Universidade Estadual de Campinas
Instituto de Computação

Helena de Almeida Maia

Visual Rhythm-based Convolutional Neural Networks and Adaptive Fusion for a Multi-stream Architecture Applied to Human Action Recognition

Redes Neurais Convolucionais Baseadas em Ritmos Visuais e Fusão Adaptativa para uma Arquitetura de Múltiplos Canais Aplicada ao Reconhecimento de Ações Humanas

CAMPINAS
2020

Helena de Almeida Maia

Visual Rhythm-based Convolutional Neural Networks and Adaptive Fusion for a Multi-stream Architecture Applied to Human Action Recognition

Redes Neurais Convolucionais Baseadas em Ritmos Visuais e Fusão Adaptativa para uma Arquitetura de Múltiplos Canais Aplicada ao Reconhecimento de Ações Humanas

Tese apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Doutora em Ciência da Computação.

Thesis presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Supervisor/Orientador: Prof. Dr. Hélio Pedrini
Co-supervisor/Coorientador: Prof. Dr. Marcelo Bernardes Vieira

Este exemplar corresponde à versão final da Tese defendida por Helena de Almeida Maia e orientada pelo Prof. Dr. Hélio Pedrini.

CAMPINAS
2020

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

M28v

Maia, Helena de Almeida, 1992-
Visual rhythm-based convolutional neural networks and adaptive fusion for a multi-stream architecture applied to human action recognition / Helena de Almeida Maia. – Campinas, SP : [s.n.], 2020.

Orientador: Hélio Pedrini.
Coorientador: Marcelo Bernardes Vieira.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Visão por computador. 2. Aprendizado de máquina. 3. Redes neurais convolucionais. I. Pedrini, Hélio, 1963-. II. Vieira, Marcelo Bernardes. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Redes neurais convolucionais baseadas em ritmos visuais e fusão adaptativa para uma arquitetura de múltiplos canais aplicada ao reconhecimento de ações humanas
Palavras-chave em inglês:
Computer vision
Machine learning
Convolutional neural networks
Área de concentração: Ciência da Computação
Titulação: Doutora em Ciência da Computação
Banca examinadora:
Hélio Pedrini [Orientador]
Rodrigo Luis de Souza da Silva
Tiago José de Carvalho
Esther Luna Colombini
Tiago Fernandes Tavares
Data de defesa: 27-10-2020
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-8253-9004
- Currículo Lattes do autor: http://lattes.cnpq.br/0942146941809337

Universidade Estadual de Campinas
Instituto de Computação

Helena de Almeida Maia

Visual Rhythm-based Convolutional Neural Networks and Adaptive Fusion for a Multi-stream Architecture Applied to Human Action Recognition

Redes Neurais Convolucionais Baseadas em Ritmos Visuais e Fusão Adaptativa para uma Arquitetura de Múltiplos Canais Aplicada ao Reconhecimento de Ações Humanas

Banca Examinadora:

- Prof. Dr. Hélio Pedrini
  IC/Unicamp
- Prof. Dr. Rodrigo Luis de Souza da Silva
  DCC/UFJF
- Prof. Dr. Tiago José de Carvalho
  IFSP
- Profa. Dra. Esther Luna Colombini
  IC/Unicamp
- Prof. Dr. Tiago Fernandes Tavares
  FEEC/Unicamp

A ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 27 de outubro de 2020

You can tell a lot about a person from his biography.

(Phill Dunphy, *Phil's-osophy*)

Acknowledgements

This work was partially funded by the Coordination for the Improvement of Higher Education Personnel (CAPES), Finance Code 001, and by the São Paulo Research Foundation (FAPESP), grant #2017/09160-1. I thank NVIDIA for the donation of a GPU as part of the GPU Grant Program.

I would like to thank the professors and staff from the Institute of Computing (IC) at the University of Campinas (UNICAMP). In particular, I thank my advisor, Hélio Pedrini, for guiding me during this Ph.D. stage and for being a dedicated and empathic professor. I also thank Denise, Marcus and Wilson for being so helpful with the program requirements. I thank my co-advisor from the Federal University of Juiz de Fora (UFJF), Marcelo Bernardes, for his guidance since the Master's stage and thorough proofreading of my texts.

I am thankful to my colleagues from the Laboratory of Visual Informatics (LIV) and from the Group for Computer Graphics, Image and Vision (GCG) for our collaborative projects and technical discussions, especially Marcos, with whom is always easy to collaborate.

I am very thankful to my family. My parents for the support, my sisters, with whom I talked almost every day during these five years, my lovely nieces Isadora and Elisa, and my unborn nephew Apolo. I am grateful to my grandmother Ana for being by my side when I decided to move to Campinas. I thank Aza for the privilege of being together with a scientist who provided me many rich discussions and for being so supportive during these years. I thank his mother, Vera, for encouraging me to travel to the conferences.

I thank my friends, Bel, Bruno, Carol, Dani, Laura, Leticia, Luiz, Ronaro, Sandra and Weslley, for the oldest and invaluable friendship, and Caetano for the talks and music sharing. I thank the DPG group for the gaming sessions. I thank my friends from Campinas Carol, Fabi, Luís, Mari, Otávio, Peixe and Samuka for their friendship during this time. I also thank my first friends from Campinas, my roommates from the first house.

Resumo

A grande quantidade de dados de vídeos produzidos e divulgados todos os dias torna a inspeção visual por um operador humano impraticável. No entanto, o conteúdo desses vídeos pode ser útil para várias tarefas importantes, como vigilância e monitoramento de saúde. Portanto, métodos automáticos são necessários para detectar e compreender eventos relevantes em vídeos. O problema abordado neste trabalho é o reconhecimento das ações humanas em vídeos que visa classificar a ação que está sendo realizada por um ou mais atores. A complexidade do problema e o volume de dados de vídeo sugerem o uso de técnicas baseadas em aprendizado profundo, no entanto, ao contrário de problemas relacionados a imagens, não há uma grande variedade de arquiteturas específicas bem estabelecidas nem conjuntos de dados anotados tão grandes quanto aqueles baseados em imagens. Para contornar essas limitações, propomos e analisamos uma arquitetura de múltiplos canais composta de redes baseadas em imagens pré-treinadas na base ImageNet. Diferentes representações de imagens são extraídas dos vídeos que servem como entrada para os canais, a fim de fornecer informações complementares para o sistema. Neste trabalho, propomos novos canais baseados em ritmo visual que codificam informações de mais longo prazo quando comparados a quadros estáticos e fluxo óptico. Tão importante quanto a definição de aspectos representativos e complementares é a escolha de métodos de combinação adequados que explorem os pontos fortes de cada modalidade. Assim, nós também analisamos diferentes abordagens de fusão para combinar as modalidades. Para definir os melhores parâmetros de nossos métodos de fusão usando o conjunto de treinamento, temos que reduzir o sobreajuste em modalidades individuais, caso contrário, as saídas 100% precisas não ofereceriam uma representação realista e relevante para o método de fusão. Assim, investigamos uma técnica de parada precoce para treinar redes individuais. Além de reduzir o sobreajuste, esse método também reduz o custo de treinamento, pois normalmente requer menos épocas para concluir o processo de classificação, e se adapta a novos canais e conjuntos de dados graças aos seus parâmetros treináveis. Os experimentos são realizados nos conjuntos de dados UCF101 e HMDB51, que são duas bases desafiadoras no contexto de reconhecimento de ações.

Abstract

The large amount of video data produced and released every day makes visual inspection by a human operator impracticable. However, the content of these videos can be useful for various important tasks, such as surveillance and health monitoring. Therefore, automatic methods are needed to detect and understand relevant events in videos. The problem addressed in this work is the recognition of human actions in videos that aims to classify the action that is being performed by one or more actors. The complexity of the problem and the volume of video data suggest the use of deep learning-based techniques, however, unlike image-related problems, there is neither a great variety of specific well-established architectures nor annotated datasets as large as image-based ones. To circumvent these limitations, we propose and analyze a multi-stream architecture containing image-based networks pre-trained on the large ImageNet. Different image representations are extracted from the videos to feed the streams, in order to provide complementary information for the system. Here, we propose new streams based on visual rhythm that encode longer-term information when compared to still frames and optical flow. As important as the definition of representative and complementary aspects is the choice of proper combination methods that explore the strengths of each modality. Thus, here we also analyze different fusion approaches to combine the modalities. In order to define the best parameters of our fusion methods using the training set, we have to reduce overfitting in individual modalities, otherwise, the 100%-accurate outputs would not offer a realistic and relevant representation for the fusion method. Thus, we investigate an early stopping technique to train individual networks. In addition to reducing overfitting, this method also reduces the training cost, since it usually requires fewer epochs to complete the classification process, and adapts to new streams and datasets thanks to its trainable parameters. Experiments are conducted on UCF101 and HMDB51 datasets, which are two challenging benchmarks in the context of action recognition.

List of Figures

List of Tables

List of Abbreviations and Acronyms

1D	One-dimensional
2D	Two-dimensional
3D	Three-dimensional
ATUS	American Time Use Survey
AVA	Atomic Visual Actions
AVR	Adaptive Visual Rhythm
BoF	Bag of Features
C-SVC	C-Support Vector Classification
C3D	Three-dimensional Convolutional Network
CNN	Convolutional Neural Network
DT	Dense Trajectories
DTPP	Deep networks with Temporal Pyramid Pooling
FC	Fully Connected
FV	Fisher-Vectors
GBH	Gradient Boundary Histograms
HAR	Human Action Recognition
HC	HandCrafted
HMDB	Human Motion DataBase
HOF	Histogram of Optical Flow
HOG	Histogram of Oriented Gradients
I3D	Inflated 3D
iDT	improved Dense Trajectories
LBP	Local Binary Pattern
LSTM	Long Short-Term Memory
LVR	Learnable Visual Rhythm
MBH	Motion Boundary Histogram
MBP	Motion Binary Pattern
ML	Machine Learning
MOMP	Motion of Oriented Magnitude Pattern

OF	Optical Flow
OvA	One-versus-All
OvO	One-versus-One
OvR	One-versus-Rest
PCA	Principal Component Analysis
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RGB	Red-Green-Blue
RGB-D	Red-Green-Blue-Depth
RoI	Regions of Interest
SGD	Stochastic Gradient Descent
STIP	Space-Time Interest Point
SVM	Support Vector Machine
TSN	Temporal Segment Networks
TV	Total Variation
VLAD	Vector of Locally Aggregated Descriptors
VLBP	Volume Local Binary Pattern
VR	Visual Rhythm
VRTD	Visual Rhythm Trajectory Descriptor
XOR	eXclusive OR

Contents

Chapter 1

Introduction

In this chapter, we introduce the target problem and define the scope of our work. We also present our main objectives, contributions and research questions. We conclude the chapter with a list of publications and an outline of the remaining text.

1.1 Problem and Motivation

Over the past few years, a large amount of video data has been produced and released due to the easy access to both devices for capturing new data such as video cameras and mobiles, and streaming platforms such as YouTube for sharing. According to data from 2019 [119], YouTube alone is responsible for an average of 500 hours of videos being uploaded per minute, over twenty times higher than 2009. Since the analysis of this large amount of data by human operators may be stressful and may involve sensitive content, automatic procedures are needed to address related problems.

The problem addressed in this work is the recognition of human actions in videos (HAR - Human Action Recognition) that aims to classify the action being performed by one or more actors based on a set of learned classes. The understanding of human activity has a wide range of applications, particularly due to the availability of digital cameras in environments such as airports, train, stadiums, hospitals, among others. Some applications, illustrated in Figure 1.1, are listed as follows:

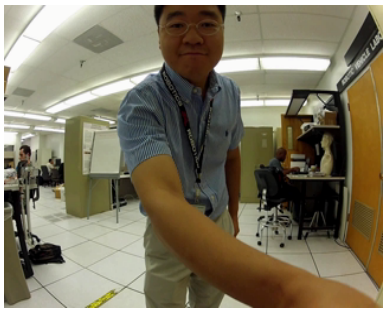
- Intelligent surveillance systems [46, 120]: use videos from security cameras to detect unusual and undesired events, such as accidents or crimes. Analysis of the activities in the video can help distinguish between normal and anomalous events, and potentially classify the type of anomaly.

- Human-computer interaction systems [36, 102]: must understand the interaction of the users with itself to perform a specific procedure or provide some information, such as directions or recommendations. Recognition of the users' actions may be crucial in determining the most appropriate response to the situation.

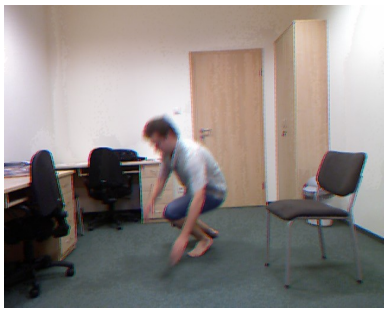
- Smart home security [3, 57, 63, 115]: allows the users to monitor and control their houses through devices connected to the Internet. In particular, these systems can be used for elderly healthcare monitoring to detect falls, reducing the interval

between the occurrence and help arrival. Once again, the classification of normal and anomalous events is the key to this type of system.

Intelligent surveillance [120] - shooting



Human-computer interaction [102]



Fall detection [57]

Figure 1.1: Examples of action recognition applications.

Similar to other video-based problems, the recognition of human actions faces some challenges related to difficult scene conditions, such as occlusions, background clutter, camera motion and lighting changes, which affect how the actions are seen in the video (Figure 1.2).

Figure 1.2: Examples of the "throw" class from HMDB51 dataset that present one or more of the following challenges: cluttered background, severe camera motion, poor lighting condition and poor clip quality.

The HAR problem also has three specific challenges: (i) the complexity of the action concept, (ii) dealing with intra and inter-class variations and (iii) understanding person-person and person-object interactions. These challenges are further detailed as follows:

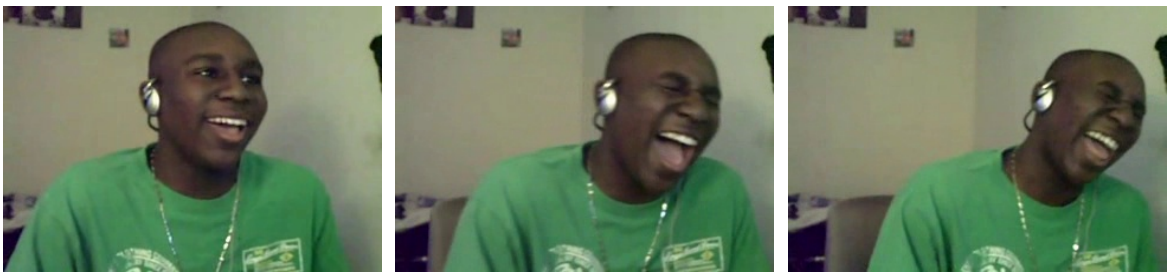
(i) Complex concept: As we will discuss in Sections 1.2 and 2.1, the action definition is very complex and may depend on the context. In fact, the definition seems to gain a new perspective as new datasets are proposed and it is quite common to find literature references that dedicate a section to define what is meant by the term "action", even in recent HAR surveys [42, 75, 94]. This abstract concept hampers the definition of standard datasets for evaluation, such as the ImageNet for image classification. As a consequence, it also difficults the generalization of findings.

(ii) Variations of the same action $\times$ very similar classes: The same action can be performed at different paces and ways by different actors. On the other hand, there

are distinct classes that are difficult to differentiate, such as “laugh” and “smile” (Figure 1.3). Therefore, the method must be able to discern what is an intra-class variation and what is a distinct action.

“smile”

“laugh”



“laugh”

Figure 1.3: Examples of the “smile” and “laugh” classes from HMDB51. The first example shows a broad smile with head motion. The second one depicts a soft laughing, whereas the third one shows a hard laughing involving more movement. Note that grinning and soft laughing are difficult to differentiate.

(iii) Understanding interactions: Some actions are characterized by the involvement of two or more actors, or objects, such as “MilitaryParade” and “PlayingGuitar”. The method has to identify, even if implicitly, that there is an interaction between the actors and objects. The presence of the skateboard in the scene, for instance, does not imply in a skateboarding action, as we can see in Figure 1.4. In fact, the depicted example corresponds to the “throw” class.

1.2 Scope and Delimitation

An action can be defined as a sequence of movements performed to accomplish a specific purpose. Nonetheless, it is still a quite vague definition, leaving room for various inter-

Figure 1.4: Example of the "throw" class. Note that, although there is a skateboard in the scene, the actor is not interacting with it.

pretations. Therefore, an initial delimitation of what is considered an action in this work and a consequent delimitation of our target problem are of paramount importance.

First, the spatial and temporal location of the action in the video are not part of our scope. Our recognition is based only on RGB information, without depth (RGB-D) or audio. We only consider trimmed and single-label videos and classify the entire scene without any spatial segmentation. In addition to the action definition, another aspect that diverges among different researches is the granularity of the action, that is, whether it is as primitive as "raising the left arm" or as complex and composed as "playing soccer". This granularity is reflected in the average duration of the actions, although some of them can be recognized in shorter time. Here, we consider a coarser granularity with actions that take a few tens of seconds to be recognized.

We work with uncontrolled and realistic scenarios. As a consequence, actions may include interactions with other actors or objects, and cluttered backgrounds. There is no explicit limit to the number of actors involved, however, this research is not focused on crowds. Both the involved actors/objects and the elements of the scene may be clues to recognize the action. They compose the so-called scene structure. A more-in-depth discussion regarding the definition of actions will be presented in the following chapters.

1.3 Objectives and Contributions

Deep networks have been widely explored for the action recognition problem in videos. However, the higher cost of video-based deep networks and the absence of datasets as large as image-based ones have led the researchers to explore image networks for the problem.

Following this trend, we propose a new image-based network inspired by the two-stream architecture [113]. This architecture is composed of two parallel deep networks working with different image modalities. Its central idea is to explore the strengths of the image modalities by combining their respective stream decisions. Thus, our main objective is to provide complementary information for both streams in order to capture new aspects of the actions. As specific goals, we can mention (i) the evaluation of visual rhythm approaches in the multi-stream framework, (ii) the analysis and reduction of overfitting in the streams and (iii) the evaluation of the overfitting impact on trainable

fusions.

Our main contributions are listed as follows:

1. **Multi-stream framework based on visual rhythm:** The two-stream architecture is based on appearance and motion, represented by a single RGB frame and 10 optical flow images, respectively. The combination of both improves the overall performance. Our proposal introduces the Visual Rhythm (VR) modality into the architecture, which consists of a compact 2D representation of the video constructed by the concatenation of features extracted from each frame. With this new modality, we aim to provide longer-term information to the network. In contrast to RGB and optical flow, the VR represents the entire video in a single image. Since the VR has no temporal limit, very long videos result in equally wide VRs, generating a memory bottleneck. However, since we are working with trimmed videos of a few tens of seconds, it is possible to represent them with appropriately-sized VR images.

2. **A novel method to construct visual rhythms:** VR enables monitoring the development of a particular feature throughout the time. The key for an adequate VR is the choice of a proper feature according to the target problem. Generally, these features are extracted in a handcrafted fashion, such as the diagonal or a color histogram of each frame. Here, we propose a novel method called Learnable Visual Rhythm (LVR) to extract frame-level features for the VR. It consists of a method based on Convolutional Neural Networks (CNNs), which is able to capture complex patterns in the frames.

3. **Extensive study on individual performances and complementarity:** We carry out a detailed analysis on the individual behavior of the streams regarding the action classes, showing their best/worst cases and the most frequent confusions. We also analyze every possible combination of streams by exploring diversity measures. Although these measures are not used to support structural decisions, it provides insights about the complementarity of the streams.

4. **Adaptive fusion:** For our multi-stream architecture, we explore fusion methods with trainable parameters. The major goal of this strategy is to adapt the fusion to different datasets and combinations of streams. This adaptive fusion learns the best weights for each stream based on the training set. Thus, it benefits from the modality that is most suitable for a specific scenario. The overfitting of the streams may be an obstacle to the adaptive fusion, as there would be a great disparity between the training and test inputs that it has to combine. As such, we assess an early stopping protocol for the streams to reduce the effect of overfitting.

1.4 Research Questions

Our central hypothesis is that a 2D deep network can be used to find patterns on an image with heterogeneous dimensions, the visual rhythm. We guide our research by means of the following investigative questions:

Q1. Can visual rhythm, as a longer-term feature, provide complementary information to characterize the action in a multi-stream scenario?

Q2. Does the temporal concatenation of the outputs generated by an image CNN (Learnable VR) present relevant patterns to characterize an action?

Q3. Can the Learnable VR outperform our baseline Adaptive VR?

Q4. Can the early stopping protocol reduce the overfitting on individual streams and improve the fusion performance?

1.5 Publications

The following papers were published since the beginning of our research. They are ordered by the publication year. Papers 1 and 4 are further detailed in Chapter 5 and paper 3 in Chapter 6.

1. D. T. Concha, H. A. Maia, H. Pedrini, H. Tacon, A. S. Brito, H. L. Chaves, and M. B. Vieira. Multi-Stream Convolutional Neural Networks for Action Recognition in Video Sequences Based on Adaptive Visual Rhythms. In *IEEE International Conference on Machine Learning and Applications*, Orlando, FL, USA, Dec. 2018. IEEE.

2. H. Tacon, A. S. Brito, H. L. Chaves, M. B. Vieira, S. M. Villela, H. Almeida Maia, D. T. Concha, and H. Pedrini. Human Action Recognition Using Convolutional Neural Networks with Symmetric Time Extension of Visual Rhythms. In *International Conference on Computational Science and Its Applications*, pages 351–366. Springer, 2019

3. H. A. Maia, M. R. Souza, A. Santos, H. Pedrini, H. Tacon, A. S. Brito, H. L. Chaves, M. B. Vieira, and S. M. Villela. Learnable Visual Rhythms Based on the Stacking of Convolutional Neural Networks for Action Recognition. In *IEEE International Conference on Machine Learning and Applications*, pages 1794–1799, Boca Raton, FL, USA, Dec. 2019. IEEE.

4. H. A. Maia, D. T. Concha, H. Pedrini, H. Tacon, A. S. Brito, H. L. Chaves, M. B. Vieira, and S. M. Villela. Action Recognition in Videos Using Multi-Stream Convolutional Neural Networks. In *Deep Learning Applications*, pages 95–111. Springer, 2020.

5. A. C. S. Santos, H. A. Maia, M. R. Souza, M. B. Vieira, and H. Pedrini. Fuzzy Fusion for Two-stream Action Recognition. In *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, volume 5, pages 117–123. INSTICC, SciTePress, 2020.

6. H. Tacon, A. S. Brito, H. L. Chaves, M. B. Vieira, S. M. Villela, H. Maia, D. T. Concha, and H. Pedrini. Multi-stream Architecture with Symmetric Extended Visual Rhythms for Deep Learning Human Action Recognition. In *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, volume 5, pages 351–358. INSTICC, SciTePress, 2020.

7. H. L. Chaves, K. S. Ribeiro, A. S. Brito, H. Tacon, M. B. Vieira, A. S. Cerqueira, S. M. Villela, H. Maia, D. T. Concha, and H. Pedrini. Filter Learning from Deep Descriptors of a Fully Convolutional Siamese Network for Tracking in Videos. In *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, volume 4, pages 685–694. INSTICC, SciTePress, 2020.

8. M. R. Souza, H. A. Maia, M. B. Vieira, and H. Pedrini. Survey on Visual Rhythms: A Spatio-Temporal Representation for Video Sequences. *Neurocomputing*, 402:409–422, 2020.

1.6 Text Organization

In this chapter, we introduced the problem of human action recognition in videos addressed in this work. The remaining text is organized as follows. In Chapter 2, we present the main fundamentals of our work. The related approaches and benchmarks available in the literature are presented in Chapter 3 and 4, respectively. In Chapter 5, we present our first collaborative method, the Adaptive Visual Rhythm (AVR). This chapter also includes the description of our general framework. In Chapter 6, we present the Learnable Visual Rhythm (LVR), an evolution of the aforementioned AVR. We show some analyses in Chapter 7 along with adaptable fusion methods for our multi-stream network. Finally, Chapter 8 presents the concluding remarks and potential directions for future research. Appendix A contains supplementary material for Section 2.9, whereas Appendix B contains reference lists for Chapters 4 and 7.

Chapter 2

Fundamentals

This chapter presents fundamental concepts of our work. In Section 2.1, we give further details about our main concept (action), including some definitions found in the literature. Sections 2.2 and 2.3 describe two features that are used in our neural network architecture. We present the Support Vector Machine classifier in Section 2.4 which is explored in the fusion stage of the proposed method. Sections 2.5 to 2.8 contain deep learning-related concepts and Section 2.9 introduces some metrics to evaluate the quality of the ensemble methods.

2.1 Action

Action is a complex concept and its definition varies among the works in the literature. A human action involves the movement of a single or multiple parts of the body to accomplish a simple or complex goal, possibly with an object or person interaction.

Some works consider a hierarchical definition of actions according to their complexity. Moeslund et al. [75] and Poppe [94] proposed a hierarchy in three groups: (i) action primitive, (ii) action and (iii) activities. Action primitives are atomic movements, such as limb-level motion. An action is composed of a sequence of primitives and can involve the entire body. Activity is a complex event formed by several actions and can include interactions and be dependent on the environmental context. However, there is not a clear delimitation among these groups. For instance, Moeslund et al. considered "run left" as an action primitive, whereas Poppe classified "running" as an action. Moeslund et al. argued that the granularity of the primitives depends on the application.

Aggarwal and Ryoo [1] proposed a hierarchy of four levels for action categorization: (i) gestures (ii) actions (iii) interactions and (iv) group activities. Similar to action primitives, gestures are elementary limb-level movements, such as "stretching an arm". The action category matches the Moeslund et al.'s definition. Interactions are actions with two or more agents and/or objects, such as "a person stealing a suitcase from another", and group activities involve conceptual groups performing an activity, such as "a group of people marching". Again, the boundaries of these groups are not well-defined, in particular, the same action can fall under (iii) or (iv) according to the chosen description. For instance, "three people having lunch" can be interpreted as a three-person interaction, whereas "a

group of people having lunch" can be seen as a group activity, although both represent the same event.

When proposing new datasets, the authors should also be concerned with the action definition. Caba et al. [29] argued that simply elaborate an activity list based on WordNet [74] verbs would generate many ambiguities and polysemism, much more than the noun-object mapping made for the ImageNet [101]. For instance, the verb "play" can be used for instruments, sports or games. To collect a large number of activity classes, they built the ActivityNet dataset upon the American Time Use Survey (ATUS [85]) taxonomy, which contains a large and hierarchical list of daily human activities. For the construction of the Atomic Visual Actions dataset (AVA [38]), on the other hand, Gu et al. argued that listing all important high-level activities would be impractical. For this reason, they focused on atomic and primitive actions with a finer temporal granularity (about 3 seconds). Unlike the aforementioned hierarchical definitions, they considered person-object and person-person interactions in their atomic classes.

In the absence of a precise action definition, here we follow a data-driven one, similar to Santos [106]. That is, we considered all the classes available in the evaluated datasets, even if they include actions from different complexity levels and temporal granularity, such as "eat" and "FieldHockeyPenalty".

2.2 Optical Flow

Optical flow consists of a vector field representing the pixel-level motion between two frames, caused by object or camera movement. In general, the vectors are obtained by the minimization of an error function based on pixel brightness/color [124]. In our work, we use the pyramidal Lucas-Kanade and TV-L^1 methods that are presented as follows.

The Lucas-Kanade method [10,69] generates a sparse field for a pair of images through the comparison of small patches. The error function is based on the L^2-norm and is optimized using a gradient descent method. It assumes that the displacement within a patch is small and homogeneous. Considering a pair of $h \times w$ images I and J, a given pixel $p = (y, x)$ and a $h' \times w'$ patch P centered on p, the method determines the displacement vector $\vec{d} = (d_y, d_x)$ with respect to p through the minimization of the error function defined in Equation 2.1. In the equation, q represents a pixel inside the patch and $I(\cdot)$ and $J(\cdot)$ represent the pixel intensity or color in each image.

$$e(\vec{d}) = \sum_{q \in P} \left[I(q) - J(q + \vec{d}) \right]^2. \tag{2.1}$$

The pyramidal version [10] computes the optical flow for different resolutions of the image, from the lowest level of the pyramid (highest resolution) to the highest one (lowest resolution). The final displacement vector $\vec{d}$ is obtained by the weighted combination of each level output d^L:

$$\vec{d} = \sum_{L=0}^{L_m} 2^L \vec{d}^L, \tag{2.2}$$

where the index L_m corresponds to the highest level. By maintaining the original resolution, it can capture more details. The insertion of lower resolutions allows the method to deal with large motion displacements and to be robust to noise.

TV-L^1 [151] is a variational method that computes dense optical flow fields. In variational approaches, the minimization considers the entire image instead of isolated patches, assuming a smooth vector field [43]. In addition to the error criterion, the TV-L^1 method includes a Total Variation (TV) regularization, both based on the L^1-norm, being robust to illumination changes, occlusions and noise. The method estimates u that minimizes the following function:

$$\min_u \left\{ \int_\Omega \left[\lambda \left| I(x) - J(x + u(x)) \right| + |\nabla u| \right] \mathrm{d}x \right\} \tag{2.3}$$

where Ω is the image domain and $u : \Omega \to \mathbb{R}^2$ is a mapping function from pixels to displacement vectors. The term $|I(x) - J(x + u(x))|$ represents the data penalty similar to the Lucas-Kanade method, whereas the term $|\nabla u|$ corresponds to the TV regularization. The parameter λ weights between penalty and regularization.

