36.1	52.7	82.5	84.7
	Max	36.4	52.2	82.9	84.7

The mAP results improved considerably, while the cAP ones remained similar, indicating that the number of false positives decreased. As expected, given a sample that contains a class label formed by a verb v and an object o, the model is predicting with high confidence the classes which contain v or o. The model seems to be learning the information of both, verbs and objects, since their results are very similar.

We believe that action models should focus on detecting solely actions, independent of the object of interaction. Trying to achieve higher results per-verb, we re-train the ResNet50 I3D models on the Charades dataset on the 33 verb classes (instead of the 157 original ones). The training process was the same as the one performed initially (described in Section 3.2.3). Table 4.4 shows the mAP and the cAP results for the ResNet50 I3D models trained.

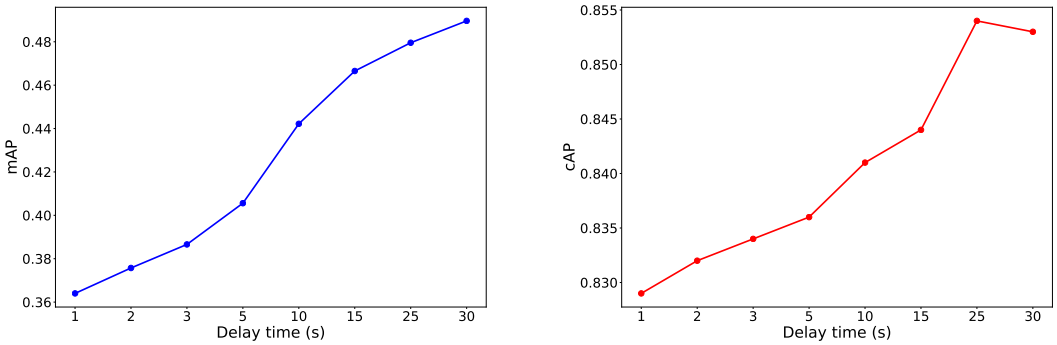
Table 4.4: Mean average precision (mAP) per-frame and per-video, and calibrated average precision (cAP) results on the Charades datasets for the ResNet50 I3D models trained for verb classes.

Model	Input size	Fully conv.	mAP per-frame	mAP per-video	cAP per-frame	cAP per-video
resnet50_I3D	32	-	32.4	50.2	81.5	82.2
resnet50_I3D	32	Yes	33.6	50.8	81.9	82.9
resnet50_I3D_NL	32	-	33.8	51.6	82.3	83.3
resnet50_I3D_NL	32	Yes	34.9	53.1	82.2	83.9
resnet50_I3D	8	-	31.7	49.6	81.0	81.9
resnet50_I3D	8	Yes	32.9	50.3	81.7	82.5
resnet50_I3D_NL	8	-	32.0	50.2	81.6	82.5
resnet50_I3D_NL	8	Yes	33.2	51.3	81.7	83.0

Interestingly, the models trained per-verb achieved results very close to the ones obtained by adapting the predictions of the original models. Since retraining did not improve the performance for verb classes, we believe that the model is using a reasonably even combination of both information, verbs and objects, to determine the classes. It is important to notice that we did not tune our training process to the verb classes, which could improve the results.

We further investigate the performance of the verb model assessing it on our proposed metrics. Figure 4.7 shows the mAP and the cAP calculated at different delay times for the non-local ResNet50 I3D verb model with an input size of 32 frames and fully convolutional inference. Then, Table 4.5 shows the FAR results obtained when evaluating the same model on the different thresholds setups. The FAR results achieved on this model are considerably smaller than the ones calculated for the original one. This fact further consolidates our hypothesis that the original

model confuses verb-related classes, causing a lot of false alarms for those classes. As for non-action detection, the verb model had the same performance as the original one, classifying all samples as positive, since the distribution between background and action frames remained the same.