The horizontal and vertical components of the displacement vectors can be mapped into a pair of grayscale images, representing the estimated motion for each direction. Figure 2.1 shows examples of optical flow images extracted from a pair of consecutive frames using the TV-L^1 method.

(a) Frame i (b) Frame $i + 1$ (c) Horizontal flow (d) Vertical flow

Figure 2.1: Pair of consecutive frames from an HMDB51 video of the class "pushup" and corresponding optical flow images using the TV-L^1 method.

2.3 Visual Rhythm

The visual rhythm [51,118] is a feature that encodes the entire video in a single image. The resulting image is obtained by the concatenation of subsampled pixels or a predefined 1D feature computed from each frame, which are called slices. By choosing proper slices, the visual rhythm can contain rich information to detect and classify events in video. In the literature, this feature is also referred to as temporal/spatio-temporal slices [84,89,99,153], epipolar-plane image [9], or video tomography [2].

The formal definition used in our work is given by Maia et al. [70]. Let $V = \{F_1, F_2, \cdots, F_t\}$ be a video with t frames F_i, where each frame is an $h \times w$ matrix. Consider $T(F_i) = S_i$ an operation that maps a frame F_i into an $n \times 1$ column vector S_i,

either by subsampling or computing features from F_i. The visual rhythm for the entire video V is given by the $n \times t$ matrix:

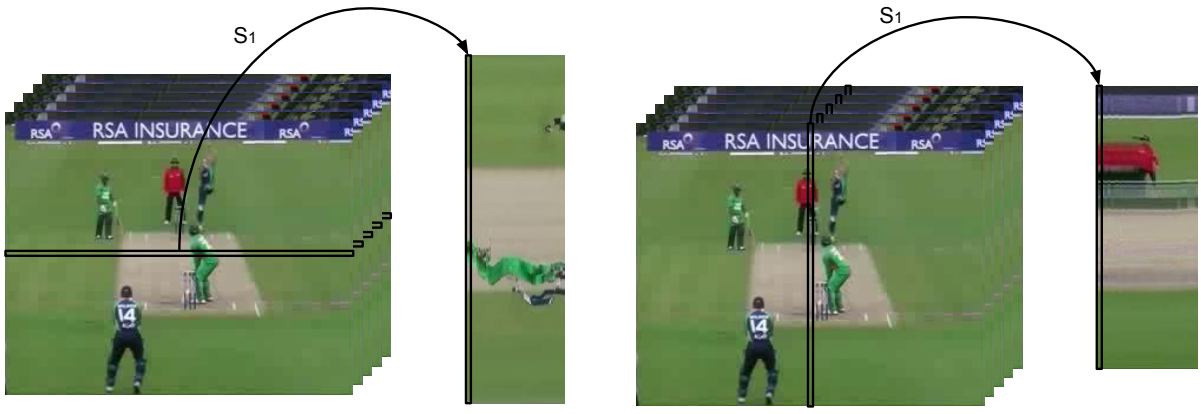
$$\mathrm{VR}(V) = [T(F_1)\ T(F_2)\ \cdots\ T(F_t)] = [S_1\ S_2\ \cdots\ S_t]. \tag{2.4}$$

Figure 2.2 shows examples of horizontal and vertical visual rhythms, where each slice corresponds to the central row/column of a frame and is placed in a column of the resulting image. The horizontal and vertical operations $T_H(\cdot)$ and $T_V(\cdot)$ are defined as follows:

$$T_H(F_i) = [F_i(a_H, 1)\ \ F_i(a_H, 2)\ \cdots\ F_i(a_H, w)]^T, \tag{2.5}$$

$$T_V(F_i) = [F_i(1, a_V)\ \ F_i(2, a_V)\ \cdots\ F_i(h, a_V)]^T, \tag{2.6}$$

where $a_H = \frac{h}{2}$, $a_V = \frac{w}{2}$ and $F_i(y, x)$ represents the pixel intensity/color at coordinate (y, x). Considering the video as volume XYT, the resulting image can be seen as a plane parallel to XT (horizontal) or YT (vertical).



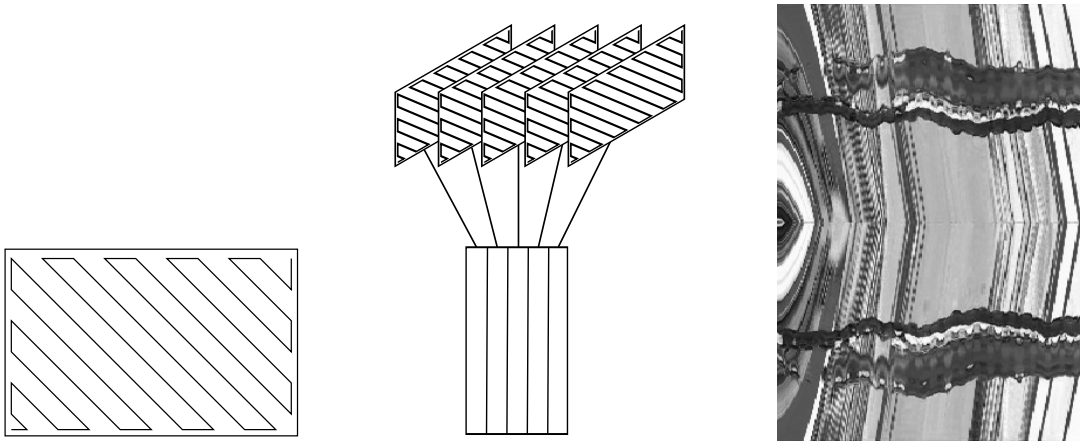
(a) Horizontal (b) Vertical

Figure 2.2: Examples of visual rhythm images generated for a video of the "CricketShot" class from the UCF101 dataset. The central row/column of each frame becomes a slice in the resulting image. Adapted from [70].

The visual rhythm can also be constructed by following a zig-zag scheme [77], as illustrated in Figure 2.3. Compared to vertical and horizontal VRs, this one covers a larger portion of the frame, but it consequently generates an image with a much greater height. Due to the dimensions of the final image, such approach may not be suitable for some scenarios, for instance, to be used without a severe reshape or cropping in a CNN that requires a smaller input.

Another possible strategy is to explore operations that generate an 1D feature per frame, such as the intensity histogram defined in Equation 2.7. The histogram represents the entire frame with a smaller slice, but it loses spatial information.

$$T(F_i) = \left[\sum_{x=1,y=1}^{w,h} 1, \text{ if } F_i(y, x) = k \right]^T, \forall k \in \{0, 1, \ldots, 255\}. \tag{2.7}$$



(a) Zig-zag scheme (b) Slice concatenation (c) Example

Figure 2.3: Visual rhythm construction using a zig-zag scheme. Extracted from [77]

2.4 Support Vector Machine

The main goal of a classifier is to predict the label of new samples after the training on a labeled set. In binary problems (that is, problems with two possible labels), this is usually carried out by estimating a hyperplane that separates the two groups of samples according to their labels. If the resulting hyperplane is too close to any of the groups, the method may fail in handling new samples. Therefore, it is desirable that the distances among the separating hyperplane and the nearest samples from each group are as great as possible. These distances are referred to as classification margins and the nearest samples as support vectors (Figure 2.4).

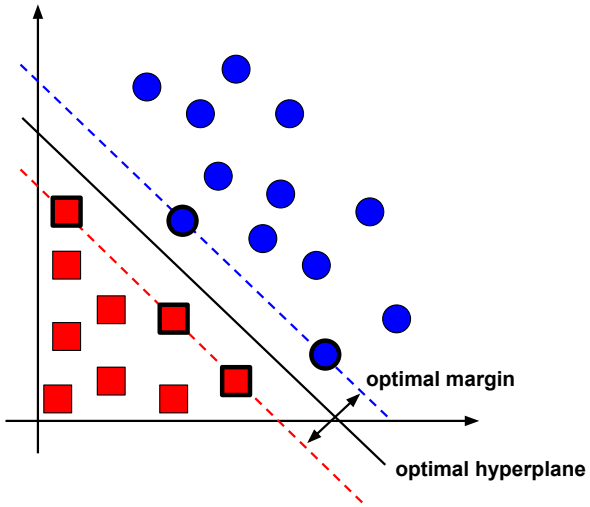


Figure 2.4: Separating hyperplane with maximum margin in a two-dimensional space. Thicker squares and circles represent the nearest samples or support vectors. Adapted from [25].

The margin maximization constraint is the central objective of the Support Vector Machine method (SVM [25, 96]). Here, we consider the C-Support Vector Classification

(C-SVC [17]). Let $\mathbf{x_i} \in \mathbb{R}^n$ be a n-dimensional feature vector from the training set and $y_i \in \{-1, 1\}$ its corresponding label, where $i \in \{1, \cdots, n_s\}$ is the sample index and n_s is the total number of training samples. The C-SVC method is based on the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\left(\sum_{i=1}^{n_s}\xi_i\right), \\ \text{subject to} \quad & y_i(\mathbf{w}\cdot\mathbf{x_i} + b) \geq 1 - \xi_i, \qquad i \in \{1, \cdots, n_s\}, \\ & \xi_i \geq 0, \qquad\qquad\qquad\quad i \in \{1, \cdots, n_s\}, \end{aligned} \tag{2.8}$$

where $\mathbf{w}$ and b are the hyperplane parameters, C is a regularization term and $\boldsymbol{\xi}$ allows a soft-margin hyperplane. The greater the C, the smaller the margin and, consequently, the weaker the regularization.

If a problem is nonlinear, that is, the samples cannot be correctly separated using a hyperplane, the data can be mapped into a higher-dimensional space where the classes become separable. However, this process can be expensive. A kernel function is used to reduce the cost by computing a similarity between a pair of vectors in which the transformation is made implicitly. This strategy is called the "kernel trick". The Radial Basis Function (RBF), also known as Gaussian kernel, is a commonly used kernel expressed in one of the following forms:

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\alpha^2}\right) = \exp\left(-\gamma\|x_i - x_j\|^2\right), \tag{2.9}$$

where $\gamma = \frac{1}{2\alpha^2}$.

There are two strategies to extend the binary C-SVC to multi-class problems: (i) One-versus-One (OvO) and (ii) One-versus-Rest (OvR) or One-versus-All (OvA). In (i), one classifier is constructed for each pair of classes ignoring the samples from the remaining ones. Considering m the number of classes, in this strategy, $\frac{m\cdot(m-1)}{2}$ binary classifiers will be constructed. In (ii), one classifier is estimated for each class considering all the other classes as a single one, resulting in m binary classifiers.

The regularization parameter C, the kernel type (linear or RBF), γ in case of RBF kernel and the multi-class strategy (OvO or OvR) can be selected using a grid search technique.

2.5 Convolutional Neural Network

Convolutional Neural Network (CNN) is a type of neural network that contains at least one layer with convolutional kernels [35]. It is mostly used for images and grid-like data that presents local relations. In contrast to the layers from traditional neural networks, the convolutional ones present two important properties: (i) weight sharing and, as a consequence, (ii) sparse connectivity. The sparsity is due to the kernel size, which is much smaller than the input, to capture local patterns in the input such as edges. The kernels

slide across the entire input and, therefore, different regions share the same parameters.

The linear convolutional operation is usually followed by a nonlinear activation function. The most popular activations are logistic sigmoid, hyperbolic tangent and Rectified Linear Units (ReLU) [80]. The first two are saturating units since most of their domain are mapped to the same values (highest values are mapped to 1 and smallest values to 0) [35]. Krizhevsky et al. [54] showed that networks composed of the non-saturating ReLU learn several times faster than using the saturating units. As such, the ReLU is a common default choice for CNN architectures.

Other layers commonly found in CNN architectures are pooling ones that help to reduce dimension and aggregate information, and fully-connected layers which are usually placed at the end of the network, where the inputs are smaller, for classification purposes. Two common operations used for pooling are the average and max. The pooling layers can be either defined by the kernel size and stride (traditional pooling) or by the output size (adaptive pooling). The stacking of several layers allows the gradual combination of local information reaching a more global representation in the later layers. This is called feature hierarchy [61,62], where the pattern complexity increases as the data goes deeper, for instance, from edges to complete objects detection.

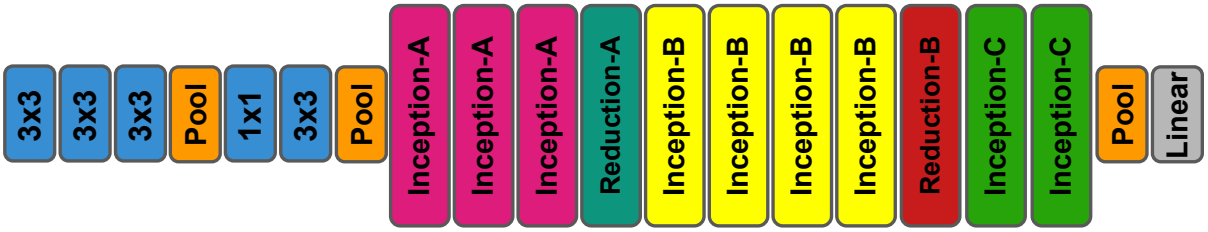
Several architectures have been proposed in the literature for the image classification problem (AlexNet [54], VGG [114], ResNet [41], EfficientNet [127], among others). Here, we are particularly interested in the Inception V3 architecture [123] (Figure 2.5). The Inception V3 is a very deep network based on Inception modules with factorized convolutions, being an efficient network with a low number of parameters. The factorization involves the replacement of larger spatial filters by multiple layers of smaller ones, for instance, replacing 5×5 convolutions by two layers of 3×3. This replacement leads to a reduction in the number of parameters, in the example, from 25 to 18 parameters. The modules also include the 1×1 convolutions from GoogLeNet for dimensional reduction.

2.6 Early Stopping

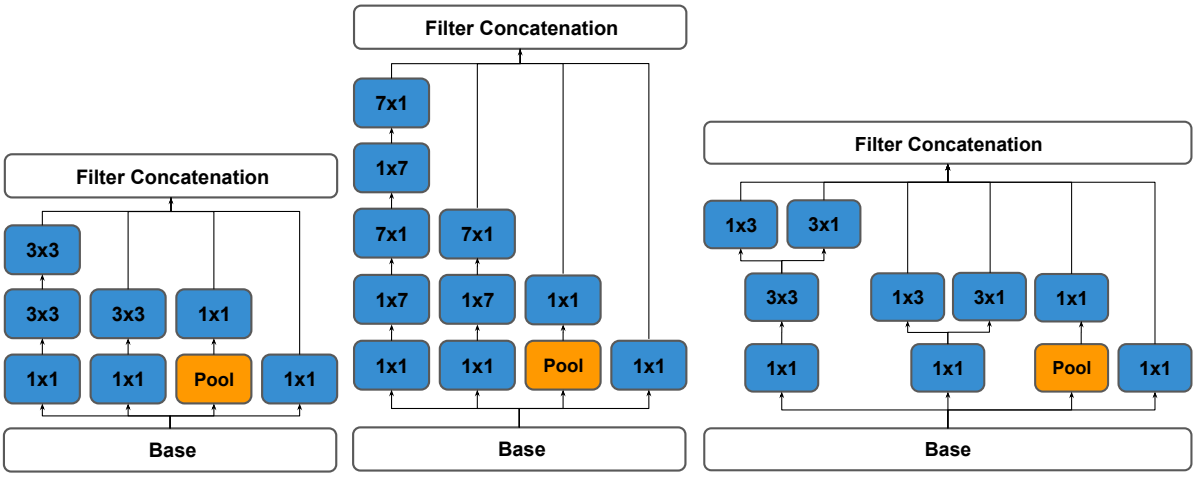
A well-known problem in machine learning is the overfitting, when a model fits the training set properly, but such performance is not reflected on new samples. A large difference between training and test accuracies/loss is a strong indication of overfitting.

There are several strategies in the literature to deal with overfitting. Some of the commonly used ones are: (i) replacement of the model by a simpler one with less trainable parameters (ii) model regularization such as dropout layers (iii) increasing the number of samples, perhaps using data augmentation techniques and (iv) training over a smaller number of epochs using the early stopping. Next, we give further details about the last strategy, which is explored in this work.

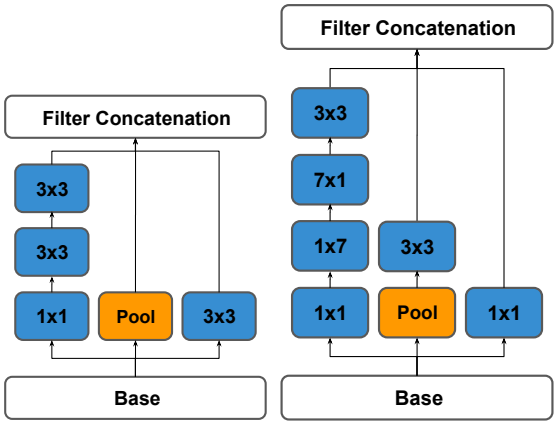
The key of the early stopping is the use of a validation set to estimate the generalization error, that is, the model behavior on an unknown set. The training stops when the performance on the validation set starts to deteriorate. Prechelt [95] presented three classes of criteria to define the stopping epoch. Here, we use the third class: the training stops after s consecutive strips of no improvement, where a strip is a group of k con-



(a) Inception architecture



(b) Inception-A (c) Inception-B (d) Inception-C



(e) Reduction-A (f) Reduction-B

Figure 2.5: Inception V3 architecture and modules [123]. Blue, orange and gray rounded rectangles represent convolutional, pooling and fully-connected layers, respectively.

secutive epochs. The s parameter is referred to as "patience". For being qualified as an improvement, the difference between the best evaluation and the current one must exceed a third parameter δ. This method is presented in Algorithm 1.

The method $EarlyStopping(l, e)$ is repeated at the end of each strip, using its last epoch e. The input l corresponds to the validation loss at e. If there is an improvement with regards to the best score l^* found at the moment, l^* and the best epoch e^* are updated. Otherwise, a counter is used to store the number of consecutive strips that the performance has worsened. The counter is reset after any improvement. After s deteriorations, the training process stops.

Algorithm 1 *EarlyStopping(l, e)*

Input
Current validation loss l;
Epoch e.
Output
stop {True or False}.

```
1: global l*, e*                                   ▷ Best loss and epoch.
2: stop ← False
3: if l* − l > δ then                              ▷ Verify if there was an improvement.
4:     l* ← l                                      ▷ Update best loss and epoch.
5:     e* ← e
6:     counter ← 0          ▷ Reset the counter of consecutive strips with no improvement.
7: else                                            ▷ The performance has worsened.
8:     counter ← counter + 1
9:     if counter ≥ s then          ▷ Stop training after s strips with no improvement.
10:        stop ← True
11: return stop
```

The validation accuracy can also be used for monitoring the performance throughout the epochs instead of the loss. In this case, the improvement condition becomes $l - l^* > \delta$ (line 2). The model obtained at the epoch e^* is the one suggested by the early stopping strategy. Optionally, the network can be retrained using both the training and validation sets for e^* epochs to obtain the final model.

2.7 Multi-stream Architecture

In contrast to image-related problems, methods involving videos do not have a great variety of well-established specific architectures. Architectures for processing raw video would have a prohibitive computational cost by sliding the filters along the temporal dimension besides the usual spatial ones. Moreover, only in recent years, larger video datasets have been released and are still considerably smaller than the ImageNet [101] used for 2D CNNs. For this reason, many video-related approaches explore 2D CNNs pre-trained on ImageNet, along with image representations containing spatio-temporal information.

This is the case of the two-stream network [113] (Figure 2.6), that explores two complementary image modalities inspired by the human visual system. Each modality is processed by a 2D CNN (or stream) and the respective outputs are fused to generate the final response. The first stream performs object recognition using a single RGB frame randomly selected from the video. In this way, it captures elements that compose the appearance of the action, such as the usual scenario, objects involved, among others. For instance, recognizing a guitar may help to recognize a video from the PlayingGuitar class. However, a green grass field or even the equipment may not be sufficient to distinguish between a CricketBowling and a CricketShot video. Therefore, some dynamics are needed to complement the spatial information. This is carried out by the second stream that receives 10 pairs of consecutive optical images in the form of a 20-channel image

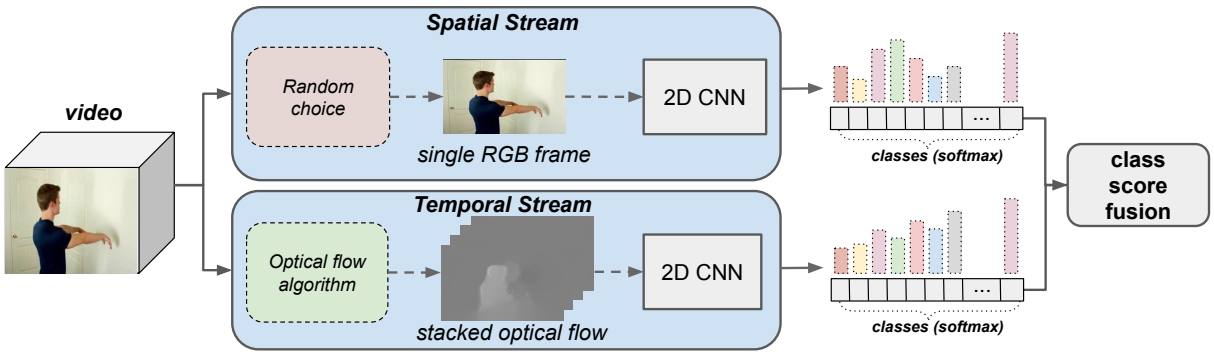


Figure 2.6: Two-stream architecture [113] composed of two parallel CNNs, the spatial and temporal streams.

representing the motion information. The 20-channel image is referred to as a stack of optical flow images.

The streams are individually trained and fused in the test stage. Originally, only the spatial one was pre-trained with the ImageNet dataset, since it is based on object classification. For the temporal stream, the authors proposed a multi-task learning, combining multiple video datasets to circumvent the lack of a larger one. However, experiments from Wang et al. [141] showed that the ImageNet pre-training improves the temporal training as well. The authors also suggested the replacement of the CNN-M-2048 [18] network used in the original two-stream by deeper architectures, such as VGG and GoogLeNet, due to the complexity of the action recognition problem.

In the training stage, the data is augmented using random cropping, horizontal flipping and RGB jittering. For testing, 25 frames/stacks of optical flow images are evenly selected from each video and used to produce 10 new samples per frame/stack by cropping and flipping techniques. Each sample is individually tested in the corresponding stream. Finally, the class scores computed in each CNN are combined using a simple average and linear SVM.

2.8 Stacking

The stacked generalization or stacking consists of placing a classifier on top of an ensemble of base classifiers to combine their outputs. The base classifiers are called first-level classifiers and the stacked one is called second-level classifier or meta-classifier [65, 155].

In stacking, a new dataset is produced using the outputs of the original one generated by the base classifiers. Each new sample receives the label of its corresponding original sample. The original dataset is used to train the base classifiers, whereas the meta-classifier is trained with the new one. The general method is shown in Algorithm 2.

To avoid overfitting, a k-fold cross-validation is recommended (Algorithm 3). The original dataset D is divided into k subsets D_l ($l \in \{1, 2, \cdots k\}$) and the base classifiers are trained k times leaving the subset D_l out at the l-th iteration. The subset that was set aside is used for generate a new subset D'_l. In this version, the new dataset D' is formed by the union of the new subsets: $D' = D'_1 \cup D'_2 \cup \cdots \cup D'_k$. After the training step

of the meta-classifier with D', the base classifiers can be retrained using the entire set D. Although this version may reduce the overfitting, the process can be computationally expensive since the base classifiers training step is repeated multiple times.

Algorithm 2 General stacking procedure adapted from [155].

Input
Training set $D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$; ▷ x_i is the i-th input and y_i its label.
Base classifiers $C_1, C_2, \cdots, C_m$;
Meta-classifier C.

for all $j \in \{1, 2, \cdots, m\}$ **do**: ▷ Base classifier training.
    $C_j.fit(D)$
$D' = \emptyset$ ▷ New training set.
for all $i \in \{1, 2, \cdots, n\}$ **do**:
    for all $j \in \{1, 2, \cdots, m\}$ **do**:
        $z_{ij} = C_j(x_i)$ ▷ Output generated by the classifier C_j for the input x_i.
    $x'_i = (z_{i1}, z_{i2}, \cdots, z_{im})$
    $D' = D' \cup \{(x'_i, y_i))\}$
$C.fit(D')$ ▷ Train the meta-classifier using the new set.

Algorithm 3 Stacking with k-fold adapted from [65].

Input
Training set $D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$; ▷ x_i is the i-th input and y_i its label.
Base classifiers $C_1, C_2, \cdots, C_m$;
Meta-classifier C.

$F = \{D_1, D_2, \cdots, D_k\}$ ▷ Randomly split D into k equal-size subsets.
for all $l \in \{1, 2, \cdots, k\}$ **do**:
    for all $j \in \{1, 2, \cdots, m\}$ **do**: ▷ Base classifier training.
        $C_j.fit(D \setminus D_l)$
    $D'_l = \emptyset$ ▷ New training subset.
    for all $x_i \in D_l$ **do**:
        for all $j \in \{1, 2, \cdots, m\}$ **do**:
            $z_{ij} = C_j(x_i)$ ▷ Output generated by the classifier C_j for the input x_i.
        $x'_i = (z_{i1}, z_{i2}, \cdots, z_{im})$
        $D'_l = D'_l \cup \{(x'_i, y_i))\}$
$D' = D'_1 \cup D'_2 \cup \cdots \cup D'_k$
$C.fit(D')$ ▷ Train the meta-classifier using the new set.
for all $j \in \{1, 2, \cdots, m\}$ **do**: ▷ Base classifier retraining.
    $C_j.fit(D)$

2.9 Diversity Measures

In order to improve individual results, a combination (also called an ensemble) must contain classifiers that do not always agree on the predictions. Otherwise, some classifiers would be redundant and the combination would be inefficient by increasing the cost

without any gain. This expected property is called ensemble diversity. Proper individual performances are also desirable.

Table 2.1: Summary of some diversity measures adapted from [155]. "↑/↓" indicates that the higher/lower the score, the higher the diversity. The "range" column indicates the measure interval and "pairwise" indicates whether it is taken from pairs or from the entire ensemble. A brief description for each measure is also presented in the table.

Measure	↑ / ↓	Range	Pairwise	Brief Description
Disagreement (DM)	↑	$[0, 1]$	Y	Proportion of different answers.
Q-statistic (Q)	↓	$[-1, 1]$	Y	Degree of agreement. Positive values indicate higher agreement and negative values indicate higher disagreement.
Correlation coefficient (ρ)	↓	$[-1, 1]$	Y	Similar to Q-statistic.
Kappa-statistic (κ_p)	↓	$[-1, 1]$	Y	Degree of agreement by chance. 1 means total agreement, 0 means agreement-by-chance and negative value means that the agreement is lesser than expected by chance (rare).
Double-fault (df)	↓	$[0, 1]$	Y	Proportion of samples misclassified by both.
Complementarity [11] ($Comp$)	↑	$[0, 1]$	Y	Frequency that a classifier is correct given that the other is incorrect.
Interrater agreement (κ)	↓	$[-1, 1]$	N	Similar to kappa-statistic.
Entropy 1 (ent_{cc})	↑	$[0, 1]$	N	Computes the Shannon entropy for every instance considering the variety of labels assigned for it by the classifiers.
Entropy 2 (ent_{sk})	↑	$[0, 1]$	N	Similar to entropy 1, but it considers the correctness of the classifiers.
Difficulty (θ)	↓	$[0, \infty)$	N	It is based on a probability mass function that considers the proportion of classifiers that correctly classify each sample.
Generalized diversity (gd)	↑	$[0, 1]$	N	Based on the proportion of cases where the failure of one classifier is accompanied by the success of another one.
Coincident failure (cfd)	↑	$[0, 1]$	N	Modified gd where the minimum value (0) indicates higher agreement (simultaneous fail or success).

Several diversity measures have been proposed to evaluate the quality of ensembles [30, 56, 155]. They can be pairwise, where the final score is given by the average measures taken from every pair of classifiers, or non-pairwise, which takes a single score considering the ensemble as a whole. Table 2.1 shows a summary of some diversity measures.

Experiments performed by Kuncheva [56] suggest that there is no clear relationship between diversity metrics and accuracy. Therefore, they are not indicated to be used to support decisions during the ensemble construction. However, they can be very useful for analysis and visualization, once the combination is already defined.

The pairwise metrics are defined using the values of a hit-and-miss table [30] (Table 2.2). The table compares two different classifiers c_i and c_j. Values a and d indicate

the number of samples in which both classifiers made a correct and incorrect prediction, respectively. The value b indicates the number of cases where c_j is correct and c_i is incorrect, whereas c indicates the opposite. The sum of the four values is equals to the total number of samples n_s.

Table 2.2: Hit-and-miss table [30].

	Hit c_i	Miss c_i
Hit c_j	a	b
Miss c_j	c	d

The first four metrics measure the disagreement between the classifiers, but they do not capture a disparity between the number of exclusive hits, that is, $b \gg c$ or $c \gg b$. In the worst case scenario, where one classifier gives an almost perfect set of predictions, whereas the other fails in most samples (without loss of generality, $a = 0$, $b = n_s - 1$, $c = 1$ and $d = 0$), they assign their best score to the combination (see Appendix A).

It is worth mentioning that a simultaneous $a = 0$ and $c = 0$ is preventable by selecting classifiers with good individual performance. The complementarity metric is capable of detecting these disparities, since it generates two values per pair according to the classifier order, but it fails with small values of d. The double-fault metric considers only the value of d, therefore, it cannot detect either disparities or the combination of perfect classifiers (that is, $a = n_s$). Since all metrics have their own weaknesses, we decided to combine two of them: (i) complementarity and (ii) kappa-statistic. We give further details on their computation as follows.

The complementarity $Comp(c_i, c_j)$ measures the percentage of c_i misclassifications that c_j is capable of covering, that is, how much c_j complements c_i. It is defined as:

$$\begin{aligned} Comp(c_i, c_j) &= 1 - \frac{\#\text{ of common errors}}{\#\text{ of } c_i \text{ errors}} \\ &= 1 - \frac{d}{(b+d)} \\ &= \frac{b}{(b+d)}. \end{aligned} \tag{2.10}$$

$Comp(c_j, c_i)$ is given in Equation 2.11. Notice that, if $b \neq c$, then $Comp(c_i, c_j) \neq Comp(c_j, c_i)$.

$$\begin{aligned} Comp(c_j, c_i) &= 1 - \frac{\#\text{ of common errors}}{\#\text{ of } c_j \text{ errors}} \\ &= \frac{c}{(c+d)}. \end{aligned} \tag{2.11}$$

To obtain a unified score, we use a harmonic mean of both values given by Equation 2.12. Therefore, we examine whether there is a balance between the two contribu-

tions.

$$\mathrm{HM}(c_i, c_j) = \frac{2 \cdot Comp(c_i, c_j) \cdot Comp(c_j, c_i)}{Comp(c_i, c_j) + Comp(c_j, c_i)} \tag{2.12}$$

The kappa-statistic κ_p uses the agreement θ_1 and agreement-by-chance θ_2 measures, defined as:

$$\theta_1 = \frac{a+d}{n_s}$$

and

$$\theta_2 = \frac{(a+b)(a+c)+(d+b)(d+c)}{n_s^2}.$$

Thus, κ_p is given by the following equation:

$$\kappa_p(c_i, c_j) = \frac{\theta_1 - \theta_2}{1 - \theta 2}. \tag{2.13}$$

A high agreement value results in a high κ_p (up to 1), whereas a higher agreement-by-chance leads to a κ_p near to 0. A negative value is a rare case where the agreement is less than expected by chance.

Concerning the non-pairwise measures, we select the interrater agreement, which is similar to the kappa-statistic. Let m be the number of classifiers in the combination and n_s the total number of samples. The interrater agreement $\kappa(c_1, c_2, \cdots, c_m)$ is defined as:

$$\kappa(c_1, c_2, \cdots, c_m) = 1 - \frac{\frac{1}{m}\sum_{k=1}^{n_s} p(x_k) \cdot (m - p(x_k))}{n_s \cdot (m-1) \cdot \bar{p} \cdot (1-\bar{p})}, \tag{2.14}$$

where $p(x_k)$ is the number of classifiers that predicted the correct label for the sample x_k, $\bar{p} = \frac{1}{m}\sum_{i=1}^{m} acc_i$ and acc_i is the accuracy of the i-th classifier (ratio between the correctly predicted samples and n_s).

Chapter 3

Related Work

In this chapter, we briefly present some approaches available in the literature that focus on the action recognition problem. The solutions generally involve the representation and classification of the actions, although this division is not always explicit.

We can categorize the approaches into two groups based on the strategy used for each of these tasks, according to the classification proposed by Goodfellow et al. [35]. The first group consists of the traditional methods, where the representation is explicitly chosen and the action recognition is defined under conventional machine learning algorithms. The second group comprises the representation learning strategies that explore machine learning techniques for both tasks.

Table 3.1 shows the two groups along with schematic diagrams and references from each one. The representation learning approaches are further divided into three subgroups, according to the depth of the machine learning portion during the representation stage. The first subgroup comprises approaches that are not entirely based on machine learning for representation, but they use it to provide a final descriptor from a set of handcrafted features, for instance, dictionary-based methods [59,90,135]. As such, they are sometimes referred to as shallow learning [18]. In contrast, approaches based on deep neural networks are called deep learning strategies (second and third subgroups). Generally, deep networks have many layers that perform successive refinements of the data, increasing the level of abstraction during the process. The last subgroup is the extreme case where the entire representation process is performed by a deep network fed with raw videos, a version analogous to image-based networks. This type of approach present a high computational cost thanks to the 3D convolutions and require a severe temporal cropping or frame subsampling.

For the best of our knowledge, only the 3-dimensional convolutional network (C3D [129]) and the slow fusion model [48] work purely with raw videos and spatio-temporal convolutions. Even some approaches that deal with spatio-temporal volumes and perform 3D convolutions include handcrafted features for the task, such as the two-stream I3D [15] and the 3D CNN from Ji et al. [46].

Despite the decrease in approaches following the traditional and shallow strategies, many recent deep learning proposals explore elements from both in order to preprocess the data or aggregate information. These elements allow the methods to take advantage of deep networks for images since the temporal information may be incorporated in a

Table 3.1: Taxonomy for action recognition approaches. The representation and classification stages are illustrated as green and red rectangles, respectively. "HC" refers to handcrafted process and "ML" to machine learning-based techniques.

Group	Representation / Classification	References
Traditional Approaches	HC | ML	[4, 12, 34, 71, 78, 91, 103, 148]
Representation Learning	HC | ML | ML	[6, 59, 77, 90, 93, 110, 135, 136]
	HC | ML (deep) | ML	[5, 15, 20, 28, 33, 46, 107] [64, 81, 87, 113, 138, 141, 142] [144, 147, 149, 157, 158]
	ML (deep) | ML	[48, 129]

two-dimensional form, instead of considering the video as a volume.

There are classical surveys for the action recognition problem [1, 94]. However, since deep learning strategies are very recent, they are not covered in these papers. Some interesting surveys that describe relevant concepts include [39, 42, 108, 156].

3.1 Traditional Approaches

Many features have been proposed in the literature for representation in computer vision problems. Here, we highlight four features that served as basis for many strategies to represent human actions, even though they were not initially proposed for such purpose. The chosen features are: local binary pattern, gradient, optical flow and visual rhythm (or temporal slices). We also present some of the derived strategies. At the end of the section, we present some methods for aggregating information.

Local Binary Pattern (LBP) is a feature proposed by Ojala et al. [86] to represent the local texture of a static image. The local representation is given by the distribution of texture patterns computed through a set of binary tests. LBP is simple, has a low computational cost and is invariant to global illumination changes. Moreover, it was an important precursor of several methods for videos based on binary/ternary information [4, 50, 93, 148, 152].

Zhao and Pietikainen [152] proposed the Volume Local Binary Pattern (VLBP) as an extension of the LBP for videos, applied to facial expression recognition. They compared intensity values from regions of three frames with distance L between two adjacent frames,

resulting in a so-called dynamic texture as it incorporates motion information. Since more pixels are compared in the VLBP, the number of possible texture patterns is larger than that in LBP. According to Baumann et al. [4], this large variety of patterns may lead to several ambiguities. Thus, they proposed the Motion Binary Pattern (MBP) to combine binary patterns through an exclusive OR (XOR). Furthermore, they ignored weak motion by filtering out some patterns with less than a certain proportion of motion.

Yeffet and Wolf [148] described ternary tests performed in sets of three evenly spaced frames. The additional value (the third one) is assigned to regions where no significant motion was detected, that is, regions that present very similar intensities. Thus, the method is able to overcome slight variations in illumination. To reduce the high computational cost related to ternary digits, they converted each ternary string (that is, the result of a sequence of ternary tests) into two binary ones, obtaining a smaller but efficient descriptor containing two histograms. One local descriptor is computed for each spatio-temporal cuboid of the video. The Motion of Oriented Magnitudes Pattern (MOMP [93]) feature is inspired in this strategy. The main difference is that, instead of using color intensities, MOMP is based on gradients, further increasing the robustness to illumination changes.

Dalal and Triggs [26] proposed the Histogram of Oriented Gradients (HOG) that encodes shape information. HOG represents the object local appearance by the distribution of intensity gradients and, for this reason, it is robust to illumination changes. It was originally proposed for human detection in static images based on two-dimensional gradient vectors. However, several works have adapted this strategy to the human action recognition problem by incorporating temporal information.

Klaser et al. [53] proposed an extended version of HOG for videos, known as HOG3D, based on spatio-temporal gradients. Additionally, they proposed an efficient method for computing the descriptor, extending integral images to integral videos. The proposed version outperforms approaches from the same period in most of the evaluated datasets. Many other HOG-based descriptors have been proposed in the literature [59,78,91,93,103, 110], achieving good rates and consolidating HOG as an important tool for the human action recognition problem.

Another interesting strategy that explores gradient information is described by Shi et al. [110]. They proposed the Gradient Boundary Histograms (GBH) descriptor to reduce the dimensionality of HOG3D and its expensive quantization cost. Instead of computing three-dimensional vectors, it calculates the spatial gradients followed by the subtraction of two consecutive image gradients. Thus, it produces two images per frame corresponding to each spatial coordinate that can be seen as a two-dimensional vector field. The resulting images contain double edges in which their distances are proportional to the moving speed of the human body parts. Additionally, the subtraction leads to a reduction of background noise.

The third strategy for describing an action is through the estimation of the object apparent motion, also know as optical flow. In addition to the Lucas-Kanade and TV-L^1 methods presented in Section 2.2, there are other approaches on the literature to compute optical flow, such as Horn-Schunck [43] and Farneback [31]. The methods compute displacement vectors for pairs of consecutive frames, resulting in a sparse or dense vector field. Vectors from consecutive pairs can be combined to form trajectories [12,135]. As gradients,

the distribution of displacement vectors or trajectories can be encoded into a representation called Histogram of Optical Flow (HOF) to explore local motion [59, 78, 135].

An alternative motion-based descriptor was presented by Maia et al. [71]. It is based on the block matching method combined with a quad-tree algorithm that results in a sparse vector field. Similar to HOG and HOF, a histogram of oriented vectors is built to represent the action. The final descriptor has shown to be compact and fast to compute.

Souza et al. [118] presented a survey of works that employ visual rhythms for several video-based problems. For instance, Ngo et al. [82,83] proposed a method for locating and classifying video transitions through the analysis of the central row (horizontal), central column (vertical) and the main diagonal subsampled from the frames. This is possible because transitions in videos generally result in boundary lines in the 2D image. The shots (that is, a video segment between two transition frames) are further subdivided according to the camera motion also based on patterns found in temporal slices, but using horizontal and vertical slices [84]. The authors argued that, compared with other spatio-temporal features, temporal slices has the advantage of providing long-term information instead of encoding only a few frames (two or three).

Valio et al. [133] proposed a method based on visual rhythm for detecting frames containing captions. The slices are given by zig-zag curves to handle different orientations and locations. Since captions remain static for a certain period of time, the locations corresponding to these texts form rectangles in the visual rhythm. In addition to zig-zag scans, other space-filling curves [104] are suggested by the authors to be explored in methods based on visual rhythms.

Torres and Pedrini [128] explored visual rhythm images to address three computer vision problems: abnormal event detection, human action recognition and hand gesture recognition. Visual rhythm is used to estimate object trajectories throughout the video. Slices that capture leg motion, for instance, produce an interesting braided pattern in actions as walking (Figure 3.1). For the action recognition problem, high-pass filters are applied, followed by the selection of Regions of Interest (RoI) to keep only the information relating to the trajectory.

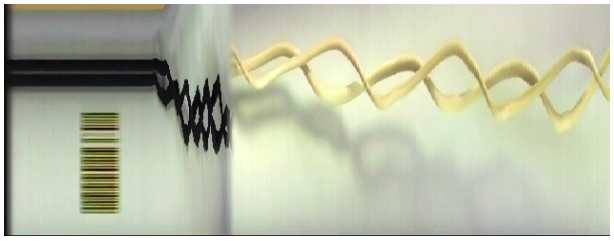


Figure 3.1: Visual rhythm of the action walking, adapted from [128]. The braided pattern corresponds to the leg motion.

Moreira et al. [77] proposed two complementary methods based on visual rhythms: the holistic Naïve VR and the patch-based VR Trajectory Descriptor (VRTD). They used zig-zag rhythms in four directions, two diagonals, vertical and horizontal. The authors argued that the objects are transferred to the rhythm as they cross the slices in the frames. The appearance of the transferred object in the rhythm highlights some aspects of the action, for instance, a flattened shape may indicate a high-speed movement. Thus, they

extracted texture and shape information from the rhythms to capture these aspects. They also explored several image domains for this purpose, such as gradient of the frames and optical flow images.

Most of the presented features are computed for a small set of frames or a region of the video volume. Thus, once the features are extracted, the next concern would be how to aggregate the information to represent the entire video. Many works use feature concatenation to combine this local information and produce a single representation, or even to combine different features to form a mixed descriptor [4, 12, 53, 78, 148, 152]. It consists in concatenating the features one after another. The major drawback of this strategy is that it does not consider the correlation among features and, therefore, it may keep redundancies and produce large descriptors.

A region descriptor given by the covariance matrix was proposed by Tuzel et al. [131]. The authors proposed a fast way to compute it based on integral images. Covariance matrix provides a natural way of combining feature vectors by exploring correlations, and presents a smaller dimension when compared with raw data and concatenated histograms. Additionally, the representation is partially invariant to rotation and scale according to the chosen features, as the order and position of the points are not considered. Since covariance matrix does not lie on Euclidean space, it cannot be directly applied in traditional machine learning methods. For this reason, the authors proposed an adapted version of nearest neighbor, using a different distance metric. Several works in the literature have been based on this region descriptor [6, 16, 21, 40, 105, 143, 150].

A similar approach is used in other methods [12,34,71,78,91] exploring tensors. Tensors are computed from local histograms and accumulated to generate a final representation for the video. The shape of the tensor provides meaningful information about the motion. Moreover, all videos are represented by a single and compact matrix regardless of the number of features and frames.

An experimental study of different fusion techniques was presented by Patel et al. [88]. Their results demonstrated the importance of the fusion in the system performance. The authors tested four methods for late fusion: (i) decision combination neural network, a fully-connected layer that combines the score vectors from different classifiers, (ii) Choquet fuzzy integral and (iii) decision template both based on fuzzy logic, and (iv) dynamic weighting by averaged distances based on nearest neighbors. According to their experiments, the Choquet method achieved superior results.

3.2 Representation Learning

In many cases, the number of feature vectors generated through the techniques presented in Section 3.1 is very large and varies from one video to another, even within the same dataset. This might increase the computational cost and make many fusion methods unfeasible (for instance, concatenation).

In addition, the set of features might include redundant and irrelevant information and compromise the classification performance. Thus, a group of methods, referred to as shallow learning approaches, attempt to learn a proper representation from a set of

features. This group includes Bag of Features (BoF), Fisher-vectors (FV) [92] and Vector of Locally Aggregated Descriptors (VLAD [45]), and has been successfully employed in many works [6, 21, 40, 59, 77, 90, 93, 110, 135, 136].

These methods find a collection of visual words or vectors that provide good discriminability and represent the entire video according to them. The resulting descriptor can be used in standard classification methods. Among the advantages of these methods, we can mention the fixed length of the output as in tensor-based approaches and the reduction of the descriptor dimensionality. Moreover, they are slightly less dependent on human experts, since the action is represented through machine learning techniques.

Before the deep learning era, the state-of-the-art results on human action recognition were achieved by the Dense Trajectories (DT) frameworks [90, 135, 136]. In fact, they are still explored in combination with deep networks, improving their performance [44, 115, 134, 140]. Chapter 2 of the work described by Sintes [115] presents a good review of DT papers. These methods are based on dense sampling at regular position in space and time, in contrast to Space-Time Interest Point (STIP) detectors, using multiple spatial scales.

The dense trajectory term comes from the dense optical flow methods used to extract the trajectories. The trajectories are described using the aforementioned HOG and HOF, and the Motion Boundary Histogram (MBH) which is based on the horizontal and vertical components of the optical flow. These local descriptors are fused using a BoF approach. This basic framework was first introduced by Wang et al. [135]. Afterwards, they proposed the Improved Dense Trajectories (iDT [136]) method that estimates the camera motion to correct the trajectories. They also show a comparison of the BoF (iDT + BoF) and FV (iDT + FV) methods, and the latter achieves superior results. Peng et al. [90] proposed a new fusion for the iDT combining different BoF methods called Hybrid SuperVector (iDT + HSV).

Over the last years, deep learning has become a popular topic of researches in machine learning and related areas due to its high performance in diverse applications and its adaptability to different scenarios [54, 114]. The evolution of optimization techniques and hardware, associated with the availability of large datasets, has also contributed towards deep learning development advances. Herath et al. [42] provided an interesting survey of notable methods for human action recognition, including traditional and deep learning strategies. By following the same categorization for deep learning approaches, our method falls under the multiple stream group.

One of the pioneer works on deep learning is proposed by Lecun et al. [60]. They presented some shortcomings of the traditional methods, such as the dependence of the designer's ability on choosing proper features. Moreover, the chosen features are generally specific for the task. They argued that a system that learns patterns outperforms handcrafting methods. Thus, they proposed a multilayer neural network based on trainable convolution filters called convolutional neural networks (CNN) used to understand the image content. The deep architecture performs a set of successive refinements in the raw data to extract complex and robust patterns. This strategy has been successfully applied in several problems involving images [18, 46, 54, 81, 114].

Inspired by the image-based networks, Ji et al. [46] proposed a straightforward ex-

tension for videos, adding a third dimension to the convolution filters. In other words, they proposed a single CNN designed to capture spatial and temporal information. The foreground is extracted from videos resulting in one or several sets of bounding boxes, according to the number of humans performing actions in the scene. Each set is converted into an n-frame cube (5-7 frames) that is used as input to the network. Five different versions of the cubes are produced in the first layer (hardwired layer): gray, horizontal and vertical gradients, horizontal and vertical optical flows. The proposed architecture achieved competitive results in controlled datasets and outperformed compared methods in more realistic ones. However, according to Herath et al. [42], this strategy presents some drawbacks. The fixed and small number of frames in the cubes does not take into account the speed of the action, for instance, different actions may take different number of frames to be completed. Moreover, 3D extensions generally increase the number of parameters in the network, leading to a significant increase in the training cost.

Karpathy et al. [48] proposed four architectures for fusing temporal information across video frames. Their experiments indicated that slow fusion outperforms the other three architectures by gradually aggregating temporal information at different levels of the network. In addition, they proposed a multi-resolution architecture consisting of two streams: context stream that receives downsampled frames and fovea stream that receives the central region of each frame at the original resolution. Both streams input has the same spatial resolution (89×89 pixels), allowing parameter sharing and, therefore, reducing training cost. In addition, this multi-resolution architecture achieved the best result compared with the previous four.

Tran et al. [129] argued that the 2D convolutions lose temporal information, since it encodes multiple images, represented as channels in the input, into a single channel output. Therefore, even though networks such as the previous one [48] combine multiple frames, the temporal information is completely collapsed after the first convolutional layer. For this reason, Tran et al. proposed the C3D network that is based on 3D convolutions. As pointed out by Carreira et al. [15], although the C3D seems to be a natural approach for video modeling and shows promising results, it was surpassed by its 2D counterparts, and a possible explanation is that it requires a larger dataset to be trained, since it cannot be pre-trained on the ImageNet.

Some works explore techniques usually found in traditional and shallow approaches for temporal fusion, using the deep network only as a frame-level feature extractor. This strategy was used by Kahani et al. [47] for first-person activity recognition and is motivated by two characteristics of the problem: relatively small datasets containing a few hundred of videos, and the requirement of real-time responses. In their framework, CNN outputs are considered as a set of time series, and intra and inter-time relations are computed through auto and cross-correlation, respectively. The final representation is given by the concatenation of both intra and inter-time relations. Similarly, Ravanbakhsh et al. [97] explored a 2D CNN as feature extractor. They built a hierarchical structure of snippets (temporal segments) and compute a CNN flow for each snippet in order to capture coarse and fine motion. The remaining stages of the framework include Principal Component Analysis (PCA) for dimensionality reduction, BoF and SVM for classification, all widely explored in standard approaches.

Inspired by the human visual system, a multi-stream network was proposed by Simonyan and Zisserman [113] (Section 2.7), where one stream performs object recognition (spatial stream) and the other recognizes the motion (temporal stream). Both streams consist of a 2D CNN, however, spatial stream receives still frames as input, while temporal stream receives stacked optical flow images. Results from several training strategies for each stream, evaluated separately, are presented in the work. Their results showed that the combination outperformed individual models, suggesting that both streams are complementary. Despite the complementarity, the short temporal extension (about 10 frames per video) is once more an issue. Nevertheless, this architecture gave rise to a variety of state-of-the-art approaches that explore RGB frames and optical flow images [5, 15, 28, 138, 142, 144, 157].

Ye et al. [147] investigated the impact of several implementation details on the two-stream network performance, such as backbone architectures, learning parameters and fusion methods. In particular, the authors tested an SVM for classification, replacing the last layer of the streams. The SVM is tested either for individual predictions and fusion. In their experiments, the original last layer with an average fusion achieve better results. Wang et al. [141] also proposed a list of good practices for the two-stream model including pre-training the temporal stream with the ImageNet and new data augmentation techniques. Most of their findings are still applied to recent multi-stream networks. Other fusion methods are proposed on the literature to replace the average combination, either fusing the predictions [107] as the original two-stream [113], or the feature vectors [33, 64, 87] similar to Ye et al. [147].

Wang et al. [138] proposed a three-stream architecture, adding a third stream to the two-stream network [113]. They suggested that the original network loses much appearance details because it uses a single frame and a single group of optical flow images per video. Thus, the additional stream is designed to combine appearance and motion information simultaneously for a longer period of time. This is accomplished through the use of dynamic images that encodes temporal evolution of the frames. Dynamic images encode information along 20 consecutive frames. Every combination of the three streams is evaluated. They concluded that the complete combination achieves better results and hence the new stream complements previous information. In addition, results suggest that the temporal stream is crucial for the network, since combinations without it reach inferior performance. Bilen et al. [5] proposed the use of dynamic images for both the spatial and temporal streams, resulting in a four-stream network.

Ng et al. [81] argued that methods that explore the entire video achieve better results than those using short segments. As such, they considered six different architectures based on RGB and optical flow, placing pooling layers in different locations of the network to obtain long-term motion information. They also proposed an architecture based on Long Short-Term Memory (LSTM) cells for temporal fusion, referred to as two-stream + LSTM. The proposed architectures are capable of processing up to 120 frames per video. A similar approach is proposed by Wang et al. [142], where multiple RGB frames and stacks of optical flow are extracted to capture longer-term information. However, for each input, this network outputs a preliminary prediction of the action classes instead of features. The predictions are then fused using a segmental consensus function that

does not impose temporal limits. This architecture is called two-stream TSN (Temporal Segment Networks).

Zhu et al. [158] proposed the gating ConvNet for the two-stream TSN fusion. Using this new module, they can assign adaptive weights for the fusion based on feature maps taken from the spatial and temporal streams, in contrast to the fixed-weight fusion. Chen et al. [20] introduced a TSN-inspired consensus fusion to capture long-term information, but they proposed a heterogeneous two-stream network, employing architectures that better adapt each stream. Both versions [20, 158] outperformed the original two-stream TSN [142] on the UCF101 dataset.

Yu et al. [149] and Kwon et al. [58] explored pooling operations to combine multiple frame-level feature vectors from RGB and optical flow and obtain a video-level representation. Both employed SVM classifiers to predict the final class from the video-level descriptors. Diba et al. [28] proposed the Temporal Linear Encoding (TLE) to combine short-term information (3 frames) in an end-to-end manner for each stream. For the late fusion, they applied an average of the scores. TLE and the four-stream network [5] achieved the highest scores on the UCF101 and HMDB51 datasets among the works presented so far.

Carreira and Zisserman [15] proposed a new network by inflating state-of-the-art image architectures into 3D CNNs. This method is also based on complementary information and is called two-Stream Inflated 3D ConvNets (two-Stream I3D). Both the filters and the parameters are replicated along the time dimension for the conversion into 3D. In addition to the traditional ImageNet pre-training step, an additional one is performed using a large video dataset (approximately 300k clips), the Kinetics [49]. This extra step led to a considerable increase in accuracy rates, especially on the HMDB51 dataset.

In fact, the current best results on the UCF101 and HMDB51 datasets are achieved by exploring the Kinetics to pre-train the network [15,22,130,139,157]. Choutas et al. [22] and Wang et al. [139] used the I3D model combined with their own methods. The DTPP [157] (Deep networks with Temporal Pyramid Pooling) is a method that aggregates frame-level features in multiple levels to obtain a video-level representation. It is composed of a fixed number of parallel image-based networks and a temporal pyramid pooling layer for aggregation and is trained in an end-to-end manner, that is, all its components are trained together. In order to perform spatio-temporal convolutions, Tran et al. [130] factorized 3D convolutional filters into spatial and temporal components (2+1D), achieving superior results when compared to 2D CNN. All of these works improved their initial results by pre-training their networks on Kinetics, or using the I3D, which is already pre-trained. However, due to Kinetics size, this strategy required further computational resources. Since 2019, we have also seen a trend in the literature towards using this dataset and other larger ones, such as AVA [38] and Charades [112], for method evaluation [32, 52, 79, 145, 146], although the HMDB51 still has room for improvement.

A recent approach that is evaluated on the UCF101 and HMDB51 datasets and achieves good results without the Kinetics pre-training is the Temporal Excitation and Aggregation (TEA) method. TEA is composed of a Motion Excitation (ME) module that computes short-term information and a Multiple Temporal Aggregation (MTA) to capture long-term evolution. The ME module integrates spatial and temporal informa-

tion, replacing the expensive computation of optical flow images. MTA is a hierarchical residual architecture that reduces the loss of frames distant in time. Their experiments showed that the method is very competitive and efficient, although it did not outperform the methods pre-trained on Kinetics.

3.3 Summary

Optical flow generally plays a major role in the characterization of the actions, either to estimate object trajectories in DT-based approaches [44, 90, 115, 134, 135, 136, 140], or to be used as input to a 2D CNN like in the multi-stream architectures [5, 15, 28, 113, 138, 142, 144, 157].

The importance of the appearance information may be dependent on the target dataset [67, 154]. For instance, on UCF101 [116] the appearance has shown to be fundamental to distinguish similar classes, whereas on Something-Something [37] the temporal information seems to be more important. This was observed by Zhou et al. [154], who classified the videos from both datasets in the original and shuffled order. They noted that the shuffling affected the Something-Something dataset rather more than the UCF101. In fact, the latter presented no difference in performance between the two versions.

The combination of complementary information, either features or entire methods, usually leads to better performance. This can be seen in multi-stream networks and methods combined with the iDT [136] or I3D [15]. How to balance between the sources is a difficult matter that is recurrently discussed in the literature works.

Deep networks are employed in most of the current works. However, there is no consensus regarding the use of 2D/3D convolutions or recurrent networks (e.g. LSTM) to capture temporal relations. Although 3D CNNs and LSTM cells seem to be more appropriate for the problem, many methods achieve good results exploring frame-level information.

Another important issue for the action recognition problem is the temporal extension of the method, and it is also dependent on the dataset. On datasets like KTH [109], a small number of frames might be sufficient, since it contains a few cyclic actions ("boxing", "hand clapping", "hand waving", "jogging", "running", "walking"). However, the UCF101 might require more frames to distinguish between "Basketball" and "BasketballDunk" [157], for instance.

In our work, we proposed a multi-stream architecture based on spatial and temporal streams [113]. Similar to Wang et al. [138] and Bilen et al. [5], we included new streams to provide complementary information to the original ones. For the new streams, we explored the visual rhythm that works as a frame-level aggregation in the form of an image. As such, unlike the aggregation methods presented here, we can explore 2D CNNs to capture long-term information through the identification of temporal patterns in the visual rhythms. We also provided a study of adaptable fusion methods for multi-stream architectures to avoid fixed weights that are specific to the involved streams and datasets.