(a) Mean average precision (mAP) (b) Calibrated average precision (cAP)

Figure 4.7: Results for mAP and cAP at different delay times for the resnet50_I3D_-NL model with an input size of 32 frames and fully convolutional inference.

4.4 Conclusion

In this section we focused on two main necessities: to improve the performance of actual models in online scenarios and to provide more tangible results regarding real-world applications. For the first need, we explored the use of a possible delay in a model's response to aggregate information that improves a past moment prediction. To measure this improvement we proposed two metrics: enhanced per-frame mAP/cAP and mAP/cAP per delay. As for the second need, we proposed two application-driven metrics: false alarm rate and non-action detection.

The model has a very high false alarm rate and could not detect non-action frames (background frames), indicating that it might not be suited for a streaming application. We investigated the reason for those results and discovered that the dataset used had a big impact on them. The Charades dataset is highly unbalanced for 'action' and 'non-action' frames, not providing enough data for the model to learn how to differentiate them. This was not a concern during this dataset creation, but it is an important point to be aware of since it can influence many tasks besides online classification; like action start detection, early action detection, and even action localization.

Moreover, we realized that many action classes are closely related, confusing the model and causing a high number of false alarms. We believe that focusing on verbs solely would be a more suitable division of the classes, and retrained the models

Table 4.5: False alarm rate (FAR), average number of positive samples and positive predictions per delay period on the resnet50_I3D_NL verb model with an input size of 32 frames and fully convolutional inference.

Threshold type	Delay time (s)	FAR (Hz)	Avg. # of annotated positives	Avg # of predicted positives	Prec (%)	Recall (%)
prec = AP	1	2.1	3.0	5.2	34.8	60.2
	3	1.0	3.2	6.1	34.4	65.1
	5	0.7	3.5	6.8	34.8	68.2
	10	0.4	3.9	7.8	36.5	72.5
	15	0.3	4.3	8.6	37.5	74.9
	20	0.3	4.5	8.8	38.2	75.7
	25	0.2	4.3	8.4	38.4	75.4
	30	0.1	4.4	8.7	37.9	75.0
max(prec × recall)	1	4.6	3.0	7.6	30.3	77.1
	3	1.8	3.2	8.4	30.7	79.7
	5	1.2	3.5	8.9	31.4	81.2
	10	0.7	3.9	9.8	33.4	83.4
	15	0.5	4.3	10.5	34.9	84.9
	20	0.4	4.5	10.6	35.6	85.0
	25	0.3	4.3	10.3	35.4	84.9
	30	0.1	4.4	10.5	35.5	84.8

in this setting. The new model achieved a considerably better performance on all metrics evaluated. This shows the influence the data has on the model's performance and highlights the need to produce more datasets focused on online tasks.

Chapter 5

Conclusions

In this work, we have presented a study of application-driven metrics for online action classification in a streaming scenario. Initially, an overview of the action recognition field was presented, detailing its main tasks and datasets. Looking towards application-driven methods, which are covered by the online action task, we have found a lack of consensus regarding the definition of the task itself and its evaluation protocols. Without a proper evaluation protocol, it is hard to understand the actual situation and performance of online action methods, and therefore, we decided to contribute to the consolidation of online classification metrics.

The main metrics applied in the action recognition field were presented and used to assess the performance of state-of-the-art action classification methods in a video streaming scenario. When necessary, the metrics were adapted to their per-frame version to allow a streaming evaluation. The results for a per-frame evaluation were considerably worse than the original ones, pointing out how those metrics do not represent the performance of the model in a real application case. Moreover, the results gave conflicting ideas regarding the performance of the model, making us question how suitable those metrics are for a streaming case, or even if we are interpreting them correctly. To address these questions, we proposed and evaluated some new metrics and protocols oriented to the online streaming case: enhanced per-frame metric, metric per delay, false alarm rate, and non-action detection.