Chapter 4

Benchmarks

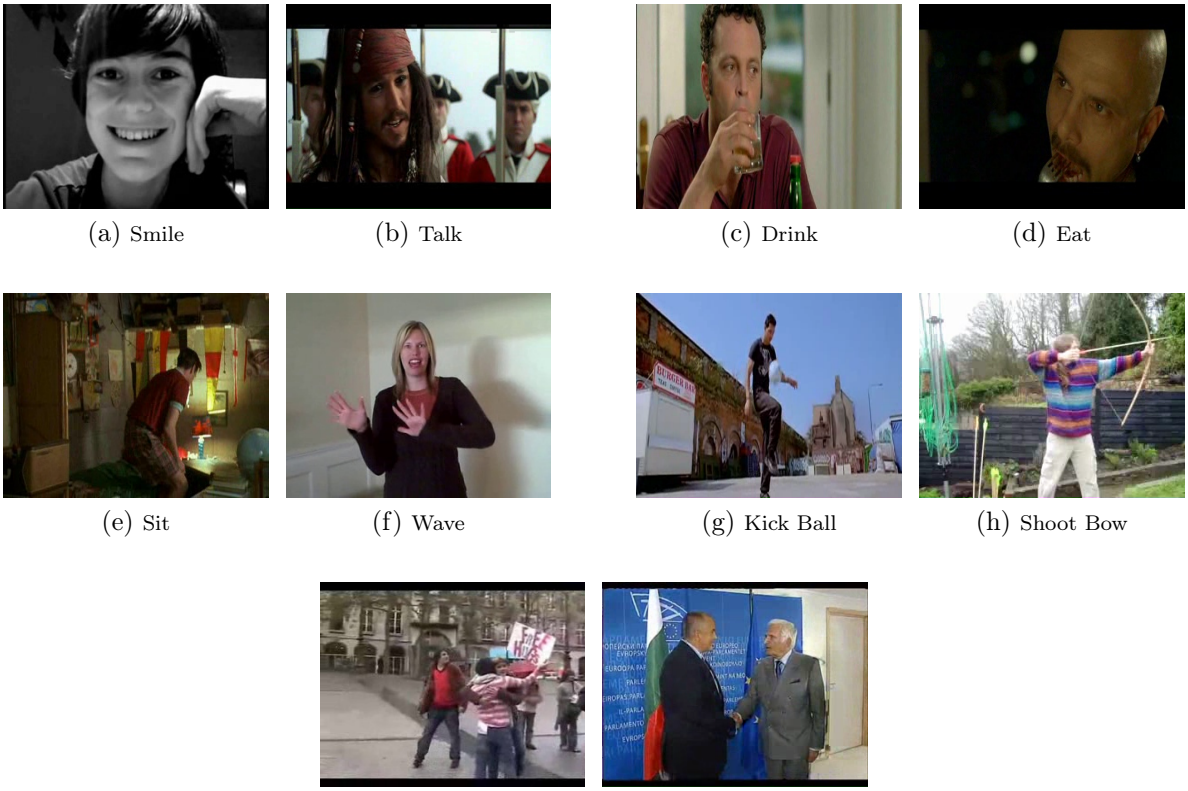
Over the last years, we have seen an increasing interest in more realistic and larger datasets with video samples obtained in less controlled scenarios. From this group, we selected HMDB51 [55], UCF101 [116] and Kinetics [49] datasets. The sequences from them were not recorded specifically for the human action recognition problem; they were extracted from different sources, such as movies and YouTube. For this reason, they present a great variety of scene conditions: different illumination conditions, cameras, actors, quality and points of view, indoor/outdoor videos, among others. The actions in these datasets may involve multiple actors and other objects, and there may be other events occurring in the background.

4.1 HMDB51

By the time the HMDB51 (Human Motion DataBase [55]) was publicly released, many methods from the literature had already achieved almost perfect scores on the datasets that were more popular back then, such as KTH [109] and Weizmann [7]. For this reason, the main goal of the authors was to propose a challenging and rich dataset with complex actions that require more than just static information such as pose and silhouette to be recognized.

The HMDB51 is composed of 6766 sequences extracted from various sources, mostly from movies. Since it combines commercial and non-commercial sources, it presents a rich variety of sequences, including blurred videos or with lower quality and actions from different points of views. The samples are categorized into 51 action classes with at least 101 clips each. The actions can be one out of five types: (i) General Facial Actions, (ii) Facial Actions with Object Manipulation, (iii) General Body Movements, (iv) Body Movements with Object Interaction and (v) Body Movements for Human Interaction. Figure 4.1 presents some examples of HMDB51 classes. The complete list of action classes along with their respective indices used here is given in Appendix B.

The authors provide a recommended split of the samples, where each split contains exactly 70 samples for training and 30 for testing per action class. They also provided a clip-level annotation, or meta-tags, that indicates the main characteristics of the videos. Five aspects are considered in this annotation:



(a) Smile (b) Talk (c) Drink (d) Eat

(e) Sit (f) Wave (g) Kick Ball (h) Shoot Bow

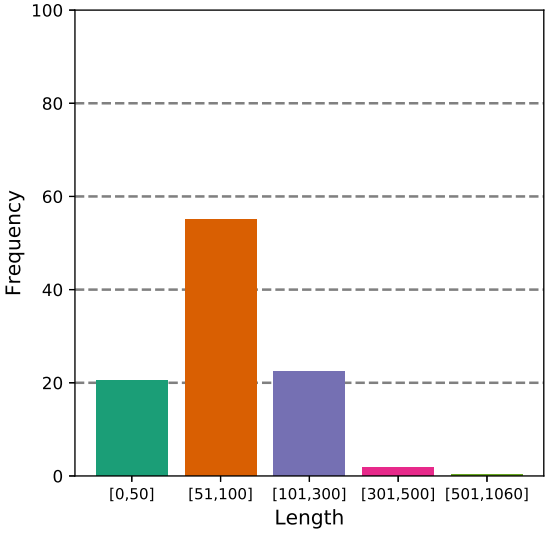
(i) Hug (j) Shake Hands

Figure 4.1: Examples of HMDB51 [55] class types. (a,b) General Facial Actions; (c,d) Facial Actions with Object Manipulation; (e,f) General Body Movements; (g,h) Body Movements with Object Interaction; (i,j) Body Movements for Human Interaction.

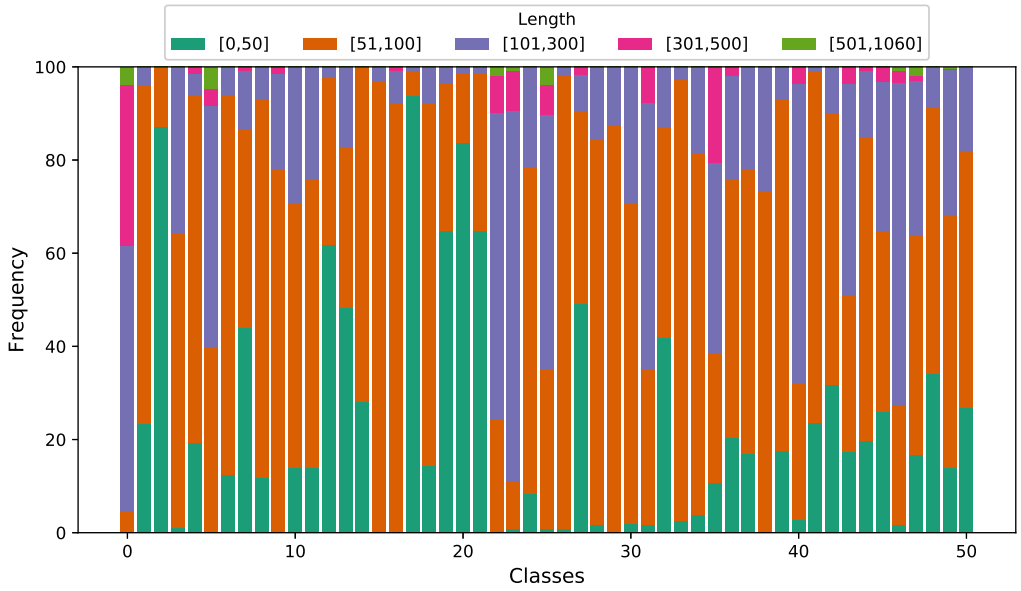
1. Visible body part: head, upper body, full body, lower body;
2. Camera motion: moving or static camera;
3. Camera view point (relative to the actor): front, back, left or right;
4. Number of people involved in the action: single, two or three;
5. Clip quality: high, medium or low.

These aspects were considered in order to generate balanced sets. Moreover, they are useful for a qualitative analysis of the methods. A stabilized version of HMDB51 (i.e. without camera motion) is also available, but it is not used in our research. Approximately 60% of the videos in the original version contain camera motion, which represents an important challenge for the action recognition methods. Most of the current datasets include this challenge and, by taking it into account, we can produce more robust and realistic methods.

The HMDB51 clips were normalized so that all of them have a fixed height of 240 pixels and 30 fps. The width was scaled accordingly to maintain the aspect ratio. The clip lengths vary from 17 to 1061 frames, with an average of 92 frames per video (about 3 seconds, considering 30 fps). Figure 4.2 shows the distribution of lengths on HMDB51, including the distribution among the classes.



(a) Overall Distribution



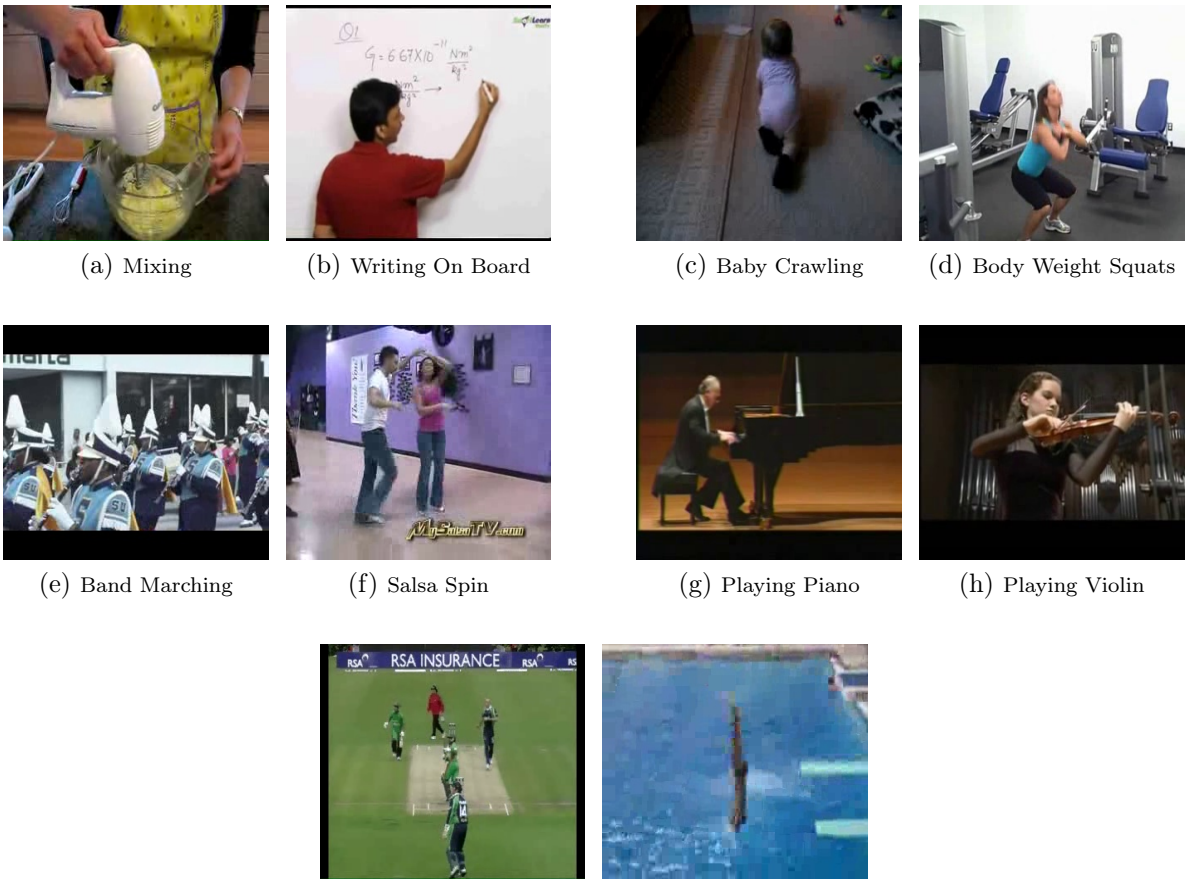
(b) Class-level Distribution

Figure 4.2: Length frequency on HMDB51 dataset.

From Figure 4.2a, we can see that a significant part of clips (almost 80%) contains fewer than 100 frames, and 20% does not exceed even 50 frames. The class-level distribution (Figure 4.2b) shows that this proportion varies considerably throughout the classes. For instance, in the classes 2 and 17 (corresponding to the Catch and Hit actions, respectively), the clips are predominantly short. The class Brush Hair (0), on the other hand, contains longer clips, with about 95% of the videos having more than 100 frames.

4.2 UCF101

UCF has proposed some datasets for human action recognition collected from YouTube (UCF Sports [100], UCF11 [68], UCF50 [98], UCF101 [116]). Each one is an extended version of the previous. The latest one is the UCF101, which contains 13320 sequences divided into 101 classes. Appendix B also contains a complete list of UCF101 classes. There are 5 types of classes on UCF101: (i) Human-Object Interaction, (ii) Body-Motion Only, (iii) Human-Human Interaction, (iv) Playing Musical Instruments and (v) Sports. Some examples of each type are illustrated in Figure 4.3.



(a) Mixing (b) Writing On Board (c) Baby Crawling (d) Body Weight Squats

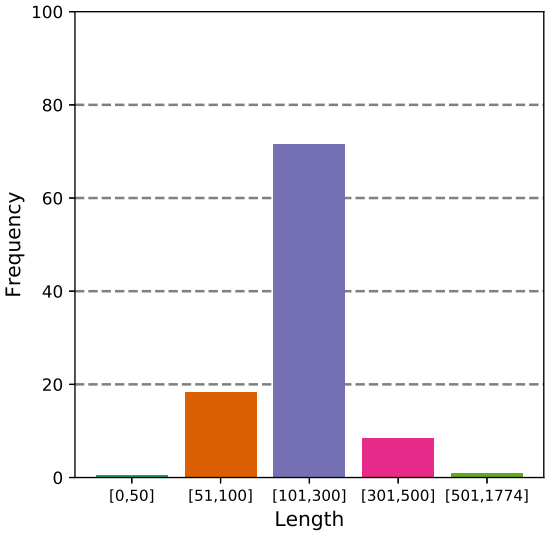
(e) Band Marching (f) Salsa Spin (g) Playing Piano (h) Playing Violin

(i) Cricket Shot (j) Diving

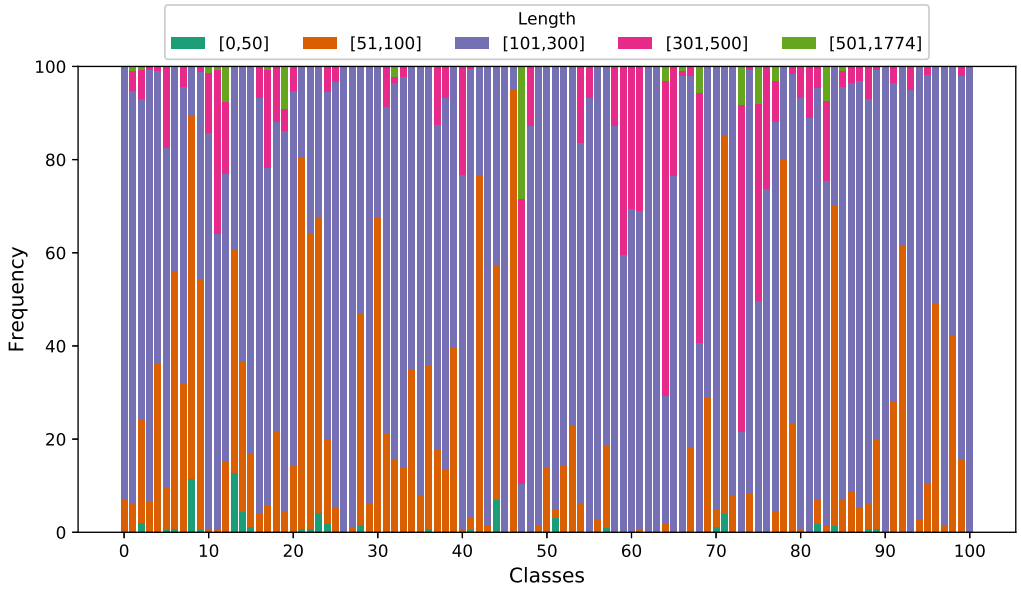
Figure 4.3: Examples of UCF101 [116] class types. (a,b) Human-Object Interaction; (c,d) Body-Motion Only; (e,f) Human-Human Interaction; (g,h), Playing Musical Instruments (g,h); (i,j) Sports.

The samples are not evenly distributed among the classes. Every action class contains 25 groups of 4-7 clips that share some features, resulting in 100-175 clips per class. The samples have a fixed resolution of 320×240 pixels, frame rate of 25 fps and lengths ranging from 1.06 to 71.04 seconds, with an average length of 7.21s (about 180 frames) per clip. The charts in Figure 4.4 illustrate the distribution of lengths on UCF101. In contrast to HMDB51, almost 72% of UCF101 clips have between 101 and 300 frames and the proportion of very short and very large videos are negligible (less than 1% each), as shown in Figure 4.4a. Although the purple color is predominant in the second chart

(Figure 4.4b), many classes present a considerable number of clips lying in the interval $[51, 100]$ (for instance, classes 8 - Basketball Dunk and 46 - Jumping Jack), whereas other classes present a significant percentage in the interval $[301, 500]$ (e.g. 47 - JumpRope and 73 - Rock Climbing Indoor).



(a) Overall Distribution

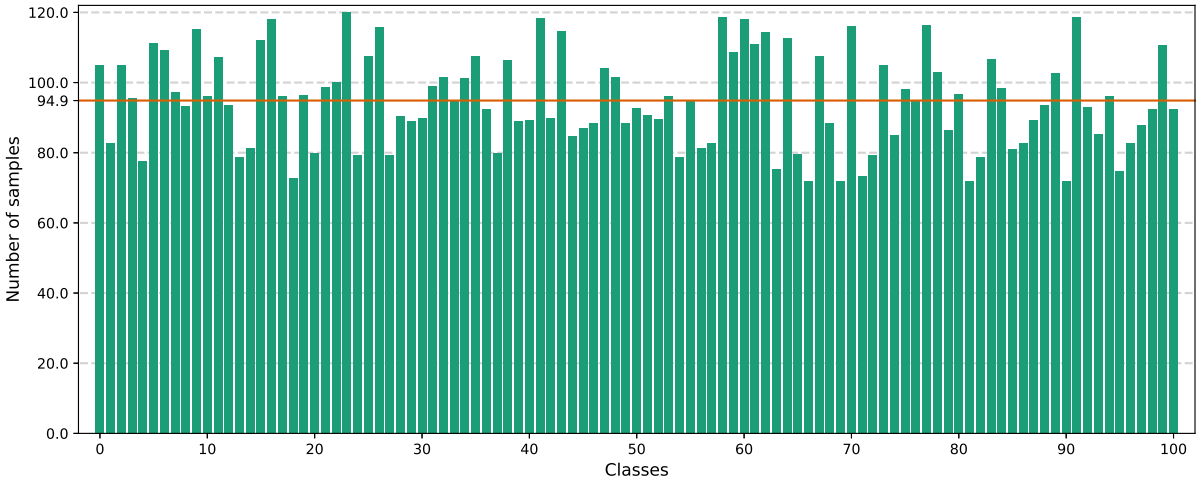


(b) Class-level Distribution

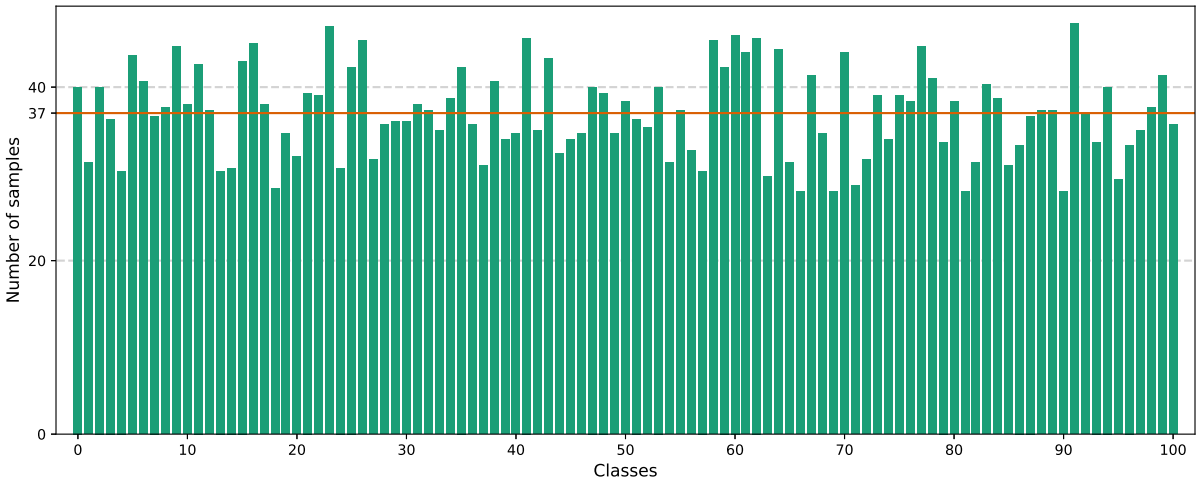
Figure 4.4: Length frequency on UCF101 dataset.

The dataset also includes recommended splits into approximately 70-30 for training and testing, respectively, without validation set. Figure 4.5 shows the average of clips over the splits for each class. The classes have an average of 94.9 clips for training and 37 for test. In both sets, the largest class is about 1.7 times greater than the smallest. The number of clips per class lies in the interval $[72, 120]$ for training and $[28, 48]$ for test.

Additional annotation is provided to allow qualitative analysis of the methods. It



(a) Training set



(b) Test set

Figure 4.5: Number of clips per class averaged over the splits for training and test sets on UCF101. The horizontal brown line indicates the average number of clips over the classes.

consists of 115 class-level attributes manually defined, for instance, "Number of People" and "Body Parts Used". The clips of 51 classes contain the original audio, but this information is not used in our research.

4.3 Kinetics

With the purpose of supplying the absence of large datasets to train video-based deep networks, Kay et al. proposed the challenging DeepMind Kinetics datasets in three versions: Kinetics-400 [49], Kinetics-600 [13] and Kinetics-700 [14]. Each version is almost a superset of the previous, except for some class removals to avoid action redundancies and splits into more specific labels. The main goal was to introduce a dataset for action recognition comparable to the ImageNet for image classification.

The clips were extracted from YouTube and, consequently, they consist of amateur videos that present realistic and potentially challenging conditions. Each clip is around 10-second long, with various resolutions and frame rates.

Table 4.1 contains some statistics of the three versions. The number of samples may vary according to the video availability on the YouTube. We can see that the total number of videos in all versions is considerably larger than on HMDB51 and UCF101 datasets. Moreover, every Kinetics clip is taken from a distinct YouTube video, so it comprises a wider variety of scenarios.

Table 4.1: Kinetics statistics. "Train", "Validation" and "Test" columns represent the number of clips per class assigned for each set. "Total" and "Classes" refer to the total number of clips and number of classes in the corresponding version. Adapted from [14].

Version	Train	Validation	Test	Total	Classes
Kinetics-400 [49]	250–1000	50	100	306245	400
Kinetics-600 [13]	450–1000	50	100	495547	600
Kinetics-700 [14]	450–1000	50	100	650317	700

The number of clips per class in the training set of Kinetics-600 is shown in Figure 4.6. Kinetics-600 has an average of 624 clips per class, and the largest class is about 2.7 times greater that the smallest.

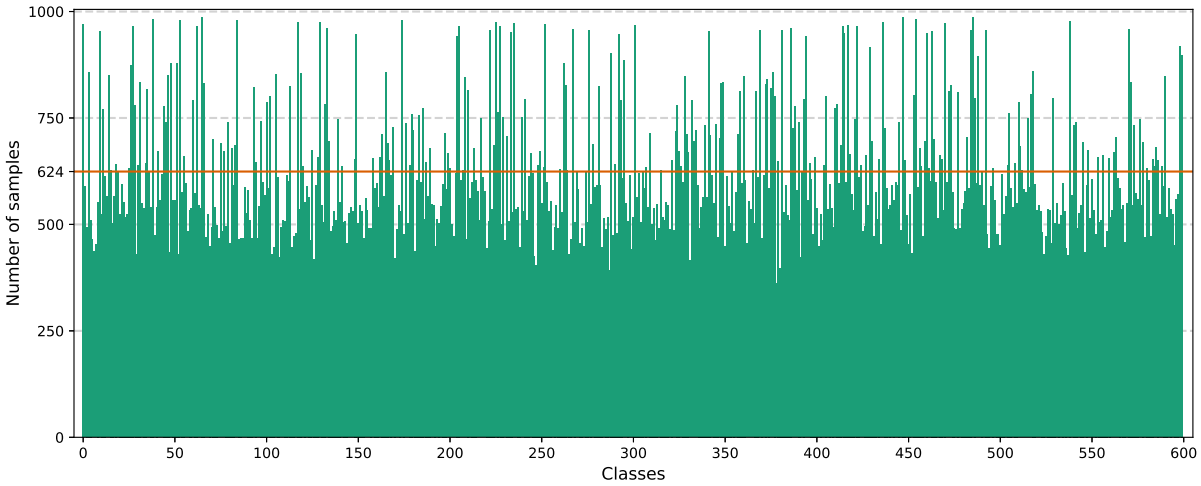


Figure 4.6: Number of clips per class in the training set of Kinetics-600. The horizontal brown line indicates the average number of clips over the classes.

There are three types of actions illustrated in Figure 4.7: (i) Singular Person, (ii) Person-Person and (iii) Person-Object. The third type includes actions of the same verb with different object such as "Reading Book" and "Reading Newspaper", and different verbs with the same object, such as "Playing Guitar" and "Tapping Guitar". The classes are also categorized into groups according to the action purpose or scenario, for instance, "Communication" and "Snow + Ice" are two superclasses of Kinetics. This categorization is non-exclusive, that is, the same class can belong to multiple groups. The class "Getting a Tattoo", for example, belongs to "Arts and Crafts" and "Makeup".

(a) Hugging Baby

(b) Shaking Hands

(c) Reading a Book

(d) Talking on Cellphone

(e) Laughing

(f) Stretching Arm

Figure 4.7: Examples of Kinetics-600 [13] class types. (a,b) Singular Person; (c,d) Person-Person; (e,f) Person-Object.

4.4 Other Datasets

Several other datasets were proposed for the action recognition problem. Table 4.2 presents a summary of some popular ones. The first three (KTH, Weizmann and Hollywood2) were widely used before the deep learning era. However, due to the reduced number of samples and the lack of realism, especially on KTH and Weizmann, these datasets became deprecated, giving place to larger datasets such as HMDB51 and UCF101.

For our research, we consider only the datasets with trimmed clips and a single label per clip. Since we are working with deep networks, large and realistic datasets are also preferable. Therefore, Something-Something, Epic Kitchens and Moments in Time would match our requirements, as well as HMDB51, UCF101 and Kinetics-600. However, Something-Something and Epic-Kitchens comprise more restricted scenarios (human-object and cooking-related actions, respectively), and Moments in Time includes non-human actions.

Table 4.2: Summary of some popular datasets for action recognition. The table contains the dataset name, release year, number of classes, number of distinct videos, number of clip labels (single label)/temporal annotations (multilabel), whether is multilabel and trimmed, and the dataset keywords.

Dataset	Year	# Classes	# Videos	# Labels	Multilabel	Trimmed	Keywords
KTH [109]	2004	6	599	2.4K	N	Y	Controlled actions
Weizmann [7]	2005	10	93	93	N	Y	Controlled actions
Hollywood2 [73]	2009	12	69	1.7K	N	Y	Movies
HMDB51 [55]	2011	51	–	6.8K	N	Y	Movies
UCF101 [116]	2012	101	–	13.3K	N	Y	YouTube
Sports1M [48]	2014	487	1.1M	1.1M	Y	N	Sports
ActivityNet [29]	2015	203	27.8K	39.2K	Y	N	Activity variety
Charades [112]	2016	157	10K	67K	Y	Y	Daily activities
Something-Something [37]	2017	174	108.5K	108.5K	N	Y	Human-object interaction
AVA [38]	2018	80	430	1.6M	Y	Y	Atomic actions
Epic Kitchens [27]	2018	149	432	39.6K	N	Y	Cooking
Kinetics-600 [13]	2018	600	500K	500K	N	Y	ImageNet-inspired
Moments in Time [76]	2019	339	1M	1M	N	Y	Event understanding

Chapter 5

Adaptive Visual Rhythm

Our basic method consists of a three-stream architecture composed of the spatial, temporal and VR-based streams (Figure 5.1). Each stream consists of an image-based CNN. The first two are based on the two-stream network [113]. They work with RGB frames to capture appearance information and optical flow images to capture short-term motion. The third stream is the main contribution of our work and is based on a long-term feature, the visual rhythm. Each stream is individually trained and their m-dimensional score vectors are fused during the test stage, where m is the number of classes. We considered different approaches to our proposed stream and for fusion that are described in this chapter and in the following ones.

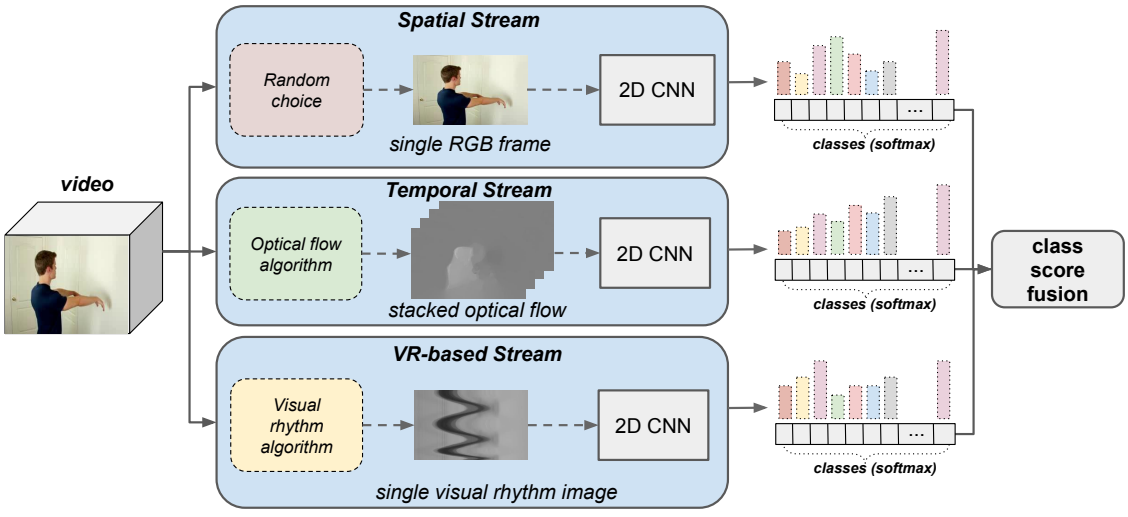


Figure 5.1: Overview of our three-stream basic method for action recognition. Adapted from [24].