Those metrics were proposed based on two necessities: to improve the performance of actual models in online scenarios, and to provide more tangible results regarding real-world applications. For the first need, assuring causality, we explored the possibility of introducing a delay, successfully enhancing the performance on those cases. Regarding the metrics proposed for the second need, the results showed that the model has an incredibly high rate of false alarms per second and cannot differentiate between background and action frames. However, the dataset used had a big impact on those results, since it has a very unbalanced rate of background and action frames, and possesses closely related, sometimes overlapping, classes.

We further investigate the influence of the dataset classes on the model performance by reallocating them in what we believe to be a more suitable arrangement. Our proposed classes focused on verbs solely, without identifying any object of interaction. In this new setup, the model achieved considerably better results in all metrics evaluated, especially on the false alarm rate, proving how the original class division was confusing the model.

5.1 Next steps

Throughout this work, we came across several open problems of online action recognition. The task covers a wide range of sub-tasks with different specifications that, in general, are not well defined, hindering the definition of an evaluation protocol to be followed or the metrics to be applied. In this work, we contributed to the understanding of some metrics performance on a streaming scenario, but there is still much to be done regarding the definition and clarification of a general evaluation framework.

Within the scope of metrics, there are some ideas that deserve exploration in the future. One of them is to consider the inference time of the model in our delay metrics, such that the performance would depend on how many times you can run the model under the delay time. This would be an interesting experiment to compare different architectures when focusing on real-time applications. Another idea is to explore a more general metric for early action detection in untrimmed datasets. Detecting the beginning of an action is a valuable task for online applications, but its actual metrics tend to be specific for the method being evaluated, generating results that are inconsistent and cannot be compared.

The topic of how to arrange actions into classes is also particularly interesting. This is a fundamental question that, as seen in Section 4.3, can highly improve or decrease the performance of a model. Action datasets are still evolving, mostly through experimentation on different definitions of actions. Given the impact this definition can have on the performance of a model, an interesting topic of investigation would be which are the actions we should focus on learning.

There is a large variety of challenging and interesting tasks to explore within the action recognition area, especially when considering the requirements of real-world applications. Although being a relatively recent area, the growing research interest has already led to important advances. Hopefully, it will not take long until the community can benefit from action recognition applications.

References

[1] REN, S., HE, K., GIRSHICK, R. B., et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[2] LALONDE, R., ZHANG, D., SHAH, M. "ClusterNet: Detecting Small Objects in Large Scenes by Exploiting Spatio-Temporal Information". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[3] SONG, L., GONG, D., LI, Z., et al. "Occlusion Robust Face Recognition Based on Mask Learning With Pairwise Differential Siamese Network". In: *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[4] TULSIANI, S., GUPTA, S., FOUHEY, D. F., et al. "Factoring Shape, Pose, and Layout From the 2D Image of a 3D Scene". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[5] JI, P., LI, H., DAI, Y., et al. ""Maximizing Rigidity" Revisited: A Convex Programming Approach for Generic 3D Shape Reconstruction From Multiple Perspective Views". In: *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[6] LEE, J.-K., YEA, J., PARK, M.-G., et al. "Joint Layout Estimation and Global Multi-View Registration for Indoor Reconstruction". In: *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[7] JACKSON, A. S., BULAT, A., ARGYRIOU, V., et al. "Large Pose 3D Face Reconstruction From a Single Image via Direct Volumetric CNN Regression". In: *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[8] YANG, Y., LOQUERCIO, A., SCARAMUZZA, D., et al. "Unsupervised Moving Object Detection via Contextual Information Separation". In: *The IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[9] CHEN, C., CHEN, Q., DO, M. N., et al. "Seeing Motion in the Dark". In: *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[10] GAO, R., GRAUMAN, K. "On-Demand Learning for Deep Image Restoration". In: *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[11] LIU, X., SUGANUMA, M., SUN, Z., et al. "Dual Residual Networks Leveraging the Potential of Paired Operations for Image Restoration". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[12] FEICHTENHOFER, C., FAN, H., MALIK, J., et al. "SlowFast Networks for Video Recognition". In: *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2019.