5.1 Adaptive Visual Rhythm

Our first strategy for the visual rhythm stream is called AVR (Adaptive Visual Rhythm). It was the result of a collaborative project published on the ICMLA 2018 [24] and was part of Concha's [23] Master's thesis.

In the AVR strategy, the rhythms for a given video $V = \{F_1, F_2, \cdots, F_t\}$ are hand-crafted images based on the following operations proposed by Souza [117] (recall the visual rhythm definition in Section 2.3):

$$T_{H'}(F_i) = \left[\frac{\sum_y F_i(y,1)}{h} \; \frac{\sum_y F_i(y,2)}{h} \; \cdots \; \frac{\sum_y F_i(y,w)}{h} \right]^T \tag{5.1}$$

$$T_{V'}(F_i) = \left[\frac{\sum_x F_i(1,x)}{w} \; \frac{\sum_x F_i(2,x)}{w} \; \cdots \; \frac{\sum_x F_i(h,x)}{w} \right]^T . \tag{5.2}$$

For the frame F_i, the VR slices $S_{H',i} = T_{H'}(F_i)$ and $S_{V',i} = T_{V'}(F_i)$ are defined as the average of the columns/rows intensities (Figure 5.2).

Figure 5.2: Examples of horizontal-mean and vertical-mean slices, extracted from [24]. The slices are defined as the average of columns/rows. They were resized for illustration purposes.

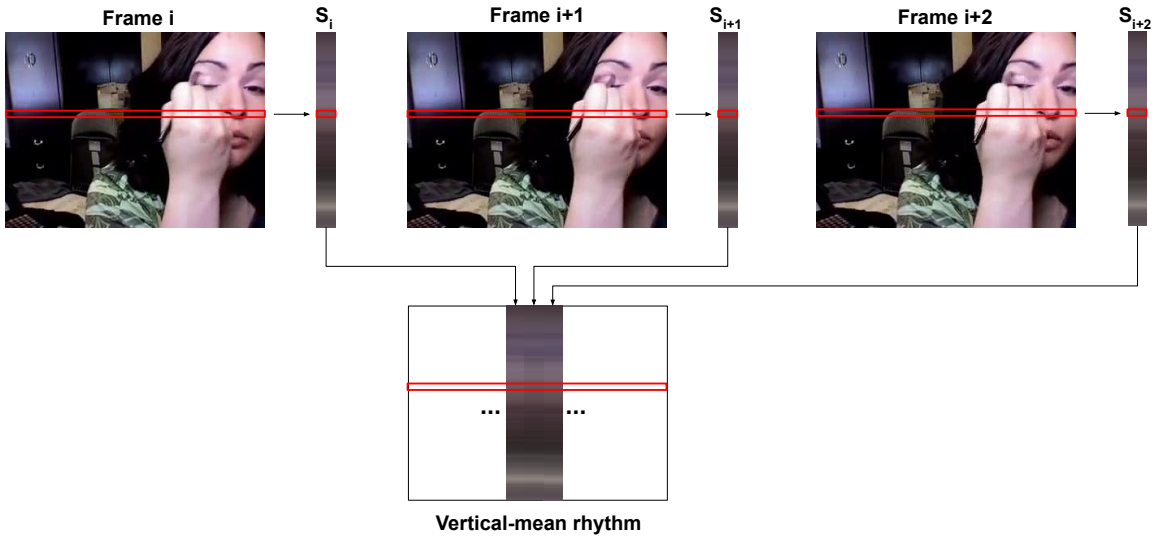


Figure 5.3: Example of vertical-mean rhythm. The j^{th} row of the vertical-mean rhythm represents the evolution of the average value of the j^{th} row in the video over time. The slices were resized for illustration purposes.

Each slice is a $w \times 1$ ($S_{H',i}$) or a $h \times 1$ ($S_{V',i}$) vector. The resulting rhythms are $w \times t$ and $h \times t$ images formed by the concatenation of the slices and are called horizontal-mean and vertical-mean:

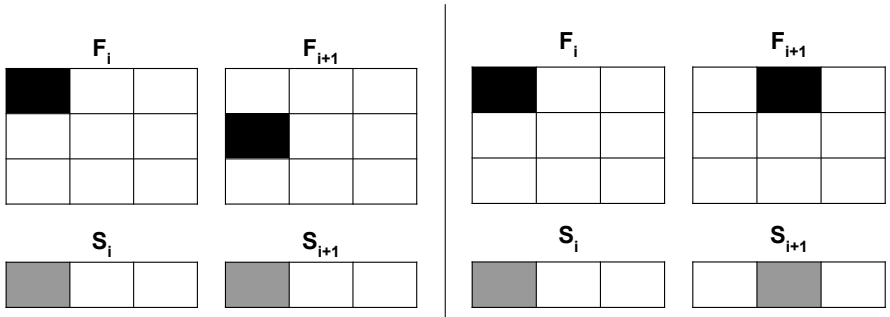
$$\mathrm{VR}_{H'}(V) = [T_{H'}(F_1)\ T_{H'}(F_2)\ \cdots\ T_{H'}(F_t)] \qquad \text{(horizontal-mean)}, \tag{5.3}$$

$$\mathrm{VR}_{V'}(V) = [T_{V'}(F_1)\ T_{V'}(F_2)\ \cdots\ T_{V'}(F_t)] \qquad \text{(vertical-mean)}. \tag{5.4}$$

Note that the slices are placed in the columns of the rhythms, regardless of the operation used, as mentioned in Section 2.3. As such, the horizontal-mean slice from Figure 5.2 is rotated to be used. Differently from the traditional horizontal and vertical rhythms, the slices in these two use the information of the entire frame.

The j^{th} row in the rhythm image represents the evolution of the average value of the respective j^{th} column/row in the video over time (Figure 5.3). Considering a VR direction D ($D = H'$ or $D = V'$), this evolution row is defined as the set $\{S_{D,1}(j), S_{D,2}(j) \cdots, S_{D,t}(j)\}$ for a fixed column/row j, and is perpendicular to the slices in the rhythm. If the average remains constant for the column/row j, the j^{th} row in the rhythm will form a line with homogeneous intensity, which indicates little or no movement in the corresponding region throughout the video.

Suppose, without loss of generality, that we are working with horizontal-mean slices. If a given object moves vertically (that is, orthogonally to the slice direction) between two frames, it is very likely that the mean color of the corresponding column remains the same (Figure 5.4). However, a horizontal movement affects the average color of all columns spanned by the object. Therefore, movements parallel to the slice direction tend to produce more distinctive patterns.



(a) orthogonal movement (b) parallel movement

Figure 5.4: A moving object considering two consecutive frames and horizontal-mean slices. Parallel movement is better captured in the slice. Extracted from [24].

Based on this observation, we proposed a method for adaptively deciding the best visual rhythm direction for each action according to the predominant movement. For estimating the predominant direction of movement, we use the pyramidal version of Lucas-Kanade point-tracker [10] and the points of interest indicated by the Shi-Tomasi method [111]. The absolute horizontal and vertical displacement estimated by the tracker are accumulated over the frames, and the highest value defines the rhythm direction. Therefore, we choose the horizontal-mean rhythm, if horizontal movement is predominant in the class and the vertical-mean otherwise. This process is only performed once,

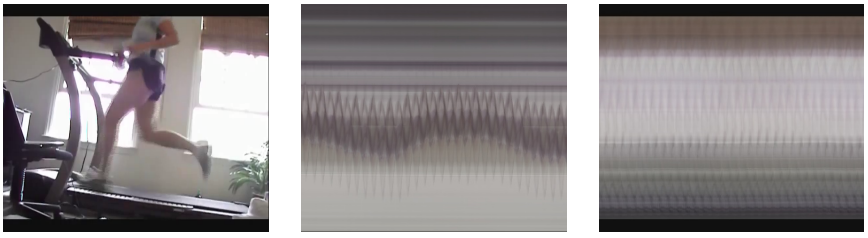
and the determined directions are used for every posterior training.

Algorithm 4 AVR-Decision(V) [24]

Input Video $V = \{F_1, F_2, \cdots, F_t\}$.
Output Visual rhythm direction {1: vertical-mean; 2: horizontal-mean}.

$H \leftarrow 0$ ▷ Initialize the accumulated horizontal displacement.
$V \leftarrow 0$ ▷ Initialize the accumulated vertical displacement.
$P_a \leftarrow goodFeaturesToTrack(F_1)$ ▷ Find key points in F_1.
for all $F_i \in V \setminus \{F_1\}$ **do**:
    $P_b, St \leftarrow PyrLK(F_{i-1}, F_i, P_a)$ ▷ Pyramidal Lucas-Kanade point tracking.
    $P_a, P_b \leftarrow SelectGoodPoints(P_a, P_b, St)$ ▷ Select good points
    $H \leftarrow H + \sum_{j=1}^{n} |P_b[j].x - P_a[j].x|$ ▷ n = size of P_a.
    $V \leftarrow V + \sum_{j=1}^{n} |P_b[j].y - P_a[j].y|$
    $P_a \leftarrow P_b$
if $H \leq V$ **then**
    return 1 ▷ Vertical movement is predominant.
else
    return 2 ▷ Horizontal movement is predominant.

Algorithm 4 presents the steps of the proposed decision method. The function *goodFeaturesToTrack()* [111] determines the key points in the first frame that are used for tracking in the following ones. For each pair of consecutive frames, the *PyrLK()* [10] method is used to estimate the new locations of the key points, and it also returns flags indicating whether the points were found. The horizontal and vertical displacements are accumulated in H and V, respectively. The key points set is updated using the *SelectGoodPoints()* function, based on the flags returned by *PyrLK()*. Finally, the method decides the predominant direction according to H and V. This method is repeated for every video.



(a) frame (b) horizontal-mean (c) vertical-mean

Figure 5.5: Example of frame and rhythms from a Kinetics video of the "running on treadmill" class. The horizontal-mean rhythm presents a wavy pattern that better characterizes the action. Extracted from [70].

Figure 5.5 shows a frame and the visual rhythms extracted from a Kinetics video of the "running on treadmill" class. This action is predominantly horizontal due to the leg motion. For this reason, the horizontal rhythm presents more relevant patterns for the classification. As can be observed in the example, the horizontal rhythm contains a wavy

pattern that represents the leg movements, whereas the vertical one is composed of quite homogeneous lines.

5.2 Improved Spatial Stream

A second contribution of the AVR paper [24] is an improved spatial stream. Instead of collecting a single RGB frame per video, we collect two, one in each half of the video. This approach is justified by the fact that the appearance of the scene may change significantly over time, either by scene conditions as lighting and occlusions or by the variety of poses, objects and background in the video (Figure 5.6). Therefore, a single appearance may not be sufficient to describe the action, since the elements that characterize it may not be apparent in the frame. The improved spatial stream continues to receive one RGB frame at a time, but it is trained with twice as many samples as proposed in the original to capture variations in appearance.

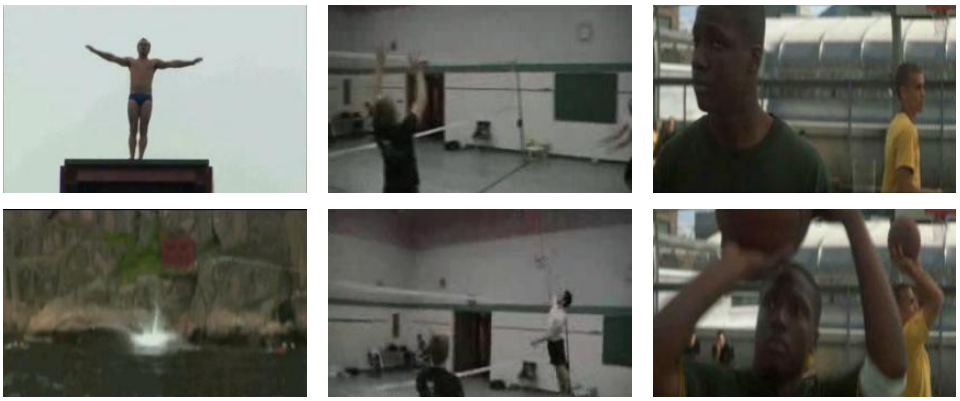


Figure 5.6: Fragments from the UCF101 that show the significant variation in appearance in the two halves of the videos. Each column contains a frame from the first and the second halves of the same video, respectively. Major changes are: the background change in the first video, different actors present in each frame in the second one and the balls are not in the first frame from the third video. Extracted from [24].

Testing protocol remains the same in this stream (Section 2.7): we use 25 frames evenly sampled from each testing video, and 10 new samples are produced from them. Each sample is individually tested, and all the 250 computed outputs are combined through the average of the scores to obtain the m-dimensional vector of the spatial stream.

5.3 Kinetics Pre-training

Many of the state-of-the-art approaches achieve great results by pre-training their networks with a large video dataset, the Kinetics [49]. However, due to Kinetics size, this strategy requires further computational resources to pre-train the three streams, especially to compute the inputs and train the temporal one. The temporal stream already achieves good results using only the ImageNet for pre-training, perhaps because the optical flow maintains a certain level of object shape.

Our visual rhythm, on the other hand, is based on operations that extract a global feature from each frame (Equations 5.1 and 5.2) and so, it does not maintain the silhouettes of the actors and objects (Figure 5.7). It is worth mentioning that not all VR operations completely deform the silhouettes. For instance, recall that in Section 3.2 we presented a work from Moreira et al. [77]. In their work, they noted that the object is transferred from video to VR since they use the diagonal of each frame as slice. However, we prefer the horizontal-mean and vertical-mean approaches since they are not dependent on the object location and represent the entire frame. Due to this deformation, the VR is very different from natural images and so its corresponding stream may take greater benefit from the additional pre-training step using the same modality, in addition to the ImageNet initialization. Therefore, we developed a different pre-training procedure for the third stream using rhythms computed from Kinetics, improving the original AVR results. This work was accepted as a chapter of the Deep Learning Applications book (DLAPP2019) [70] composed of expanded versions of ICMLA 2018 selected papers.

Figure 5.7: Modalities used in our three-stream network. The object silhouette can be seen in RGB image and optical flow, but not in visual rhythm. Extracted from [70].

5.4 Stacking

In our experiments, we noticed that an improvement in the individual streams does not necessarily imply in an improvement in their combination through the multi-stream strategy. For this reason, a good fusion strategy is fundamental for the method effectiveness.

In the extension of the AVR paper [70], in addition to the simple and weighted average fusion [24,113,141], we explore another fusion strategy using external fully connected (FC) layers as a meta-classifier. Thus, the network automatically defines how much the features contribute to the final prediction. The external network is trained using the same training set as the 2D CNNs, but using the features computed by them as input (Figure 5.8). This combination of classifiers is called stacking.

The training procedure is divided into two stages: (1) 2D CNNs and (2) meta-classifier training. The input to the meta-classifier is formed by the concatenation of the streams outputs. The output is an m-dimensional vector with the class scores.

The idea behind this proposal is to learn misclassification patterns from the streams combined with the others. That is, if two classes are poorly discriminated in a given stream, but well classified in another one, the external network may capture this pattern. The main advantage of this method is the automatic weight assignment that adapts to the inclusion of new streams and modifications in the approaches.

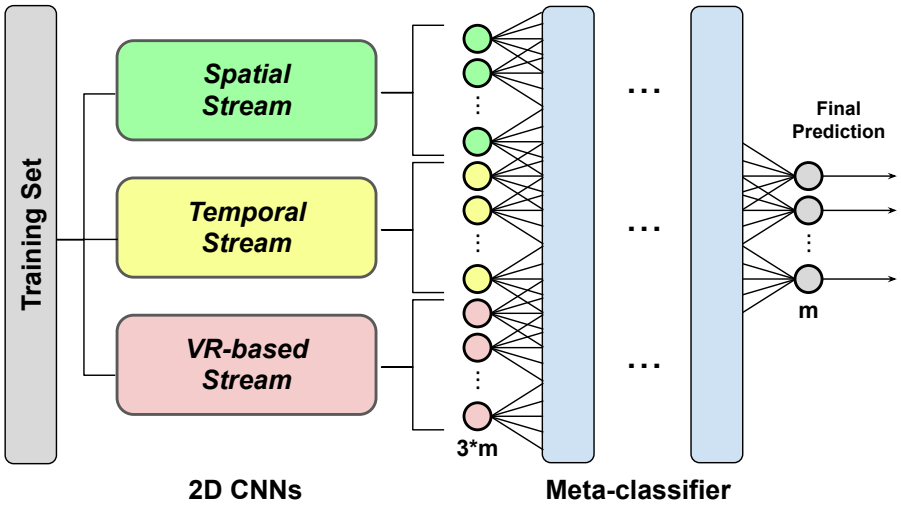


Figure 5.8: Stacking: the vectors computed by the streams are fused using fully connected layers to obtain the final prediction for each video. The sizes of the input and output are based on the number of classes in the dataset. Extracted from [70].

Our first meta-classifier proposal consists of a single FC layer. It is composed of m neurons corresponding to the classes. The operation in each neuron can be seen as a weighted average of the $3 \cdot m$ scores with trainable parameters, followed by a Rectified Linear Unit (ReLU) activation. Differently from the weighted average, in this approach all the classes predictions are considered to compute each score. In total, the layer contains $3 \cdot m^2$ trainable weights.

5.5 Results

The baseline for our work was provided by Zhu [160], which is a public implementation of the very deep two-stream network [141] using PyTorch framework. The Inception V3 [123] architecture was adopted as CNN for our three streams using the ImageNet parameters for initialization provided by PyTorch. We report the results of the AVR stream with and without the extra pre-training on the Kinetics dataset, performed after the ImageNet initialization.

The networks are trained using the Stochastic Gradient Descent (SGD) optimizer with Nesterov momentum [122] and the cross-entropy loss. For either the pre-training on Kinetics and training on UCF101/HMDB51, we use 250 epochs and learning rate of 0.001. Since the Kinetics contains a validation set, we use it to determine the best pre-trained model for the posterior UCF101/HMDB51 training. According to our experiments, the best pre-trained model was the one at the 50-th epoch. The UCF101 and HMDB51 datasets, on the other hand, do not provide a validation set, and so we use the model of the last epoch for testing.

All experiments were performed on a machine with an Intel® Core™ i7-3770K 3.50GHz processor, 32GB of memory, an NVIDIA GeForce® GTX 1080 GPU and Ubuntu 16.04.

The results reported on the AVR paper [24] are shown in Table 5.1. The table includes

different approaches for each stream and every combination of the best approaches. The approaches were tested on UCF101 and HMDB51, using the Inception V3. Note that, the improved spatial stream (RGB*) outperformed the original approach (RGB) in the two cases. Regarding the VR-based stream, we tested four approaches: using each direction separately, using both directions stacked in a 2-channel image and the AVR. AVR presented the best results. It is worth mentioning that the combination of both directions achieves the lowest accuracies, even compared to individual ones. Although each video is better represented by a specific direction, in the stacked version the presence of the second one had an adverse effect on the efficacy. Due to the rhythms constructions, the pixels at the same position of horizontal-mean and vertical-mean images do not represent the same video region. Thus, the stacked image associates pixels that are not related through the channels, which may explain the decrease in accuracy compared to individual results. AVR improves the performance using only one chosen direction per video. In both datasets, the temporal stream achieves the best individual results, as in the original two-stream network.

Table 5.1: Results and comparison of different approaches used for each stream and combinations. Cells on bold represents the overall highest accuracy rates, whereas underlined cells consist of the best individual approaches.

Stream	Approach	UCF101	HMDB51
Spatial	RGB image	86.09	50.76
	RGB* images	<u>86.61</u>	<u>51.77</u>
Temporal	Optical flow	<u>86.95</u>	<u>59.91</u>
VR-based	Horizontal - mean	62.37	35.57
	Vertical - mean	55.16	30.27
	Stacked	48.65	29.74
	AVR	<u>64.39</u>	<u>39.63</u>
Combination	RGB* images + AVR	90.74	61.31
	RGB* images + optical flow	92.94	66.43
	Optical flow + AVR	89.23	65.45
	RGB* image + optical flow + AVR	**93.74**	**69.98**

For the weighted average fusion, we use the weights 2 for the spatial, 3 for the temporal and 1 for the AVR stream, based on the individual performances. Note that RGB* + AVR and optical flow + AVR outperform individual RGB* and optical flow. Therefore, AVR provides complementary information for the network by encoding long-term dynamics. The worst results were achieved by the combination of two temporal features (optical flow + AVR), suggesting that appearance is very relevant for the recognition. The combination of the three streams outperforms the others, therefore, all the three features contribute to the recognition process.

Results for the AVR stream with (AVR-K) and without (AVR) the pre-training on Kinetics are shown in Table 5.2. The extra pre-training step improves the results in both

datasets, especially HMDB51 that has an increase of 9.28%. This may be explained by the dataset size: since HMDB51 is half the size of UCF101 and is more challenging, it takes greater benefit from the pre-training.

Table 5.2: AVR stream results using different pre-training datasets and the Inception V3.

Approach	Pre-training Dataset	UCF101 (%)	HMDB51 (%)
AVR	ImageNet	64.74	39.63
AVR-K	ImageNet+Kinetics	**66.68**	**48.91**

In Table 5.3, we show the results for each of the three streams separately and every combination of them using different fusion strategies. For the combination, we consider the simple and weighted averages of the output vectors, and stacking with a meta-classifier. In the weighted average, we maintained the weights 2, 3 and 1 for the spatial, temporal and VR-based stream, respectively. Our meta-classifier is composed of a single FC layer that receives the concatenation of the stream vectors as input, and returns a vector with the scores for each class.

Table 5.3: Different strategies for fusing the outputs of the three streams.

Approach	Streams	UCF101 (%)	HMDB51 (%)
Individual	RGB* Images	86.61	51.77
	Optical Flow	86.95	59.91
	AVR-K	66.68	48.91
Simple Average	RGB* + Optical Flow	93.08	65.03
	RGB* + AVR-K	87.37	62.00
	Optical Flow + AVR-K	82.66	61.55
	RGB* + Optical Flow + AVR-K	91.97	67.65
Weighted Average	RGB* + Optical Flow	93.06	65.80
	RGB* + AVR-K	90.73	63.12
	Optical Flow + AVR-K	88.52	66.43
	RGB* + Optical Flow + AVR-K	**93.91**	**70.07**
Stacking	RGB* + Optical Flow	92.35	65.05
	RGB* + AVR-K	88.58	61.70
	Optical Flow + AVR-K	85.79	60.13
	RGB* + Optical Flow + AVR-K	91.94	67.14

Concerning the two-stream combinations, most of them surpass the individual results, except for optical flow + AVR-K with simple average and stacking on the UCF101. In these two cases, the combinations perform worse than optical flow alone. In general, the presence of the spatial stream boosts the accuracies in comparison with optical flow + AVR-K, which again suggests the importance of appearance information for the recognition.

For the HMDB51, the three-stream versions outperform all other combinations, whereas for the UCF101 this can only be verified for the weighted average fusion. However, the gain on the challenging HMDB51 (approximately +3%) is more significant than

the loss on the UCF101 (approximately -1%). We can conclude from this table that the weighted average of the three outputs outperforms the other fusion strategies. A comparison with the state-of-the-art is shown in Table 6.6 at the end of Chapter 6.

Although the table shows that the stacking approach did not achieve good results, it still has room for improvement since there are other architectures and hyperparameters to be explored. In addition, among all fusion methods, the stacking is the only trainable one and, therefore, it is the most affected by stream overfitting. In other words, it cannot estimate good weights if the individual streams achieve 100% training accuracy. Overfitting will be further discussed in Chapter 7.

Chapter 6

Learnable Visual Rhythm

In this chapter, we introduce a new VR-based stream called Learnable Visual Rhythm (LVR). We present three versions of the LVR referred to as LVR_0, LVR_1 and LVR_2, all of them are described in Section 6.1. These versions can be used in the multi-stream network from Figure 5.1. In Section 6.2, we present our results using these streams and a comparison to some state-of-the-art methods.

6.1 Learnable Visual Rhythm

Image-based networks have achieved great results for image classification, describing objects and appearance. For this reason, they are good tools for obtaining frame-level descriptors. Therefore, in our second proposal, we use a 2D CNN as the operation $T(\cdot)$ to produce the rhythm slices. This VR-based stream is called Learnable Visual Rhythm (LVR), thanks to their trainable operation. In addition to this feature-extractor CNN, the LVR stream is composed of a second one that predicts the action from the produced rhythm, similar to the CNN from AVR. Each CNN consists of an Inception V3 network pre-trained on ImageNet, and only the second one is fine-tuned on each video dataset. Figure 6.1 illustrates the LVR stream.

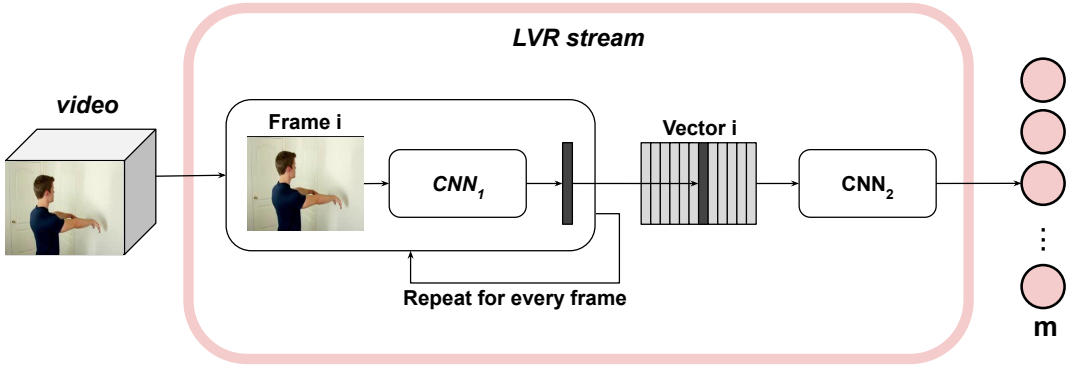


Figure 6.1: LVR stream composed of two stacked CNNs. The first one computes a 1D descriptor for each video frame. The second one predicts the action based on the 2D concatenation of the descriptors. The output of the latter CNN is an m-dimensional score vector, the final LVR response. Adapted from [72].

To generate the slices, we consider three distinct points of the Inception network, called LVR_0, LVR_1 and LVR_2 (Figure 6.2). Each level results in a different VR image. The three positions are also shown in Table 6.1, along with the corresponding input and feature vector sizes. This strategy enables the analysis of different abstraction levels, from the lowest LVR_0 to the highest LVR_2.

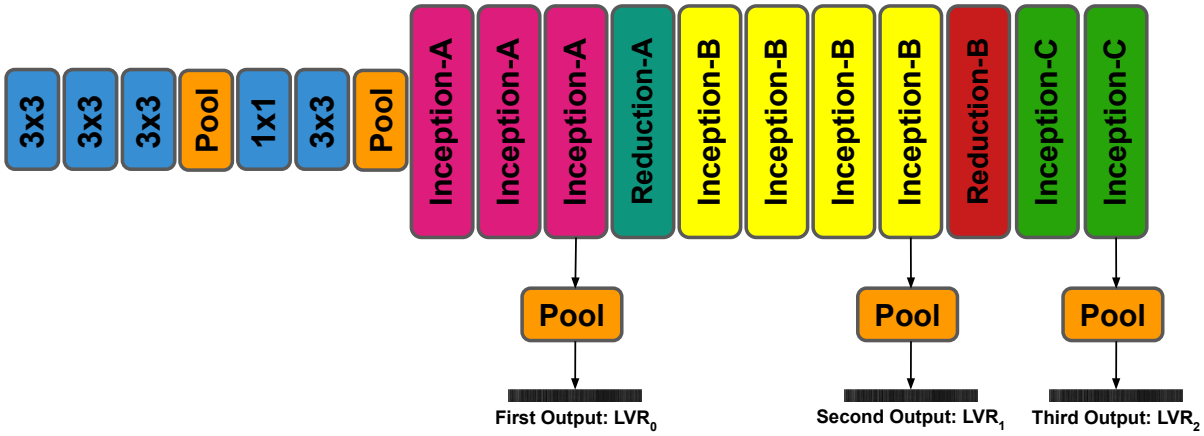


Figure 6.2: An illustration of the three distinct positions considered to extract the features in the Inception architecture (Figure 2.5). Each position represents a different level of abstraction. These features compose the LVR_0, LVR_1 and LVR_2 rhythms.

Table 6.1: Inception V3 architecture with extra pooling layers to produce intermediate outputs. Magenta, yellow and green rows represents LVR_0, LVR_1 and LVR_2 feature vectors, respectively. The pooling layers do not affect original connections.

Type	**Input Size**	**Feature Vector Size**
conv	299×299×3	-
conv	149×149×32	-
conv	147×147×32	-
pool	147×147×64	-
conv	73×73×64	-
conv	73×73×80	-
pool	71×71×192	-
3×Inception-A	35×35×192	-
pool	35×35×288	3528
Reduction-A	35×35×288	-
4×Inception-B	17×17×768	-
pool	17×17×768	3468
Reduction-B	17×17×768	-
2×Inception-C	8×8×1280	-
pool	8×8×2048	2048

Average pooling layers and reshape method are used to reduce the size of the intermediate images and collapse it into one dimension. The connections between the original

layers are maintained, therefore, the extra pooling layers do not affect the results of the following ones. The resulting feature vectors are normalized using min-max normalization, which helps mapping them into a grayscale image.

The rationale for using intermediate outputs is that networks trained for image classification tend to be invariant to the object position and poses at the latter layers, but this information is rather relevant to distinguish actions. The former outputs, on the other hand, represent less refined information. Since the three outputs can be computed in a single forward propagation through the network, they can be combined at a minimum extra computational cost for the extractions.

The first CNN is initialized with ImageNet parameters and frozen to be used as feature extractor. We use the same 10 data augmentation techniques from spatial test in the inputs to this CNN, that is, 5 crops (4 corners + 1 center crop) with and without horizontal flipping. The cropping width and height are both equal to 299, the Inception input size. Thus, for each abstraction level, we produce 10 distinct slices, resulting in 30 slices per frame. The corresponding slices, that is, the slices with common data augmentation technique and Inception depth, are concatenated to form 30 different visual rhythms per video.

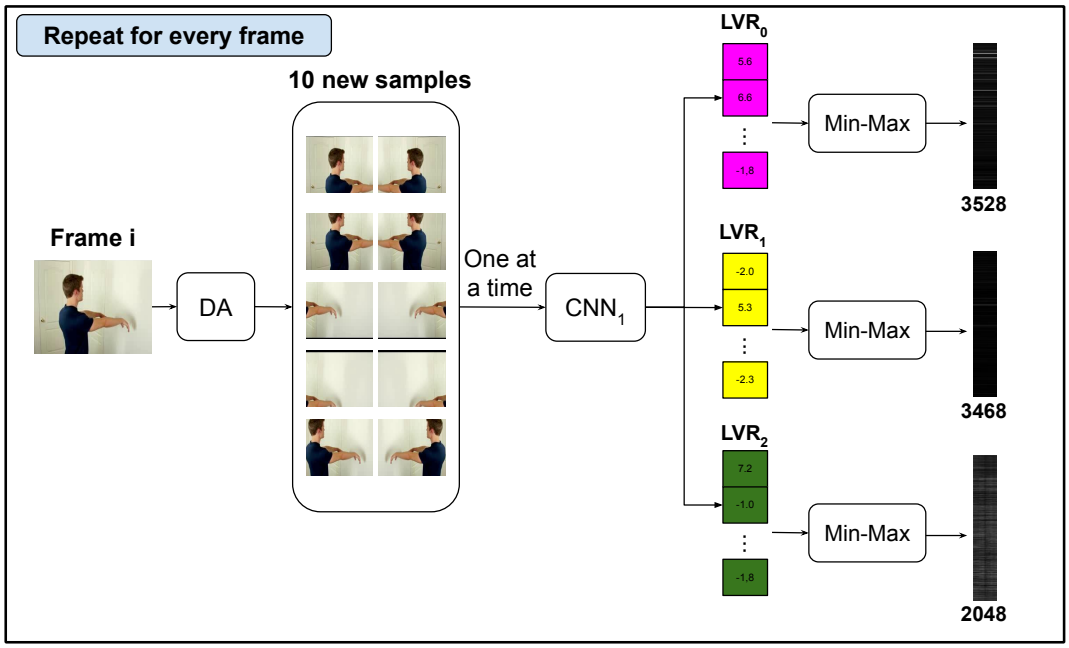
Each rhythm is f pixels high and t pixels wide, where f is the size of the feature vector and t is the number of video frames. These rhythms are individually used as input to the second CNN of the LVR stream, which is responsible for action classification. For matching the second CNN input dimension, we apply an external adaptive average pooling along the vertical axis and a resize method along the horizontal one. The adaptive average pooling is a type of pooling that generates outputs of fixed size, regardless of the input size. We apply distinct methods for the axes because the rhythm height is much greater than its width (about 10 times greater). The entire process of VR extraction is depicted in Figure 6.3.

The main purpose of the classification network is to find temporal patterns in the rhythm images. As the first one, it is initialized with ImageNet weights, but fine-tuned on the video dataset. Random horizontal flipping is applied in this training step, in addition to the 10 data augmentation techniques applied in the first CNN. The training and testing processes consider a single abstraction level for the inputs at a time, that is, LVR_0, LVR_1 and LVR_2 are trained/tested separately. For test, the same 10 techniques are used to augment the number of samples.

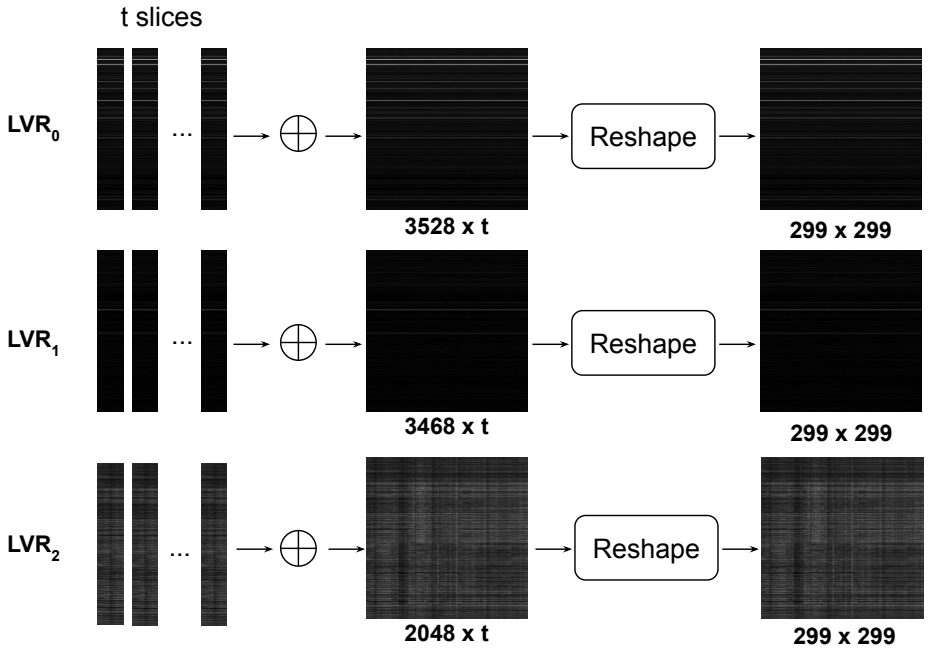
6.2 Results

To obtain the LVR slices, we apply a 2D average pooling with kernel size of 10 for LVR_0 and of 8 for both LVR_1 and LVR_2. Considering t the number of frames in a given video, LVR dimensions become $3528\times t$, $3468\times t$ and $2048\times t$, respectively (recall Table 6.1). These dimensions are reduced through an adaptive pooling operation along the vertical axis and resizing along the horizontal one, resulting in a 299×299 image that matches the input size of Inception V3.

Table 6.2 shows the stream results separately and includes different approaches using



(a) Slice Extraction



(b) Visual Rhythm Construction

Figure 6.3: LVR extraction. (a) We produce 10 new samples from each frame i using cropping and horizontal flipping techniques. The extractor CNN (CNN_1) receives each of these 10 samples as input at a time, generating three slices per sample. (b) The corresponding slices (same technique and depth) are combined to form a rhythm image. The dimensions of the rhythm are modified to match the Inception input size. To deal with the disparity between the rhythm height and width, we apply different techniques for each dimension. This process generates 30 different rhythms per video to be used in the second CNN.

visual rhythm images. We can see that the temporal stream still outperforms the others on both datasets. Among the rhythm approaches, LVR_1 presents the best accuracies, considerably exceeding the AVR approach, which suggests that deep features may be more representative than handcrafted ones.

Table 6.2: Individual Results. Cells on bold represents the overall highest accuracy rate.

Modality	UCF101	HMDB51
RGB*	86.61	51.77
Optical Flow	**86.95**	**59.91**
AVR	64.74	39.63
LVR_0	63.64	35.06
LVR_1	81.26	51.94
LVR_2	78.75	45.53

These results also indicate the superiority of the intermediate depth in providing good frame descriptions for action recognition. Since the Inception is originally trained for the object recognition problem, later layers may produce features more invariant to poses and positions. Hence, the classification CNN that works with LVR_2 images might face difficulties in capturing the temporal evolution and distinguishing the actions. LVR_0, on the other hand, may lack information about the scene structure. This last observation is reinforced by the fact that the handcrafted AVR outperforms LVR_0. Moreover, compared to the other depths, this image undergoes a significant reduction in size to feed the CNN.

In our second experiment (Table 6.3), we combine the visual rhythm approaches with the improved spatial (RGB*) and temporal (OF) streams. In the combination, we assign weight equal to 2 to the RGB*, 3 to the OF, and 1 to any of the rhythm approaches based on individual results. We can see that all the combinations outperform individual versions, suggesting that the streams complement each other in some levels.

Concerning the RGB* combinations, the AVR surpasses all the LVR approaches. A possible explanation is that the feature computed in the spatial network is already embedded into the LVR images, since the feature-extractor CNNs are very similar to the spatial stream. As such, they may present redundant information. In contrast, the OF stream benefits most from LVR information. The three-stream combinations reach the highest accuracies, except for the RGB* + OF + LVR_0 on HMDB51. In all cases, the contribution of the LVR_0 is lower, in line with individual results. The combination RGB* + OF + LVR_2 achieves the best result on UCF101, whereas RGB* + OF + AVR has the best performance on HMDB51. However, OF + LVR_1 and OF + LVR_2 present good results at a lower computational cost, since they are composed of two streams.

From Table 6.3, we notice that AVR and LVR seem to provide different contributions to the overall network. Therefore, in our third experiment (Table 6.4), we combine the visual rhythm approaches to assess their complementarity. For this experiment, we use an arithmetic average to fuse the outputs, that is, every rhythm stream receives 1 as weight. It is possible to observe that the presence of the AVR and LVR_1 streams tends to increase the accuracy rates. For instance, LVR_0 + LVR_1 increases from 82.49% and 51.85% to

87.59% and 60.09% with AVR + LVR_0 + LVR_1; AVR + LVR_2 increases from 85.54% and 56.54% to 89.34% and 62.16%, after the inclusion of the LVR_1. In fact, the combination AVR + LVR_1 presents sufficient competitive results compared to the top combinations in this table.

Table 6.3: Fusion of spatial, temporal and visual rhythm streams. Cells on bold represent the overall highest accuracy rate, whereas underlined cells consist of combinations that present satisfactory results at a lower computational cost.

Combination	UCF101	HMDB51
RGB* + AVR	90.74	61.31
RGB* + LVR_0	86.97	53.99
RGB* + LVR_1	89.29	57.30
RGB* + LVR_2	88.90	56.88
OF + AVR	89.23	65.45
OF + LVR_0	89.71	63.16
OF + LVR_1	<u>91.59</u>	<u>66.49</u>
OF + LVR_2	<u>92.15</u>	<u>65.56</u>
RGB* + OF + AVR	93.74	**69.98**
RGB* + OF + LVR_0	93.04	65.66
RGB* + OF + LVR_1	93.47	67.28
RGB* + OF + LVR_2	**94.00**	66.71

Table 6.4: Fusion of visual rhythm approaches using arithmetic average. The top-3 combinations are highlighted in bold. The underlined values correspond to the combination with a good trade-off between accuracy and computational cost.

Combination	UCF101	HMDB51
AVR + LVR_0	79.75	51.48
AVR + LVR_1	<u>86.52</u>	<u>59.78</u>
AVR + LVR_2	85.54	56.54
LVR_0 + LVR_1	82.49	51.85
LVR_0 + LVR_2	82.48	50.59
LVR_1 + LVR_2	85.47	55.93
AVR + LVR_0 + LVR_1	**87.59**	**60.09**
AVR + LVR_0 + LVR_2	87.55	59.46
AVR + LVR_1 + LVR_2	**89.34**	**62.16**
LVR_0 + LVR_1 + LVR_2	86.31	56.99
AVR + LVR_0 + LVR_1 + LVR_2	**89.47**	**62.68**

We select the top-3 combinations of the visual rhythm approaches from Table 6.4 for the following experiment, along with the lower-cost AVR + LVR_1. These combinations

are fused with the spatial and temporal streams. As in the second experiment, we assign the weights 2 and 3 to RGB* and OF streams, respectively, and 1 to each visual rhythm approach. The results are shown in Table 6.5.

Table 6.5: Fusion of the best visual rhythm combinations and the spatial and temporal stream.

Combination	UCF101	HMDB51
RGB* + AVR + LVR_1	91.87	64.12
RGB* + AVR + LVR_0 + LVR_1	91.47	64.12
RGB* + AVR + LVR_1 + LVR_2	92.20	64.64
RGB* + AVR + LVR_0 + LVR_1 + LVR_2	91.90	64.47
OF + AVR + LVR_1	92.31	70.02
OF + AVR + LVR_0 + LVR_1	92.75	70.20
OF + AVR + LVR_1 + LVR_2	93.59	70.85
OF + AVR + LVR_0 + LVR_1 + LVR_2	93.72	70.46
RGB* + OF + AVR + LVR_1	94.30	70.70
RGB* + OF + AVR + LVR_1 + LVR_2	94.37	70.33
RGB* + OF + AVR + LVR_0 + LVR_1	94.43	**70.96**
RGB* + OF + AVR + LVR_0 + LVR_1 + LVR_2	**94.49**	69.91

The behavior of the combinations is very similar to the ones shown in Table 6.3, that is, most combinations using only the OF outperform those using only RGB*, however, both together surpass others. The combinations using RGB* and OF present very similar accuracy rates on UCF101. From the top-2, for instance, the difference is only 0.06% (from 94.43% to 94.49%), at a cost of a new stream. Since the five-stream RGB* + OF + AVR + LVR_0 + LVR_1 presents the highest accuracy on HMDB51, it might be the best option. The four-stream RGB* + OF + AVR + LVR_1 also presents satisfactory results at a even lower computational cost. Therefore, we select both approaches to compare against state-of-the-art methods (Table 6.6).

In Table 6.6, we separate the methods according to the pre-training strategy. Some of them are not based on deep networks and so they do not have a pre-training step. We highlight the overall highest accuracies and the highest that do not pre-train on Kinetics dataset. The works pre-trained with both have an advantage over those trained only with ImageNet, since the Kinetics is one of the largest and most varied dataset for the action recognition problem, and generates more robust networks. For this reason, these methods achieve considerable higher results. Considering the first group, we achieve the 6th best accuracy on UCF101 and the 5th best on HMDB51, both with the five-stream LVR.

Table 6.6: Comparison of accuracy rates (%) for UCF101 and HMDB51 datasets. Cells on bold represents the overall highest accuracy rates, whereas underlined cells consist of the best results using only ImageNet to pre-train the network. The works are ordered by their publication year.

Method	Year	Pre-training Dataset	UCF101	HMDB51
iDT + FV [136, 137]	2013	—	85.9	57.2
iDT + HSV [90]	2016	—	87.9	61.1
Two-stream + SVM [113]	2014	ImageNet	88.0	59.4
Two-stream + LSTM [81]	2015	ImageNet + Sports-1M	88.6	—
TDD + iDT [140]	2015	ImageNet	91.5	65.9
KVMDF [159]	2016	ImageNet	93.1	63.3
Two-stream fusion + iDT [33]	2016	ImageNet	93.5	69.2
Two-stream TSN [142]	2016	ImageNet	94.0	68.5
Three-stream TSN [142]	2016	ImageNet	94.2	69.4
Recurrent hybrid network [149]	2017	ImageNet	93.2	71.8
L^2STM [121]	2017	ImageNet	93.6	66.2
Three-stream [138]	2017	ImageNet	94.1	70.4
STP [144]	2017	ImageNet	94.6	68.9
TLE [28]	2017	ImageNet	95.6	71.1
Two-stream LTC + iDT [134]	2018	Sports-1M	92.7	67.2
Gated TSN [158]	2018	ImageNet	94.5	—
Four-stream + iDT [5]	2018	ImageNet	96.0	<u>74.9</u>
Heterogeneous two-stream [20]	2019	ImageNet	94.4	67.2
Two-stream Choquet [107]	2020	ImageNet	92.9	65.9
TEA [66]	2020	ImageNet	<u>96.9</u>	73.3
Two-Stream I3D [15]	2017	ImageNet+Kinetics	97.9	80.2
SVMP + I3D [139]	2018	ImageNet+Kinetics	—	81.3
R(2+1)D-TwoStream [130]	2018	Kinetics	97.3	78.7
DTPP [157]	2018	ImageNet+Kinetics	98.0	**82.1**
I3D+PoTion [22]	2018	ImageNet+Kinetics	**98.2**	80.9
TAV [8]	2020	ImageNet+Kinetics	97.1	77.0
Our methods (Chapters 5 and 6)				
AVR (three-stream) [24]	2018	ImageNet	93.7	69.9
LVR (four-stream) [72]	2019	ImageNet	94.3	70.7
LVR (five-stream) [72]	2019	ImageNet	94.4	71.0
AVR-K (three-stream) [70]	2020	ImageNet	93.9	70.1

Chapter 7

Analysis and Fusion

In this chapter, we show an analysis of individual streams and their combinations. Based on these analyses and the experiments described in Chapter 5, we propose some modifications on the streams aiming at the reduction of overfitting. After that, we revisit the trainable fusions introduced in Section 5.4 to assess the impact of the new streams in these methods. It is worth recalling that the stacking training is divided into (1) CNN training and (2) meta-classifier training. Henceforth, we refer to (1) as stream-training and (2) as meta-training. Analogously, the testing steps are referred to as stream-test (individual test) and meta-test (combination test).

7.1 Overfitting

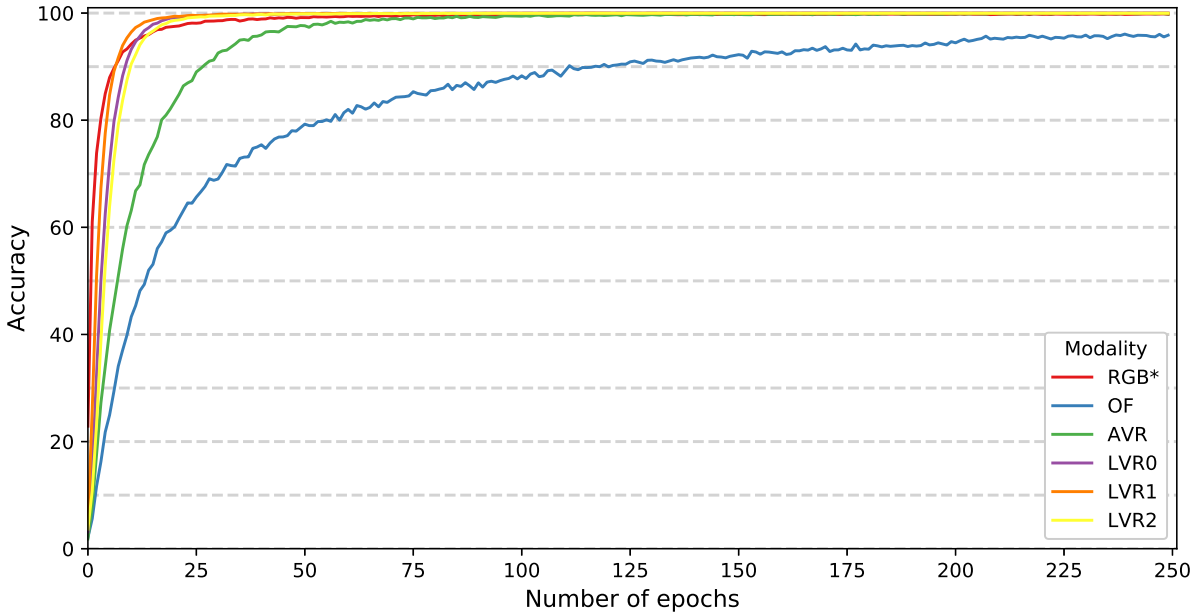
As mentioned at the end of Chapter 5, overfitting might be a problem for trainable fusions, since the vectors received in the meta-training do not accurately represent the meta-test inputs. In particular, if the streams achieve perfect training scores, many combinations of the vectors can reach high fusion scores, hampering parameter estimation.

Table 7.1: Stream-training and stream-test accuracies (%) at the 250th epoch for each dataset in Split 1.

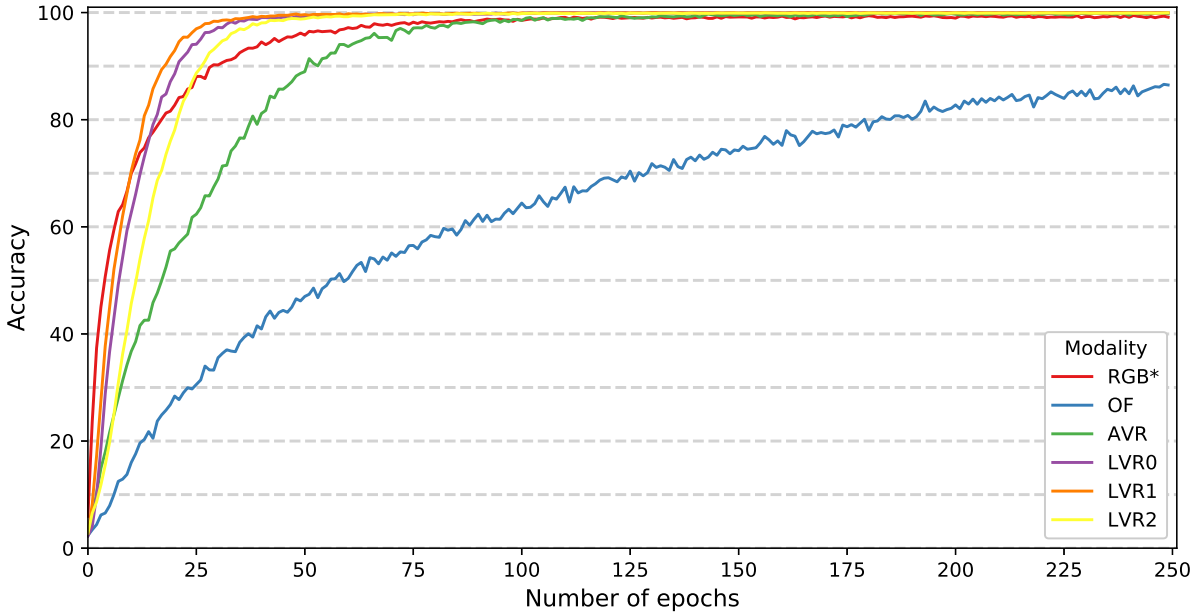
Stream	UCF101		HMDB51	
	Training	Test	Training	Test
RGB*	99.77	85.86	99.14	50.33
OF	95.87	85.36	86.48	58.69
AVR	99.84	63.49	99.76	38.04
LVR_0	99.99	65.61	99.96	38.76
LVR_1	99.98	79.43	99.95	50.52
LVR_2	99.97	79.20	99.95	45.16

To show the overfitting in individual streams, we train them for 250 epochs in Split 1 of UCF101 and HMDB51 datasets. Stream-training and stream-test accuracies at the last epoch are shown in Table 7.1. The large differences between both scores might indicate

overfitted models. On HMDB51, the overfitting is even worse, reaching a difference of 61.72 percentage points in AVR.



(a) UCF101



(b) HMDB51

Figure 7.1: Evolution of the stream-training accuracies throughout the epochs in the first split of (a) UCF101 and (b) HMDB51. All streams were trained for 250 epochs.

Figure 7.1 shows the evolution of the training accuracy rates throughout the epochs in Split 1 of both datasets for every modality. The behavior is very similar in the two datasets: RGB* and the three versions of LVR present the fastest growth, followed by the AVR and finally the OF. On UCF101, the four fastest modalities reach 90% of accuracy before the 12th epoch. On HMDB51, this growth is slightly slower, but the curves cross the 90% mark around the 25th epoch, only 10% of the total number of epochs. AVR also

crosses this mark around the epoch 25 on UCF101 and around the epoch 50 on HMDB51. These five modalities end with an almost perfect score on both datasets. Unlike the others, the OF takes more than 100 epochs to reach the 90% mark on UCF101 and it completes the training process under this value on HMDB51. As we will show later, this behavior is propitious for improvements by the early stopping strategy. Although the OF growth is slower, it reaches a high final training accuracy of 95.87% on UCF101.

In order to reduce overfitting, we train the streams with an early stopping strategy. We consider a strip length of 1 ($k = 1$), that is, the validation set is evaluated at every epoch. We use the validation loss to verify the improvement condition ($l - l^* > \delta$). Since the datasets do not include a validation set, we randomly separate 20% of the training set for validation. For the best of our knowledge, there is no specific strategy for selecting the parameters δ and patience s. However, they can be selected through the analysis of the accuracy/loss curve and by following some guidelines. Greater s is more suitable for models that present a noisy curve with frequent deteriorations and improvements in the training accuracy/loss. Models that present a fast training improvement may require a higher δ, which implies in a stricter improvement condition. We perform this analysis in Split 1 from the two datasets and replicate the selected values for the other splits. We test different values for delta ($\delta \in \{0.00, 0.05, 0.10, 0.15, 0.20\}$) and patience ($s \in \{1, 2, 3, 4, 5, 6, 7\}$).

In our analysis, to support the parameter decision, we consider (i) the epochs when the curves from Figure 7.1 cross the 90% marker and (ii) the stream-training accuracy obtained in the early stopping procedure. We seek a training accuracy that is below the perfect score, but not so smaller that could lead to an underfitting. It is worth highlighting that the training accuracy during the early stopping procedure is related to the subset with 80% of the original training samples. We refer to this subset as training subset.

Table 7.2: Experiments on UCF101 dataset in Split 1 with different δ values. For these experiments, we use a fixed patience $s = 7$. The column "e^*" shows the best epoch found for each δ. The column "acc" contains the accuracy of the training subset at e^*. Cells on bold represent the selected δ for each modality.

δ	RGB*		OF		AVR		LVR_0		LVR_1		LVR_2	
	e^*	acc	e^*	acc	e^*	acc	e^*	acc	e^*	acc	e^*	acc
0.00	35	98.55	**57**	**78.56**	43	95.45	26	99.49	28	99.60	41	99.60
0.05	13	94.37	35	67.68	**32**	**90.56**	**12**	**91.60**	12	96.45	19	96.65
0.10	**10**	**92.13**	34	67.57	22	80.21	15	95.73	**9**	**93.05**	**15**	**92.28**
0.15	10	92.13	26	60.59	21	79.88	10	86.12	12	96.45	13	88.60
0.20	5	81.53	20	53.45	17	72.63	14	94.98	9	93.05	12	85.43

Tables 7.2 and 7.3 show the experiments on UCF101 using Split 1. In the first table, we test different values of δ with a fixed patience $s = 7$. The table shows the training subset accuracy at the corresponding epoch. Since the OF presents the slowest growth, we select the lowest and most flexible value for δ, in such a way that any improvement is considered in the stop condition. Even with the less strict parameter setting, its training

Table 7.3: Experiments on UCF101 dataset in Split 1 with different s values. For each modality, we use the δ value highlighted in Table 7.2. Column "e^*" shows the best epoch found for each s. Column "acc" contains the accuracy of the training subset at e^*. Cells on bold represent the selected s for each modality.

s	RGB*		OF		AVR		LVR_0		LVR_1		LVR_2	
	e^*	acc	e^*	acc	e^*	acc	e^*	acc	e^*	acc	e^*	acc
1	4	76.93	17	48.72	15	68.17	7	70.90	3	44.44	8	65.56
2	6	85.13	35	67.68	19	74.72	12	91.60	6	81.35	12	85.43
3	6	85.13	35	67.68	25	83.73	12	91.60	9	93.05	15	92.28
4	10	92.13	35	67.68	25	83.73	12	91.60	9	93.05	15	92.28
5	**10**	**92.13**	**57**	**78.56**	25	83.73	**12**	**91.60**	**9**	**93.05**	**15**	**92.28**
6	10	92.13	57	78.56	25	83.73	12	91.60	9	93.05	15	92.28
7	10	92.13	57	78.56	**32**	**90.56**	12	91.60	9	93.05	15	92.28

accuracy is below 80%. For AVR and LVR_0, we use $\delta = 0.05$, because the corresponding accuracy rates are close to 90%. The accuracy for the AVR drops sharply for greater δ values, whereas for the LVR_0 the accuracy oscillates. RGB*, LVR_1 and LVR_2 still present higher accuracies with $\delta = 0.05$, so $\delta = 0.10$ seems more appropriate for them. Using these selected δ's, we tested different values for the patience s (Table 7.3). Unlike the δ, a change in s does not greatly affect the accuracy. Most of them remain the same for $s \geq 3$. In this way, we select $s = 5$ for all streams, except AVR that uses the default $s = 7$.

Table 7.4: Experiments on HMDB51 dataset in Split 1 with different δ values. For these experiments, we use a fixed patience $s = 7$. Column "e^*" shows the best epoch found for each δ. Column "acc" contains the accuracy of the training subset at e^*. Cells on bold represent the selected δ for each modality.