[13] WU, C.-Y., FEICHTENHOFER, C., FAN, H., et al. "Long-Term Feature Banks for Detailed Video Understanding". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[14] SZEGEDY, C., LIU, W., JIA, Y., et al. "Going Deeper with Convolutions", *CoRR*, v. abs/1409.4842.

[15] HE, K., ZHANG, X., REN, S., et al. "Deep Residual Learning for Image Recognition". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[16] CARREIRA, J., ZISSERMAN, A. "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[17] OTSU, N. "A Threshold Selection Method from Gray-Level Histograms", v. 9, pp. 62–66, 01 1979.

[18] HARALICK, R. M., SHANMUGAM, K., DINSTEIN, I. "Textural Features for Image Classification", *IEEE Transactions on Systems, Man, and Cybernetics*, v. 3, n. 6, pp. 610–621, 1973.

[19] KARPATHY, A., TODERICI, G., SHETTY, S., et al. "Large-scale Video Classification with Convolutional Neural Networks". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[20] JI, S., XU, W., YANG, M., et al. "3D Convolutional Neural Networks for Human Action Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Jan 2013.

[21] LAPTEV, I., MARSZALEK, M., SCHMID, C., et al. "Learning realistic human actions from movies". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[22] XUE, D., SAYANA, A., DARKE, E., et al. "Vision-Based Gait Analysis for Senior Care", *CoRR*, v. abs/1812.00169, 2018.

[23] YAO, T., LIU, J. "Human Behavior Understanding: From Action Recognition to Complex Event Detection". In: *Proceedings of the 26th ACM International Conference on Multimedia*, 2018.

[24] WANG, D., YUAN, Y., WANG, Q. "Early Action Prediction With Generative Adversarial Networks", *IEEE Access*, v. 7, pp. 35795–35804, 2019.

[25] WU, C.-Y., ZAHEER, M., HU, H., et al. "Compressed Video Action Recognition". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[26] ZHOU, Y., SUN, X., ZHA, Z.-J., et al. "MiCT: Mixed 3D/2D Convolutional Tube for Human Action Recognition". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[27] KUEHNE, H., JHUANG, H., GARROTE, E., et al. "HMDB51: A Large Video Database for Human Motion Recognition". In: *The IEEE International Conference on Computer Vision (ICCV)*, 2011.

[28] SOOMRO, K., ZAMIR, A. R., SHAH, M. "UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild", *CoRR*, v. abs/1212.0402, 2012.

[29] KAY, W., CARREIRA, J., SIMONYAN, K., et al. "The Kinetics Human Action Video Dataset", *CoRR*, v. abs/1705.06950, 2017.

[30] GOYAL, R., EBRAHIMI KAHOU, S., MICHALSKI, V., et al. "The "Something Something" Video Database for Learning and Evaluating Visual Common Sense". In: *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[31] WANG, L., XIONG, Y., WANG, Z., et al. "Temporal Segment Networks: Towards Good Practices for Deep Action Recognition". In: *The European Conference on Computer Vision (ECCV)*, 2016.

[32] FEICHTENHOFER, C., PINZ, A., ZISSERMAN, A. "Convolutional Two-Stream Network Fusion for Video Action Recognition". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[33] JIANG, Y.-G., LIU, J., ROSHAN ZAMIR, A., et al. "THUMOS Challenge: Action Recognition with a Large Number of Classes". `http://crcv.ucf.edu/THUMOS14/`, 2014.

[34] KRIZHEVSKY, A., SUTSKEVER, I., HINTON, G. E. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25 (NIPS)*, pp. 1097–1105, 2012.