δ	RGB*		OF		AVR		LVR_0		LVR_1		LVR_2	
	e^*	acc	e^*	acc	e^*	acc	e^*	acc	e^*	acc	e^*	acc
0.00	**19**	**78.84**	**90**	**58.12**	**34**	**67.04**	**13**	**68.46**	**15**	**82.42**	**24**	**77.82**
0.05	13	69.76	52	45.08	29	60.88	13	68.46	15	82.42	20	66.56
0.10	13	69.76	35	33.00	29	60.88	11	61.13	12	73.22	18	61.67
0.15	9	61.00	32	33.32	24	54.60	12	64.89	12	73.22	19	64.18
0.20	9	61.00	26	27.60	16	41.08	9	52.41	14	79.5	21	70.34

The same experiments for the HMDB51 dataset are shown in Tables 7.4 and 7.5, again using Split 1. With the lowest δ (Table 7.4), all streams present an adequate accuracy, far from the perfect score. Therefore, we select $\delta = 0.00$ for all of them. Similar to UCF101, there is almost no variation in the accuracy as we change the patience (Table 7.5), especially for RGB* and LVR versions. We choose $s = 5$ for all streams, the exact value in which the last stabilization occurs (OF).

We use these selected parameters to train all the streams in the other splits with the early stopping strategy. The best epoch found in each experiment is used to retrain the

Table 7.5: Experiments on HMDB51 dataset in Split 1 with different s values. For each modality, we use the δ value highlighted in Table 7.4. Column "e^*" shows the best epoch found for each s. Column "acc" contains the accuracy of the training subset at e^*. Cells on bold represent the selected s for each modality.

s	RGB*		OF		AVR		LVR_0		LVR_1		LVR_2	
	e^*	acc	e^*	acc	e^*	acc	e^*	acc	e^*	acc	e^*	acc
1	11	64.94	32	33.32	21	49.16	9	52.41	6	41.86	16	55.26
2	19	78.84	48	42.00	29	60.88	13	68.46	15	82.42	21	70.34
3	19	78.84	55	43.52	29	60.88	13	68.46	15	82.42	24	77.82
4	19	78.84	73	54.48	29	60.88	13	68.46	15	82.42	24	77.82
5	**19**	**78.84**	**90**	**58.12**	**34**	**67.04**	**13**	**68.46**	**15**	**82.42**	**24**	**77.82**
6	19	78.84	90	58.12	34	67.04	13	68.46	15	82.42	24	77.82
7	19	78.84	90	58.12	34	67.04	13	68.46	15	82.42	24	77.82

streams with the entire training set (training subset and validation set). In Table 7.6, we show the selected parameters, the best epoch found and the stream-training and stream-test accuracy after the retraining. We can see that the difference between the training and testing accuracy rates were reduced, especially for the HMDB51 (as shown in Table 7.1). The early stopping strategy was particularly effective for the OF stream on both datasets. Oddly, in some cases, the testing accuracy was even greater than the training one, but the values are quite close.

Despite the good reduction in the accuracy differences, the early stopping strategy had a negative impact on the test accuracies. Comparing the averages from Table 7.6 with the results reported in Table 6.2, we can see that most of the current scores are approximately 1.5 percentage points below. This is a small reduction, considering that we also reduced the training epochs significantly, in addition to the accuracy differences. However, AVR and LVR_1 lost around 4.0 percentage points, which can affect the combinations that include them. The only stream that benefited from the early stopping method with respect to the test accuracy was the LVR_2 on HMDB51, presenting an increase of 1.27 points.

7.2 Analysis

With the models obtained after the early stopping retraining, we analyze the behavior of the new streams inspired by the analysis of the Kinetics paper [49]. Figure 7.2 shows the 5 easiest and 5 hardest classes for each modality on UCF101. To generate these lists, we considered the stream-test class recall[1] averaged over the splits. Only the "Billiards" class appears in all lists as an easy class. None of the classes are in the easiest list of one

[1]Some works use the so-called class accuracy metric [49, 132] for a class-level analysis. However, accuracy is a classifier-level metric. For the best of our knowledge, class accuracy matches the class recall definition, although no formal definition was provided. Therefore, here we use the term recall defined as $R(c) = \frac{tp_c}{n_c}$, where c is a given class, tp_c is the number of samples from c that were correctly classified and n_c is the total number of c samples.

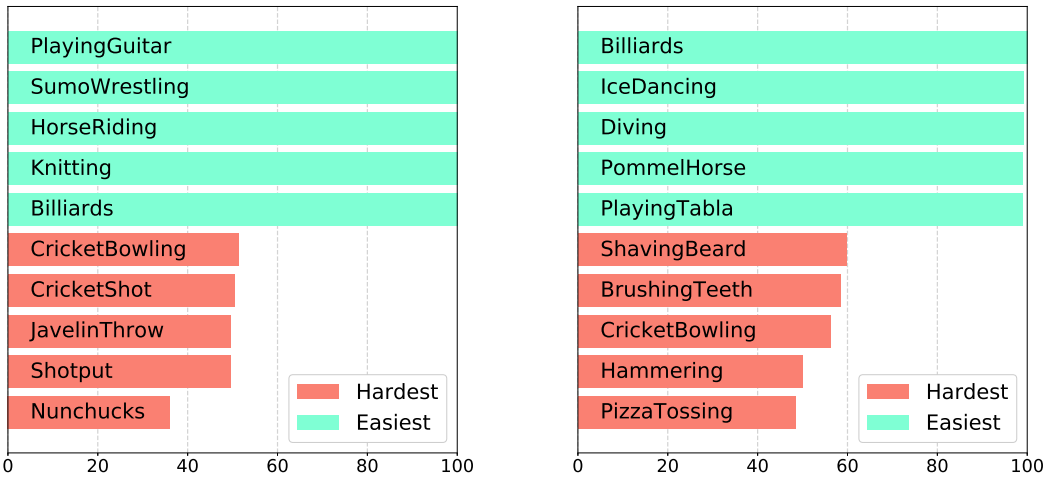
Table 7.6: Individual results after retraining on UCF101 (left) and HMDB51 (right). Column "(δ, s)" contains the parameters for the early stopping procedure selected after the analysis in Split 1. "e^*" refers to the best epoch found for each split. "Train" and "Test" refer to the training (training subset + validation set) and test accuracy after the retraining.

Stream	(δ, s)	Split	e^*	Train	Test
RGB*	$(0.10, 5)$	1	10	93.88	85.73
		2	8	91.16	84.68
		3	11	94.16	85.88
				Avg	**85.43**
OF	$(0.00, 5)$	1	57	79.78	83.51
		2	63	81.65	86.02
		3	87	86.47	85.85
				Avg	**85.13**
AVR	$(0.05, 7)$	1	32	92.81	60.43
		2	30	91.94	60.95
		3	23	86.69	58.93
				Avg	**60.10**
LVR_0	$(0.05, 5)$	1	12	95.27	59.53
		2	13	95.05	61.62
		3	17	98.36	63.93
				Avg	**61.69**
LVR_1	$(0.10, 5)$	1	9	94.46	77.58
		2	12	97.28	78.44
		3	8	91.98	77.33
				Avg	**77.78**
LVR_2	$(0.10, 5)$	1	15	95.26	76.90
		2	15	94.94	78.33
		3	15	95.26	76.46
				Avg	**77.23**

Stream	(δ, s)	Split	e^*	Train	Test
RGB*	$(0.00, 5)$	1	19	81.25	51.96
		2	18	80.38	49.22
		3	21	84.60	49.41
				Avg	**50.20**
OF	$(0.00, 5)$	1	90	61.40	56.60
		2	88	60.68	57.71
		3	67	53.12	58.37
				Avg	**57.56**
AVR	$(0.00, 5)$	1	34	75.76	35.03
		2	38	79.12	36.93
		3	32	69.28	36.54
				Avg	**36.17**
LVR_0	$(0.00, 5)$	1	13	72.71	37.25
		2	15	75.84	32.09
		3	18	83.72	31.83
				Avg	**33.72**
LVR_1	$(0.00, 5)$	1	15	81.64	50.33
		2	14	80.73	47.12
		3	12	74.60	46.41
				Avg	**47.95**
LVR_2	$(0.00, 5)$	1	24	83.24	45.42
		2	24	83.99	46.67
		3	23	81.37	48.30
				Avg	**46.80**

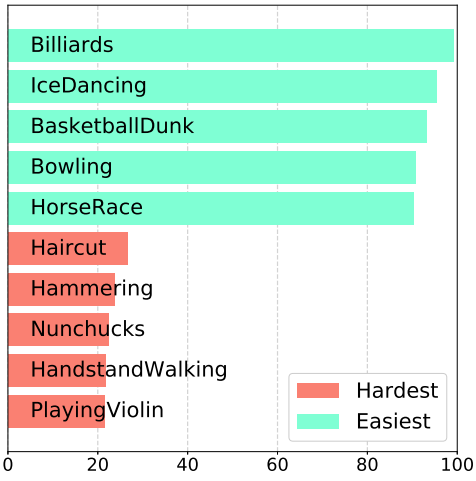
stream and the hardest list of another one.

Concerning the RGB* stream, its easiest classes generally involve specific objects such as "PlayingGuitar" and "HorseRiding". On the other hand, its hardest classes present common scenarios ("CricketShot" and "CricketBowling", "JavelinThrow" and "Shotput") that may have affected the appearance recognition. "CricketBowling" and "Nunchucks" reached lower scores in most streams. The majority of the easiest classes of the OF list present large and characteristic movements involving the entire body (for instance, "IceDancing"), in contrast with the hardest ones that contain more subtle motions (for instance, "BrushingTeeth"). In addition to "Billiards", other two classes are present in the easiest list of every VR-based modality: "BasketBallDunk" and "IceDancing". "IceDancing" is also common to the OF stream, perhaps because it requires temporal information

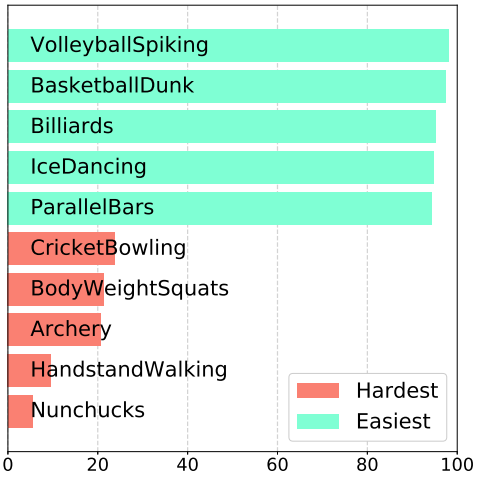


(a) RGB*

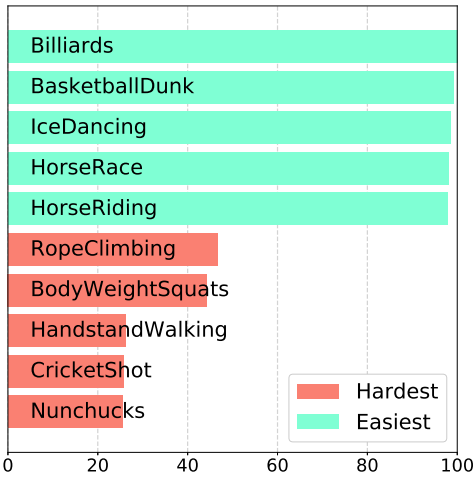
(b) OF



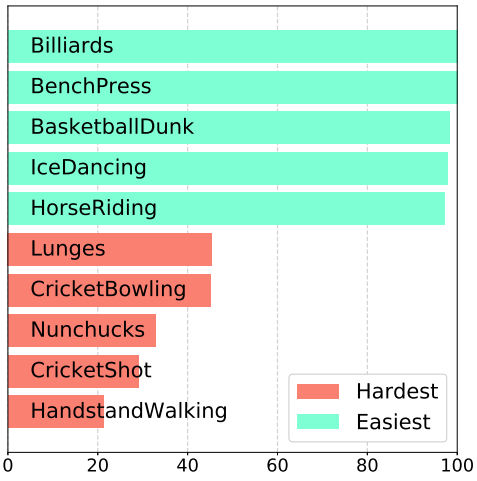
(c) AVR



(d) LVR_0



(e) LVR_1



(f) LVR_2

Figure 7.2: List of the 5 easiest (highest scores) and 5 hardest (smallest scores) classes for each modality on UCF101. We use the stream-test recall per class (horizontal axis) averaged over all splits.

to be distinguished. A significant number of videos from the "BasketBallDunk" (8) class have between 51 to 100 frames (Figure 4.4). It was expected that the reshape process, regarding the Inception input size, would distort the rhythm, causing a negative impact in the scores, but it was not the case. A possible explanation is that the CNN learned the distortion, since the majority of the "BasketBallDunk" clips fall in the same length interval ($[51, 100]$). which implies that many short videos were available during the training stage. However, it is not the only factor, because these streams do not achieve good results in other classes predominantly short such as "JumpingJack" (46), as shown ahead in Figure 7.6. Considering the hardest classes, "Nunchucks" and "HandStandWalking" are common to the four VR-based streams. The confusion analysis can provide new insights about the stream behavior.

Table 7.7 shows a list of the 5 ordered pairs of classes that reached the highest confusion rates on the UCF101 dataset. Each row contains the percentage of samples from one class that was predicted as the second one. Most of the pairs present at least one of the following situations: (i) similar scenarios (for instance, "CricketShot" and "CricketBowling"), (ii) similar poses (for instance, "HammerThrow" and "ThrowDiscus") and (iii) similar motion involving the same parts of the body (for instance, "PlayingGuitar" and "PlayingSitar"). On the other hand, a few confusions are not so easy to explain, for instance, "JavelinThrow" and "HighJump". Among the streams, the OF presents the smallest confusion rates, whereas the LVR_0 presents the highest ones.

The top 5 hardest and easiest classes on HMDB51 are shown in Figure 7.3. The "golf" class was effectively recognized by every stream, whereas all of them achieved low scores for the classes "swing_baseball", "throw" and "wave". Each hardest list contains exactly one class involving swords ("sword", "sword_exercise" and "draw_sword"). Table 7.8, which contains the top 5 confusion rates for HMDB51, shows that the pair "sword_exercise" and "draw_sword" achieved high rates for every stream. This might indicate some ambiguity in these two classes. There is a significant overlap between the hardest/easiest list from the RGB*, LVR_1 and LVR_2 streams. Their lists of the most confused pairs are also quite similar. Therefore, a complementarity analysis is required to assess whether these three streams can be combined or are redundant.

Tables 7.9 and 7.10 show the pairwise complementarity and kappa-statistic (agreement rate) on UCF101 and HMDB51, respectively. The complementarity rates indicate a target accuracy for the pair after the insertion of the second stream. For instance, for the spatial stream on UCF101, we have an average accuracy of 85.43%, which means that 14.57% of the samples were misclassified. Since the average complementarity rate of the pair RGB* + OF is $Comp(RGB*, OF) = 0.631$, the potential accuracy of the combination is 94.62% ($= 85.43 + 0.631 \cdot 14.57$). Two factors can influence the combination accuracy. First, the ability of the fusion method to select the best output, which can negatively affect the final accuracy. Second, since the fusion method combines score vectors and not labels, it can find the correct prediction from incorrect ones, positively affecting the accuracy. For instance, suppose a 3-class problem, two classifiers c_i and c_j and an arithmetic average fusion. Consider that c_i outputs the score vector $[0.30, 0.00, 0.20]$ for a given sample x_i, that is, it votes on class 0 for x_i. The classifier c_j outputs the vector $[0.00, 0.30, 0.20]$ for the same sample, that corresponds to the class 1. Based on both vectors, the fusion

Table 7.7: Top-5 class confusion rates per stream on UCF101. "Confusion (%)" column represents the proportion of samples from "Ground Truth" class that were classified as "Prediction" class. The percentage rates are given by the average of the split rates.

Stream	Ground Truth	Prediction	Confusion (%)
RGB*	CricketShot	CricketBowling	37.54
	CricketBowling	CricketShot	31.48
	JavelinThrow	HighJump	25.77
	HammerThrow	ThrowDiscus	21.22
	ThrowDiscus	HammerThrow	21.07
OF	PlayingGuitar	PlayingSitar	19.54
	BrushingTeeth	ShavingBeard	18.90
	BalanceBeam	FloorGymnastics	18.35
	SoccerPenalty	FieldHockeyPenalty	17.74
	ShavingBeard	Haircut	15.63
AVR	BreastStroke	FrontCrawl	32.64
	StillRings	ParallelBars	32.49
	ApplyLipstick	ApplyEyeMakeup	22.83
	FrontCrawl	BreastStroke	21.31
	MilitaryParade	BandMarching	21.27
LVR_0	ApplyLipstick	ApplyEyeMakeup	52.21
	CricketBowling	CricketShot	45.19
	Nunchucks	JugglingBalls	43.74
	HighJump	JavelinThrow	37.69
	ApplyEyeMakeup	ApplyLipstick	29.80
LVR_1	CricketShot	CricketBowling	40.94
	CricketBowling	CricketShot	36.80
	ShavingBeard	BrushingTeeth	27.02
	BrushingTeeth	ShavingBeard	24.78
	MilitaryParade	BandMarching	17.64
LVR_2	CricketShot	CricketBowling	34.82
	BrushingTeeth	ShavingBeard	25.33
	Nunchucks	JugglingBalls	19.44
	CricketBowling	CricketShot	18.30
	BreastStroke	FrontCrawl	17.36

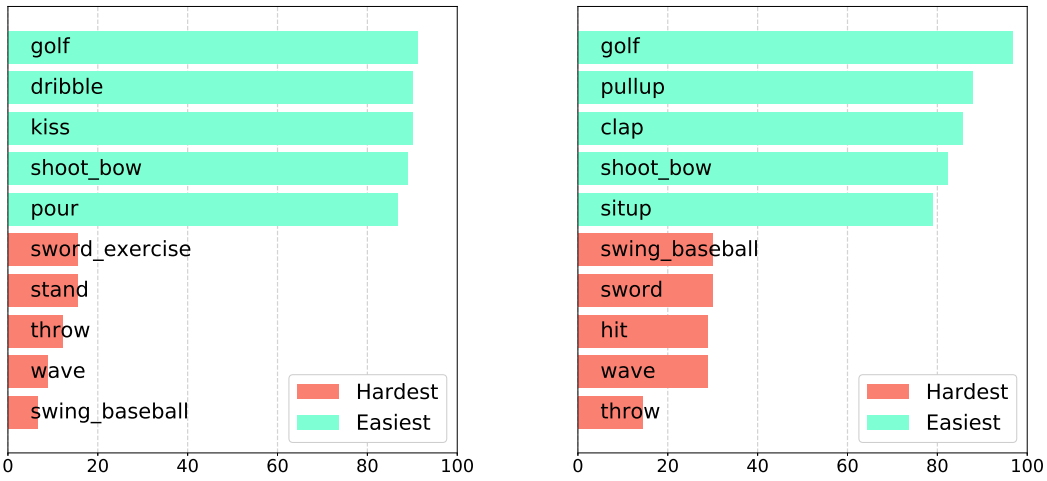
generates the final vector $[0.15, 0.15, 0.2]$, corresponding to the class 2, different from the two predictions. If one of the classifiers was correct (that is, the true label is either 0 or 1), the fusion method failed to select it (factor 1). If both were incorrect (that is, the true label is 2), the fusion was able to produce a correct prediction, despite the misleading inputs. Therefore, the complementarity measure is used for analytical purposes only.

Table 7.9 shows that all streams present more than 0.272 of complementarity and up to 0.411 of agreement rates on UCF101. It is worth mentioning that if c_j complements

Table 7.8: Top-5 class confusion rates per stream on HMDB51. "Confusion (%)" column represents the proportion of samples from "Ground Truth" class that were classified as "Prediction" class. The percentage rates are given by the average of the split rates.

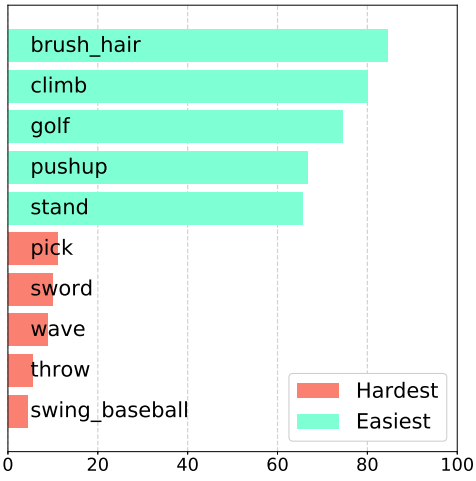
Stream	Ground Truth	Prediction	Confusion (%)
RGB*	swing_baseball	throw	50.00
	stand	sit	40.00
	sword_exercise	draw_sword	34.44
	cartwheel	flic_flac	31.11
	shoot_ball	dribble	28.89
OF	sword_exercise	draw_sword	25.56
	drink	eat	24.44
	eat	drink	23.33
	smoke	eat	18.89
	sword	fencing	17.78
AVR	drink	stand	31.11
	draw_sword	sword_exercise	28.89
	cartwheel	flic_flac	25.56
	ride_bike	ride_horse	25.56
	pullup	pushup	20.00
LVR_0	shoot_bow	pick	36.67
	handstand	cartwheel	32.22
	swing_baseball	throw	30.00
	kick	fall_floor	26.67
	sword_exercise	draw_sword	24.44
LVR_1	swing_baseball	throw	32.22
	sword_exercise	draw_sword	32.22
	stand	sit	31.11
	handstand	cartwheel	30.00
	sit	stand	30.00
LVR_2	swing_baseball	throw	52.22
	drink	eat	33.33
	sword_exercise	draw_sword	30.00
	stand	sit	27.78
	flic_flac	cartwheel	26.67

c_i much more than the opposite, i.e. $Comp(c_i, c_j) \gg Comp(c_j, c_i)$, the stream c_i is not being very useful for the pair. For this reason, we use the harmonic mean introduced by Equation 2.12. We can see that the pair RGB* and OF presents a balanced complementarity with harmonic mean $HM(\text{RGB*}, \text{OF}) = 0.635$. The LVR versions present higher complementarity rates combined with the OF stream rather than in the RGB* combinations, reaching $HM(\text{OF}, \text{LVR}_2) = 0.611$, whereas the AVR better complements the RGB* ($HM(\text{RGB*}, \text{AVR}) = 0.506$). The pair RGB* + AVR also presents the smallest

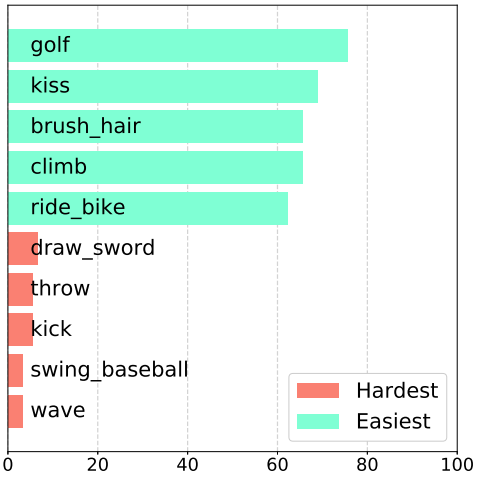


(a) RGB*

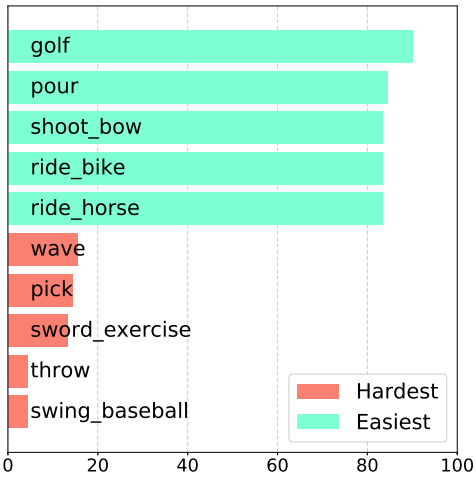
(b) OF



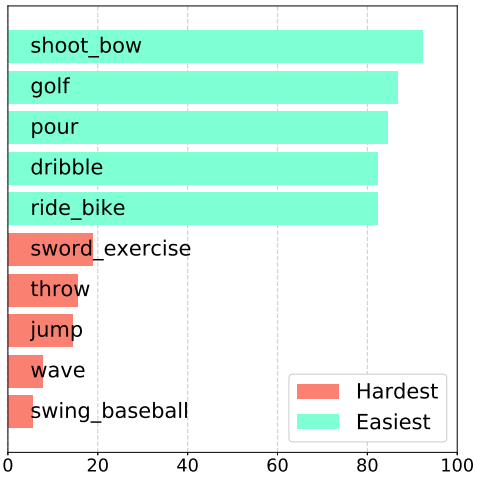
(c) AVR



(d) LVR_0



(e) LVR_1



(f) LVR_2

Figure 7.3: List of the 5 easiest (highest scores) and 5 hardest (smallest scores) classes for each modality on HMDB51. We use the stream-test recall per class (horizontal axis) averaged over all splits.

Table 7.9: Pairwise complementarity $Comp(c_i, c_j)$ (left) and kappa-statistic $\kappa_p(c_i, c_j)$ (right) on UCF101 dataset averaged over the splits. The stream in the row corresponds to c_i and c_j is the stream in the column. Recall that the complementarity is noncommutative, in contrast to the kappa-statistic.

Complementarity						
	RGB*	OF	AVR	LVR_0	LVR_1	LVR_2
RGB*	0.000	0.631	0.376	0.272	0.355	0.365
OF	0.640	0.000	0.299	0.369	0.525	0.541
AVR	0.772	0.738	0.000	0.463	0.663	0.656
LVR_0	0.722	0.755	0.441	0.000	0.598	0.609
LVR_1	0.579	0.683	0.395	0.308	0.000	0.450
LVR_2	0.594	0.701	0.398	0.343	0.463	0.000

Kappa-statistic						
	RGB*	OF	AVR	LVR_0	LVR_1	LVR_2
RGB*	1.000	0.254	0.153	0.242	0.405	0.385
OF	0.254	1.000	0.210	0.176	0.246	0.222
AVR	0.153	0.210	1.000	0.258	0.206	0.208
LVR_0	0.242	0.176	0.258	1.000	0.316	0.286
LVR_1	0.405	0.246	0.206	0.316	1.000	0.411
LVR_2	0.385	0.222	0.208	0.286	0.411	1.000

agreement rate of 0.153. The highest agreements were achieved by the pairs involving the RGB*, LVR_1 and LVR_2, all of them reaching about 0.4 of agreement. Despite this, they were able to complement each other with harmonic means of approximately 0.45. Although the agreement rates of the LVR_0 were small, it has little contribution to the other streams. Even the AVR, which reached the lowest individual accuracy rates, achieves better scores with the diversity metrics.

On HMDB51 (Table 7.10), all pairs achieved at least 0.184 of complementarity and at most 0.442 of agreement rates. In contrast to UCF101, the OF stream complements the RGB* more than the other way around ($HM(\text{RGB*}, \text{OF}) = 0.358$). The RGB* contribution to OF was similar to LVR_1 and LVR_2. The LVR_1 complementarity rates were higher than the other VR-based to both the RGB* and OF streams. However, the pair RGB* + LVR_1 presents a high agreement. As on UCF101, the pairs containing the RGB*, LVR_1 and LVR_2 present good scores, despite their similar behavior on the easiest/hardest classes analysis. In conclusion, we can see that every pair presents a promising contribution and a lower agreement rate on both datasets. Even the combinations between VR-based streams achieved good scores, although the LVR_0 shows inferior results.

Table 7.10: Pairwise complementarity $Comp(c_i, c_j)$ (left) and kappa-statistic $\kappa_p(c_i, c_j)$ (right) on HMDB51 dataset averaged over the splits. The stream in the row corresponds to c_i and c_j is the stream in the column. Recall that the complementarity is noncommutative, in contrast to the kappa-statistic.

Complementarity						
	RGB*	OF	AVR	LVR_0	LVR_1	LVR_2
RGB*	0.000	0.416	0.230	0.184	0.268	0.246
OF	0.314	0.000	0.203	0.205	0.312	0.308
AVR	0.399	0.470	0.000	0.239	0.384	0.366
LVR_0	0.387	0.491	0.267	0.000	0.361	0.369
LVR_1	0.300	0.439	0.244	0.186	0.000	0.272
LVR_2	0.294	0.448	0.239	0.214	0.288	0.000

Kappa-statistic						
	RGB*	OF	AVR	LVR_0	LVR_1	LVR_2
RGB*	1.000	0.319	0.263	0.307	0.422	0.442
OF	0.319	1.000	0.259	0.214	0.282	0.268
AVR	0.263	0.259	1.000	0.275	0.247	0.265
LVR_0	0.307	0.214	0.275	1.000	0.319	0.267
LVR_1	0.422	0.282	0.247	0.319	1.000	0.408
LVR_2	0.442	0.268	0.265	0.267	0.408	1.000

The interrater agreement is shown in Figure 7.4. The chart presents the interrater score for every possible r-combination using the 6 streams, except the pairs. Refer to Table B.3 in Appendix B for the index-combination mapping. We can see that the dataset curves in the figure present similar behavior, that is, their increases and decreases are aligned over

most of the interval. However, the UCF101 agreement rates are lower. On both datasets, the combinations that include any two streams from the group RGB*, LVR_1 and LVR_2 present higher agreement rates. They cause local maxima on the curves, for instance, the combinations RGB* + LVR_1 + LVR_2 (25) and OF + LVR_1 + LVR_2 (31). Combinations including the pair AVR and LVR_0, on the other hand, cause local minima, such as RGB* + AVR + LVR_0 (20) and RGB* + OF + AVR + LVR_0 (36). The best combination in terms of agreement rate was RGB* + OF + AVR on UCF101 (16) and OF + AVR + LVR_0 (26) on HMDB51.