[35] CABA HEILBRON, F., ESCORCIA, V., GHANEM, B., et al. "ActivityNet: A Large-Scale Video Benchmark for Human Activity Understanding". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[36] SIGURDSSON, G. A., VAROL, G., WANG, X., et al. "Hollywood in Homes: Crowdsourcing Data Collection for Activity Understanding". In: *The European Conference on Computer Vision (ECCV)*, 2016.

[37] GEEST, R. D., GAVVES, E., GHODRATI, A., et al. "Online Action Detection". In: *The European Conference on Computer Vision (ECCV)*, 2016.

[38] YEUNG, S., RUSSAKOVSKY, O., JIN, N., et al. "Every Moment Counts: Dense Detailed Labeling of Actions in Complex Videos", *International Journal of Computer Vision*, 2017.

[39] D. RODRIGUEZ, M., AHMED, J., SHAH, M. "Action MACH a spatio-temporal Maximum Average Correlation Height filter for action recognition". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[40] JHUANG, H., GALL, J., ZUFFI, S., et al. "Towards Understanding Action Recognition". In: *The IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[41] GU, C., SUN, C., ROSS, D. A., et al. "AVA: A Video Dataset of Spatio-Temporally Localized Atomic Visual Actions". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[42] SCOVANNER, P., ALI, S., SHAH, M. "A 3-dimensional Sift Descriptor and Its Application to Action Recognition". In: *Proceedings of the 15th ACM International Conference on Multimedia*, 2007.

[43] LAPTEV, LINDEBERG. "Space-time interest points". In: *The IEEE International Conference on Computer Vision (ICCV)*, 2003.

[44] KLÄSER, A., MARSZAŁEK, M., SCHMID, C. "A Spatio-Temporal Descriptor Based on 3D-Gradients". In: *Proceedings of the British Machine Vision Conference (BMVC)*, 2008.

[45] WANG, H., KLÄSER, A., SCHMID, C., et al. "Action recognition by dense trajectories". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[46] WANG, H., SCHMID, C. "Action Recognition with Improved Trajectories". In: *The IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[47] THERIAULT, C., THOME, N., CORD, M. "Dynamic Scene Classification: Learning Motion Descriptors with Slow Features Analysis". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[48] DESAI, C., RAMANAN, D., FOWLKES, C. "Discriminative models for static human-object interactions". In: *CVPR workshops*, 2010.

[49] DELAITRE, V., LAPTEV, I., SIVIC, J. "Recognizing human actions in still images: a study of bag-of-features and part-based representations." In: *Proceedings of the British Machine Vision Conference (BMVC)*, 2010.

[50] LI, L.-J., FEI-FEI, L. "What, where and who? Classifying events by scene and object recognition". In: *The IEEE International Conference on Computer Vision (ICCV)*, 2007.

[51] JHUANG, H., SERRE, T., WOLF, L., et al. "A Biologically Inspired System for Action Recognition". In: *The IEEE International Conference on Computer Vision (ICCV)*, 2007.

[52] KARPATHY, A., TODERICI, G., SHETTY, S., et al. "Large-scale Video Classification with Convolutional Neural Networks". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[53] SIMONYAN, K., ZISSERMAN, A. "Two-Stream Convolutional Networks for Action Recognition in Videos". In: *Advances in Neural Information Processing Systems 27 (NIPS)*, pp. 568–576, 2014.

[54] BROX, T., BRUHN, A., PAPENBERG, N., et al. "High Accuracy Optical Flow Estimation Based on a Theory for Warping". In: *The European Conference on Computer Vision (ECCV)*, 2004.

[55] WANG, L., XIONG, Y., WANG, Z., et al. "Towards Good Practices for Very Deep Two-Stream ConvNets", *CoRR*, v. abs/1507.02159, 2015.

[56] ZHANG, B., WANG, L., WANG, Z., et al. "Real-Time Action Recognition With Enhanced Motion Vector CNNs". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[57] TRAN, D., BOURDEV, L., FERGUS, R., et al. "Learning Spatiotemporal Features With 3D Convolutional Networks". In: *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[58] DIBA, A., FAYYAZ, M., SHARMA, V., et al. "Temporal 3D ConvNets Using Temporal Transition Layer". In: *CVPR Workshops*, 2018.

[59] DONAHUE, J., HENDRICKS, L. A., GUADARRAMA, S., et al. "Long-term Recurrent Convolutional Networks for Visual Recognition and Description", *CoRR*, v. abs/1411.4389, 2014.

[60] MA, S., SIGAL, L., SCLAROFF, S. "Learning Activity Progression in LSTMs for Activity Detection and Early Detection". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[61] SADEGH ALIAKBARIAN, M., SADAT SALEH, F., SALZMANN, M., et al. "Encouraging LSTMs to Anticipate Actions Very Early". In: *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[62] SIMONYAN, K., ZISSERMAN, A. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *International Conference on Learning Representations (ICLR)*, 2015.

[63] FEICHTENHOFER, C., PINZ, A., WILDES, R. "Spatiotemporal Residual Networks for Video Action Recognition". In: *Advances in Neural Information Processing Systems 29 (NIPS)*, pp. 3468–3476, 2016.

[64] ZHU, Y., LAN, Z., NEWSAM, S. D., et al. "Hidden Two-Stream Convolutional Networks for Action Recognition", *CoRR*, v. abs/1704.00389, 2017.

[65] HUANG, G., LIU, Z., VAN DER MAATEN, L., et al. "Densely Connected Convolutional Networks". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[66] XIE, S., SUN, C., HUANG, J., et al. "Rethinking Spatiotemporal Feature Learning: Speed-Accuracy Trade-offs in Video Classification". In: *The European Conference on Computer Vision (ECCV)*, September 2018.

[67] QIU, Z., YAO, T., MEI, T. "Learning Spatio-Temporal Representation With Pseudo-3D Residual Networks". In: *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[68] TRAN, D., WANG, H., TORRESANI, L., et al. "A Closer Look at Spatiotemporal Convolutions for Action Recognition". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[69] ZHAO, Y., XIONG, Y., LIN, D. "Trajectory Convolution for Action Recognition". In: *Advances in Neural Information Processing Systems 31 (NIPS)*, pp. 2208–2219, 2018.

[70] WANG, X., GIRSHICK, R., GUPTA, A., et al. "Non-Local Neural Networks". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[71] BUADES, A., COLL, B., MOREL, J. . "A non-local algorithm for image denoising". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2005.

[72] SIGURDSSON, G. A., RUSSAKOVSKY, O., GUPTA, A. "What Actions Are Needed for Understanding Human Actions in Videos?" In: *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[73] ALWASSEL, H., CABA HEILBRON, F., ESCORCIA, V., et al. "Diagnosing Error in Temporal Action Detectors". In: *The European Conference on Computer Vision (ECCV)*, September 2018.

[74] SHOU, Z., CHAN, J., ZAREIAN, A., et al. "CDC: Convolutional-De-Convolutional Networks for Precise Temporal Action Localization in Untrimmed Videos". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[75] XU, H., DAS, A., SAENKO, K. "R-C3D: Region Convolutional 3D Network for Temporal Activity Detection". In: *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[76] CHAO, Y.-W., VIJAYANARASIMHAN, S., SEYBOLD, B., et al. "Rethinking the Faster R-CNN Architecture for Temporal Action Localization". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[77] BAHDANAU, D., CHO, K., BENGIO, Y. "Neural Machine Translation by Jointly Learning to Align and Translate", *CoRR*, v. abs/1409.0473, 2014.

[78] GKIOXARI, G., MALIK, J. "Finding Action Tubes". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[79] UIJLINGS, J. R. R., VAN DE SANDE, K. E. A., GEVERS, T., et al. "Selective Search for Object Recognition", 2013.

[80] PENG, X., SCHMID, C. "Multi-region Two-Stream R-CNN for Action Detection". In: *The European Conference on Computer Vision (ECCV)*, 2016.

[81] ZITNICK, C. L., DOLLÁR, P. "Edge Boxes: Locating Object Proposals from Edges". In: *The European Conference on Computer Vision (ECCV)*, 2014.

[82] KALOGEITON, V., WEINZAEPFEL, P., FERRARI, V., et al. "Action Tubelet Detector for Spatio-Temporal Action Localization". In: *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[83] LIU, W., ANGUELOV, D., ERHAN, D., et al. "SSD: Single Shot MultiBox Detector", *The European Conference on Computer Vision (ECCV)*, 2016.

[84] SINGH, G., SAHA, S., SAPIENZA, M., et al. "Online Real-Time Multiple Spatiotemporal Action Localisation and Prediction". In: *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[85] ZHOU, B., KHOSLA, A., LAPEDRIZA, A., et al. "Learning Deep Features for Discriminative Localization". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[86] SHOU, Z., PAN, J., CHAN, J., et al. "Online Detection of Action Start in Untrimmed, Streaming Videos". In: *The European Conference on Computer Vision (ECCV)*, September 2018.

[87] GOODFELLOW, I., POUGET-ABADIE, J., MIRZA, M., et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27 (NIPS)*, pp. 2672–2680, 2014.

[88] GAO, J., YANG, Z., NEVATIA, R. "RED: Reinforced Encoder-Decoder Networks for Action Anticipation", *Proceedings of the British Machine Vision Conference (BMVC)*, 2017.

[89] VONDRICK, C., PIRSIAVASH, H., TORRALBA, A. "Anticipating Visual Representations From Unlabeled Video". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[90] HOIEM, D., CHODPATHUMWAN, Y., DAI, Q. "Diagnosing Error in Object Detectors". In: *The European Conference on Computer Vision (ECCV)*, 2012.

[91] JENI, L., COHN, J., DE LA TORRE, F. "Facing Imbalanced Data - Recommendations for the Use of Performance Metrics". v. 2013, 09 2013. doi: 10.1109/ACII.2013.47.

[92] PASZKE, A., GROSS, S., CHINTALA, S., et al. "Automatic differentiation in PyTorch". 2017.

[93] MARTINEZ, B., MODOLO, D., XIONG, Y., et al. "Action Recognition With Spatial-Temporal Discriminative Filter Banks". In: *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[94] SHAN, C., ZHANG, Z., CHEN, Z. "A Coarse-to-Fine Framework for Learned Color Enhancement with Non-Local Attention". In: *2019 IEEE International Conference on Image Processing (ICIP)*, Sep. 2019.

[95] "Online Action GitHub repository". `https://github.com/patykov/OnlineAction`. Accessed: 2020-02-10.

[96] VASWANI, A., SHAZEER, N., PARMAR, N., et al. "Attention is all you need". In: *Advances in neural information processing systems (NIPS)*, pp. 5998–6008, 2017.

[97] "Non-local networks GitHub repository". `https://github.com/facebookresearch/video-nonlocal-net`. Accessed: 2020-02-10.

[98] "Caffe2 framework". `https://caffe2.ai/`. Accessed: 2020-02-10.

[99] SUTSKEVER, I., MARTENS, J., DAHL, G., et al. "On the importance of initialization and momentum in deep learning". pp. 1139–1147, 01 2013.

[100] RODRIGUEZ, C., FERNANDO, B., LI, H. "Action Anticipation by Predicting Future Dynamic Images". In: *Computer Vision – ECCV 2018 Workshops*, 2019.

[101] "Charades Challenge website". `http://vuchallenge.org/charades.html`. Accessed: 2020-02-10.

[102] WANG, X., GUPTA, A. "Videos as Space-Time Region Graphs". In: Ferrari, V., Hebert, M., Sminchisescu, C., et al. (Eds.), *The European Conference on Computer Vision (ECCV)*, 2018.