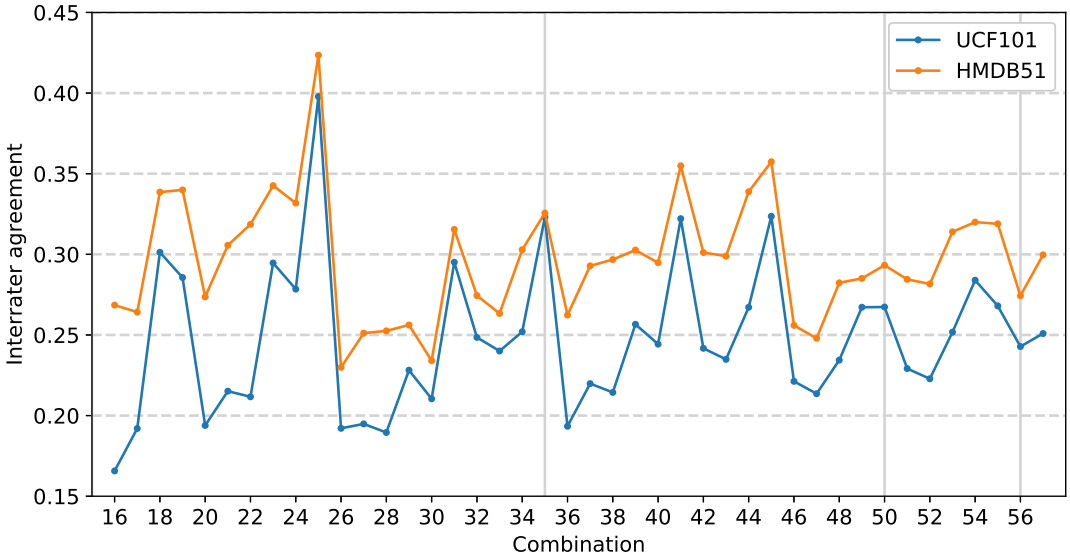


Figure 7.4: Interrater agreement for every possible r-combination of the 6 streams with $r \in \{3, 4, 5, 6\}$ on UCF101 and HMDB51. Vertical lines indicate the end of each combination group with regards to r.

Figures 7.5, 7.6 and 7.7 show the class recall per stream on UCF101 and HMDB51 averaged over the splits. The classes were ordered according to the maximum score obtained among the streams. On UCF101 (Figures 7.5 and 7.6), we can see that the RGB* and OF achieved the highest scores in most classes. However, in many classes, the LVR_1 and LVR_2 reached close results. The most interesting cases are those in which the first stream achieved results considerably higher than the second one. For instance, in the "Typing" class, the RGB* has a clear lead. This suggests that the aspect captured in that stream was crucial for the class. Although the VR-based streams reach the first place in some classes, none of them were by a wide margin. This suggests, in turn, that their complementarity might be in an intra-class level.

Unlike UCF101, superior scores from VR-based streams are more frequent on HMDB51. In some of them, there is a significant difference between the first and second recall rates, such as in "punch" and "stand" classes. These charts also show that the RGB*, LVR_1 and LVR_2 have different behavior on the datasets, although some of their best and worst cases have coincided, according to the easiest/hardest analysis. The RGB* outperformed the OF more frequently on UCF101 than on HMDB51, which may indicate that the UCF101 classes are better recognized by the appearance, whereas the HMDB51 requires temporal information.

Figure 7.5: Class recall per stream on UCF101 averaged over all splits. The scores were ordered according to the maximum recall in each class. UCF101 was divided into two groups for illustration purposes, here we show the first 51 classes.

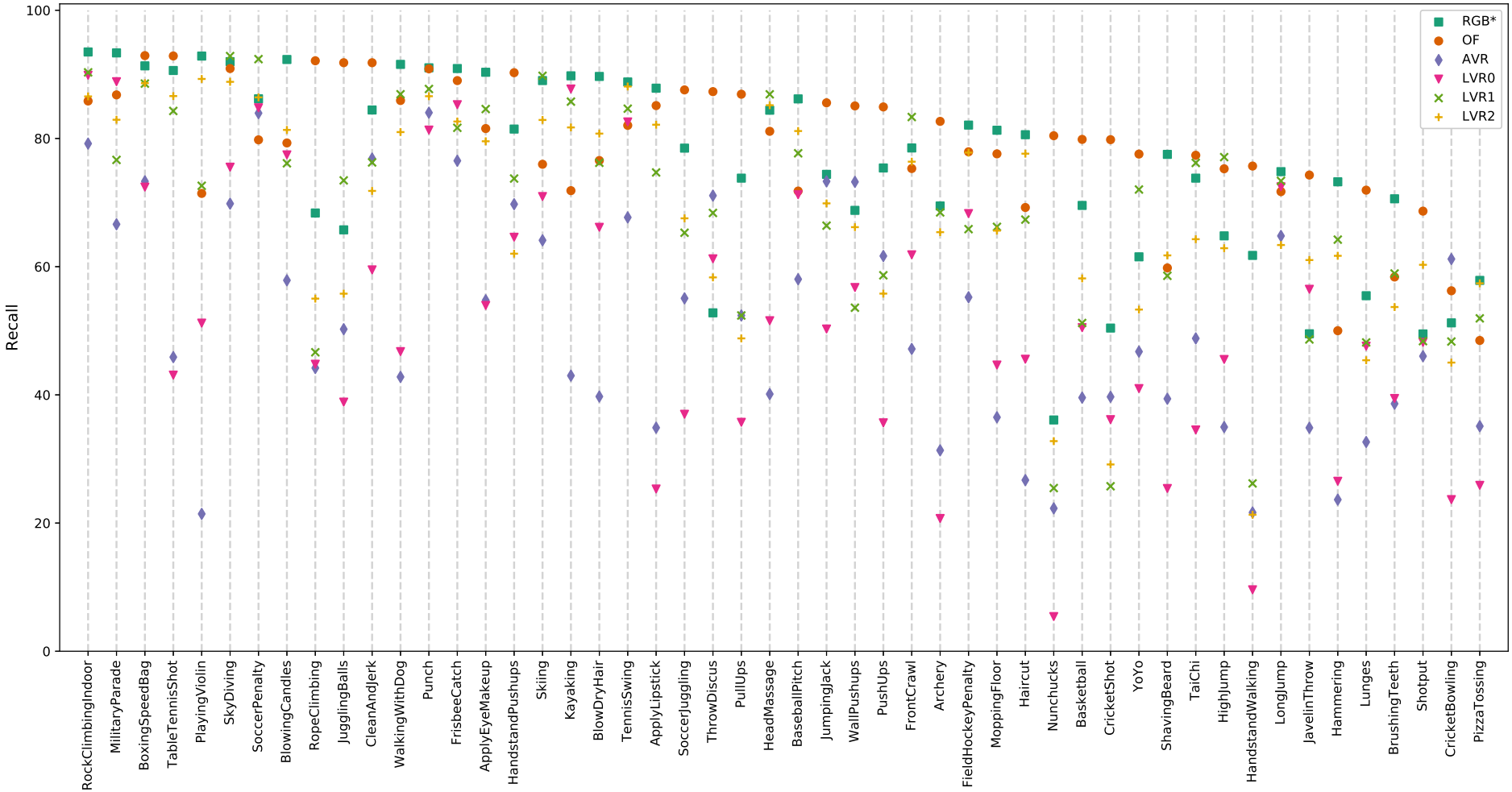


Figure 7.6: Class recall per stream on UCF101 averaged over all splits. The scores were ordered according to the maximum recall in each class. UCF101 was divided into two groups for illustration purposes, here we show the last 50 classes.

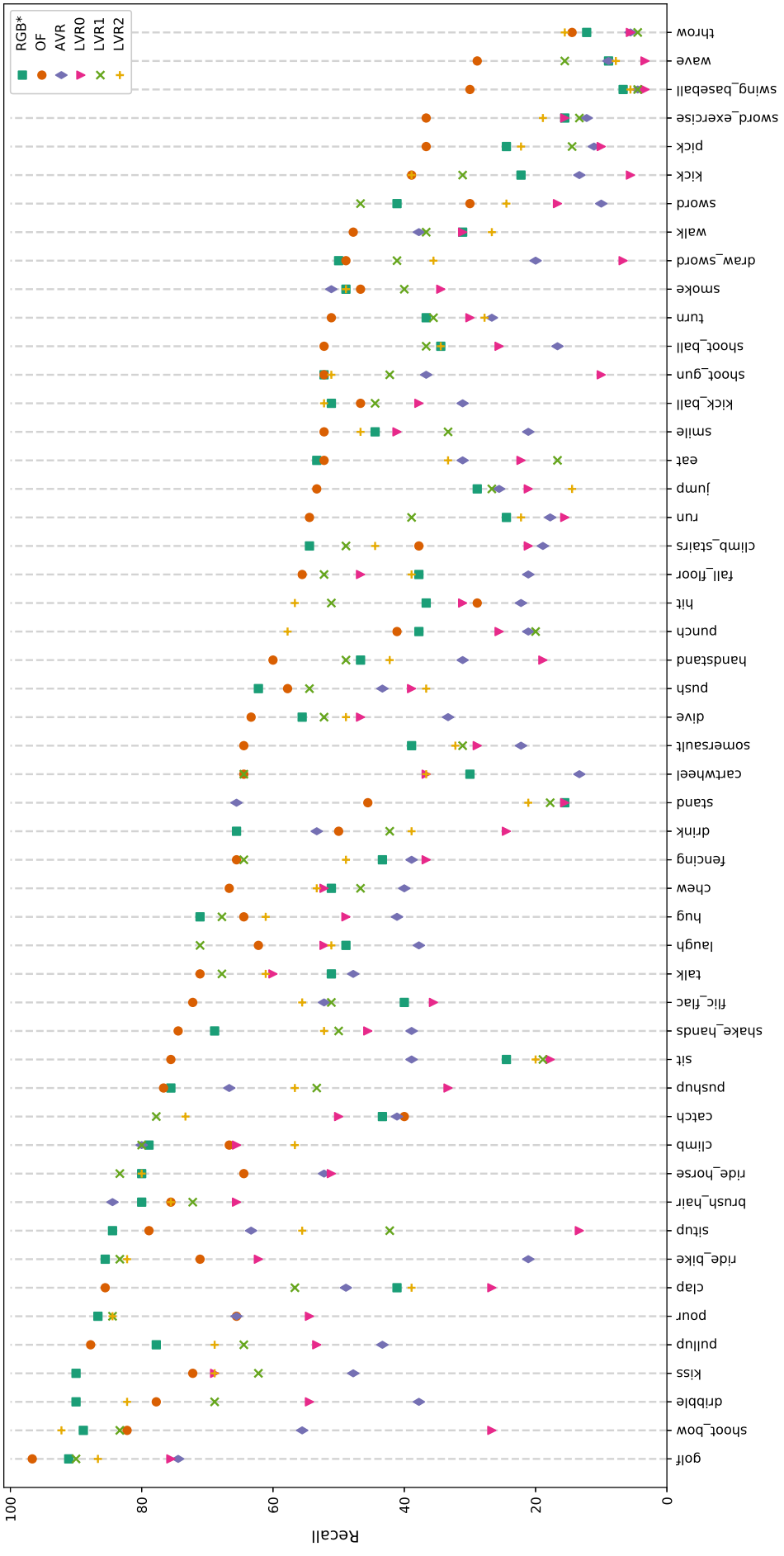


Figure 7.7: Class recall per stream on HMDB51 averaged over all splits. The scores were ordered according to the maximum recall in each class.

7.3 Fusion

We explore three fusion methods for the combination of the score vectors: (i) weighted average, (ii) fully connected layer and (iii) SVM classifier. Their parameters are estimated using the training set and, for this reason, are better adapted to different datasets and stream-combinations. The fusion methods are further detailed as follows. For the methods, consider r the number of parallel streams (after the early stopping retraining) in the combination and m the number of classes.

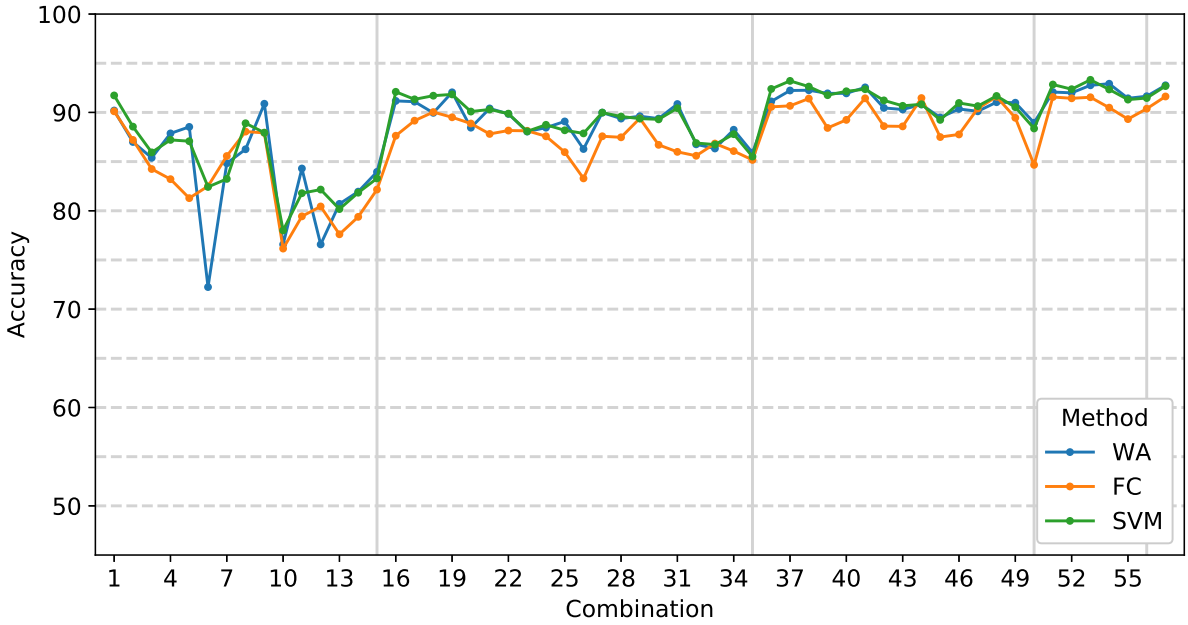
Weighted Average (WA): Unlike the weighted average described in Chapters 5 and 6, for this one we explore a grid search strategy to select the weights. For each weight, we tested every value from 1 to 10 with step 1. It is worth mentioning that the arithmetic average is also covered with this parameter interval. For the weight selection, we consider only the training samples. The weight set that achieves the highest accuracy is used in the meta-test stage.

Fully Connected Layer (FC): The same FC layer introduced in Section 5.4 is tested here, the difference is that it includes more streams and they are trained with the early stopping strategy. The layer is still composed of m neurons, but the size of the input is $r \cdot m$, which corresponds to the concatenation of the streams score vectors. It has a total of $r \cdot m^2$ trainable weights. The fusion network is also trained using an early stopping protocol, with $\delta = 0$, $s = 7$, and the same training-validation division from the stream-training. The score vectors are standardized, that is, they have zero mean and are scaled by the standard deviation. Both the mean and the standard deviation are computed from the training subset and replicated for the other sets.

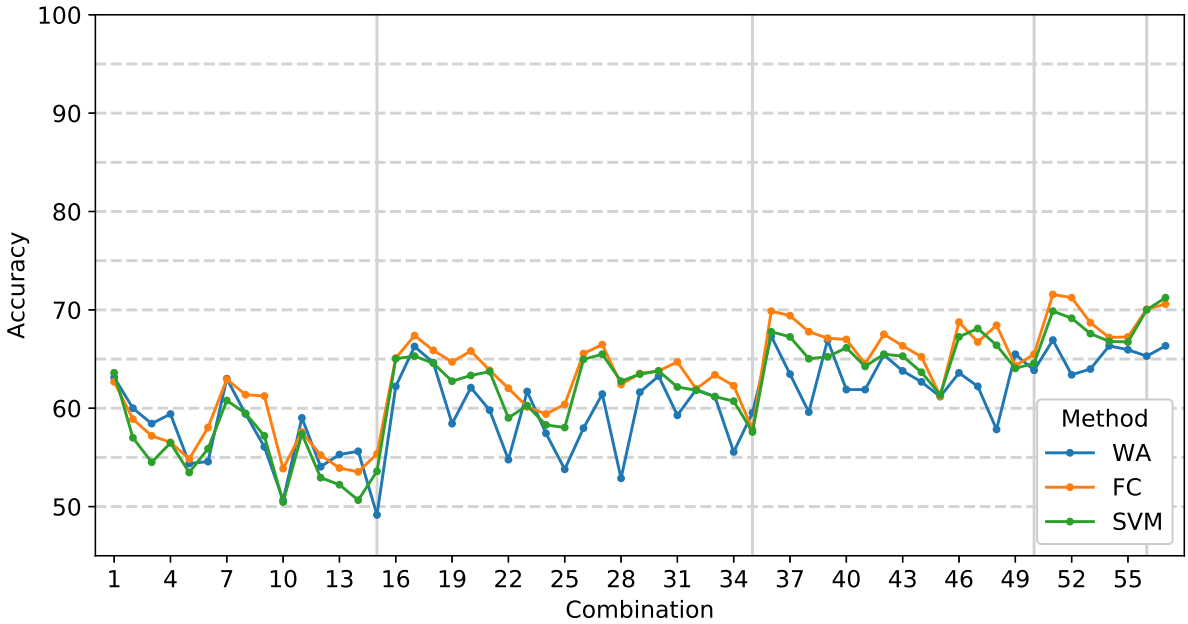
Support Vector Machine (SVM): We train an SVM classifier for fusion using the C-SVC approach. As in the weighted average, we perform a grid search for selecting the SVM parameters, that is, the regularization factor C, the kernel type with γ and the multi-class strategy. For C and γ, we consider the set $\{10^{-4}, 10^{-3}, \cdots, 10^{4}\}$. We test linear and RBF kernels, and the γ parameter is only used for the RBF kernel. The multi-class strategy can be either OvO or OvR. The grid search is performed over the training set with a 3-fold cross-validation, and the accuracy is used for the selection. The score vectors are also standardized using the mean and standard deviation from the training set.

Figure 7.8 shows a comparison of the three methods in the first split of UCF101 and HMDB51. For comparison, we use the accuracy of every r-combination of streams $r \in \{2, 3, 4, 5, 6\}$ (Table B.3). The results were similar, but SVM outperforms the others in most combinations on UCF101 and the FC on HMDB51. The weighted average presents a greater variation on accuracy across the combinations and its results are inferior to the other two in many cases. We select the SVM fusion, because in addition to being the best method on UCF101, it achieves comparable results on HMDB51.

Tables 7.11 and 7.12 show the 5 combinations that achieve the highest accuracy rates on each dataset. The best combination on UCF101 is the five-stream RGB* + OF + AVR



(a) UCF101



(b) HMDB51

Figure 7.8: Fusion methods on UCF101 and HMDB51 using every possible r-combination of the 6 streams, where $r \in \{2, 3, 4, 5, 6\}$. Vertical lines indicate the end of each combination group with regards to r.

+ LVR_1 + LVR_2 with 93.39% of accuracy. On HMDB51, the six-stream version outperforms the others, achieving 69.98%. Some four-stream combinations achieve comparable performance on UCF101, which involve a lower computational cost. For the HMDB51, the top-5 combinations present at least five streams. Considering other options with lower computational cost, we have the RGB* + OF + AVR and RGB* + OF on UCF101 that reach the 8th and 16th place with 92.66% and 91.80% of accuracy, respectively, and the

Table 7.11: Meta-test accuracy rates of the top-5 best combinations on UCF101 using the SVM fusion. Column "(C,ms,k[,γ])" contains the SVM parameters found with the grid search strategy, where "ms" and "k" refer to the multi-class strategy and kernel type, respectively. For RBF kernel, we also show the γ value.

Combination	Split	(C,ms,k[,γ])	Test
RGB* + OF + AVR + LVR_1 (37)	1	(10^{-3}, ovr, linear)	93.21
	2	(10^{-3}, ovr, linear)	93.41
	3	(10^{-3}, ovr, linear)	92.48
		Avg	**93.03**
RGB* + OF + AVR + LVR_0 + LVR_1 + LVR_2 (57)	1	(10^{-4}, ovr, linear)	92.68
	2	(10^{-3}, ovr, linear)	93.44
	3	(10^{-3}, ovr, linear)	93.15
		Avg	**93.09**
RGB* + OF + AVR + LVR_0 + LVR_2 (52)	1	(10^{-4}, ovr, linear)	92.36
	2	(10^{-3}, ovr, linear)	93.65
	3	(10^{-3}, ovr, linear)	93.7
		Avg	**93.24**
RGB* + OF + AVR + LVR_2 (38)	1	(10^{-3}, ovr, linear)	92.62
	2	(10^{-3}, ovr, linear)	93.36
	3	(10^{-2}, ovr, linear)	93.86
		Avg	**93.28**
RGB* + OF + AVR + LVR_1 + LVR_2 (53)	1	(10^{-3}, ovr, linear)	93.31
	2	(10^{-3}, ovr, linear)	93.57
	3	(10^{-3}, ovr, linear)	93.29
		Avg	**93.39**

OF + AVR + LVR_1 and RGB* + OF on HMDB51 that achieve 65.90% (11th) and 61.26% (34th). The OF stream presents the highest computational cost, since it computes 250 optical flow fields using the dense TV-L^1. The best combination that excludes the OF is the five-stream RGB* + AVR + LVR_0 + LVR_1 + LVR_2 that achieves 91.14% on UCF101 and 65.80% on HMDB51.

Although the differences between the stream-training and stream-test accuracy rates were reduced with the early stopping strategy, the final results with the approach presented in this chapter are slightly lower than those shown in Table 6.6, losing approximately 1 percentage point on both datasets. This might be caused by a decrease in the stream-test accuracy rates after the early stopping. However, the reductions in the meta-test scores were smaller than in the stream-test ones, especially if considering the 4-point reduction in the AVR and LVR_1. Moreover, the current approach has some advantages compared to the previous one, such as the reduced stream-training cost thanks to the early stopping protocol and the fusion adaptability to other datasets and streams. In particular, the reduction in the stream-training cost facilitates the use of larger datasets, such as the Kinetics. Both the early stopping and the adaptable fusion can also be applied

Table 7.12: Meta-test accuracy rates of the top-5 best combinations on HMDB51 using the SVM fusion. Column "(C,ms,k[,γ])" contains the SVM parameters found with the grid search strategy, where "ms" and "k" refer to the multi-class strategy and kernel type, respectively. For RBF kernel, we also show the γ value.

Combination	Split	(C,ms,k[,γ])	Test
RGB* + OF + AVR + LVR_1 + LVR_2 (53)	1	(10^{-1}, ovr, linear)	67.58
	2	(10^{2}, ovr, rbf, 10^{-4})	68.10
	3	(10, ovr, rbf, 10^{-3})	69.41
		Avg	**68.37**
RGB* + OF + AVR + LVR_0 + LVR_1 (51)	1	(10, ovr, rbf, 10^{-3})	69.87
	2	(10^{-2}, ovr, linear)	68.95
	3	(10, ovr, rbf, 10^{-3})	68.17
		Avg	**69.00**
RGB* + OF + AVR + LVR_0 + LVR_2 (52)	1	(10^{-1}, ovr, linear)	69.15
	2	(10^{2}, ovr, rbf, 10^{-3})	68.95
	3	(10^{-2}, ovr, linear)	69.15
		Avg	**69.08**
OF + AVR + LVR_0 + LVR_1 + LVR_2 (56)	1	(10, ovr, rbf, 10^{-3})	70.00
	2	(10^{-2}, ovr, linear)	68.95
	3	(10^{-2}, ovr, linear)	68.30
		Avg	**69.08**
RGB* + OF + AVR + LVR_0 + LVR_1 + LVR_2 (57)	1	(10, ovr, rbf, 10^{-3})	71.24
	2	(10^{-1}, ovr, linear)	69.02
	3	(10^{-2}, ovr, linear)	69.67
		Avg	**69.98**

to other multi-stream architectures, such as two-stream TSN and two-stream I3D.

Figures 7.9, 7.10 and 7.11 show a comparison of the class recall from the best combinations and the maximum recall among the individual streams. The scores are averaged over the splits. We indicate whether there is an increase or decrease through the bar color. Green bars mean that the corresponding classes are better recognized by the combination than by the stream with the best score. Otherwise, the bar is colored red. We can see that the increases are more frequent and more significant than the decreases on both datasets, especially in the most difficult classes. The classes with the highest decreases, "CricketShot" on UCF101 and "swing_baseball" on HMDB51, are in the top-5 confusions of various streams (Tables 7.7 and 7.8). Furthermore, only the OF achieves good results for them, presenting scores significantly higher than the other streams (Figures 7.5, 7.6 and 7.7). Therefore, the SVM might be negatively affected by the frequency of incorrect score vectors that it receives.

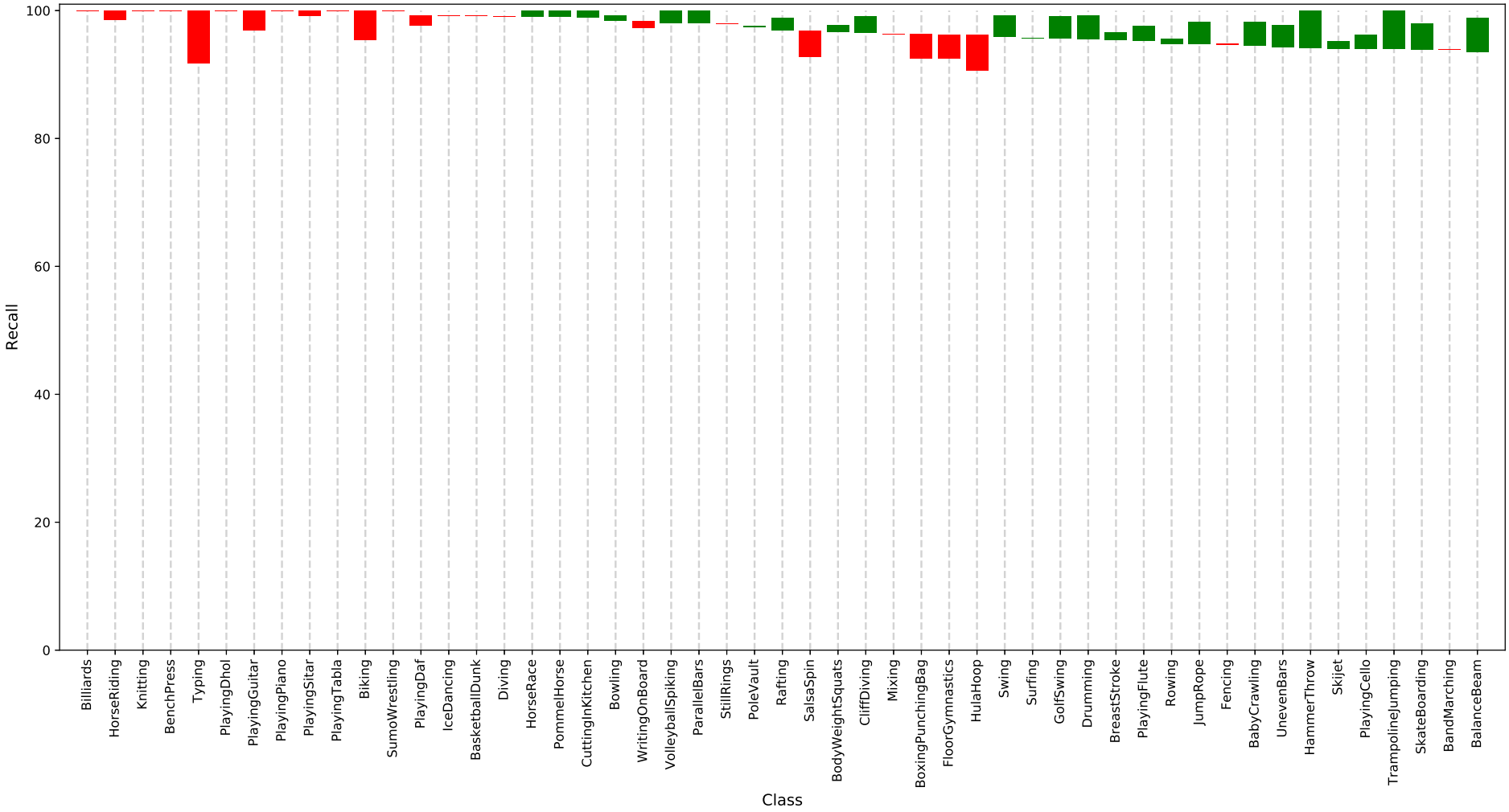


Figure 7.9: Class recall for the combination RGB* + OF + AVR + LVR_1 + LVR_2 compared to the maximum recall among the streams on UCF101. The scores were averaged over all splits. We follow the same order used in Figure 7.5. UCF101 was divided into two groups for illustration purposes, here we show the first 51 classes. Green bars indicate an increase in the recall value from individual stream to the fusion. Red bars indicate a decrease.

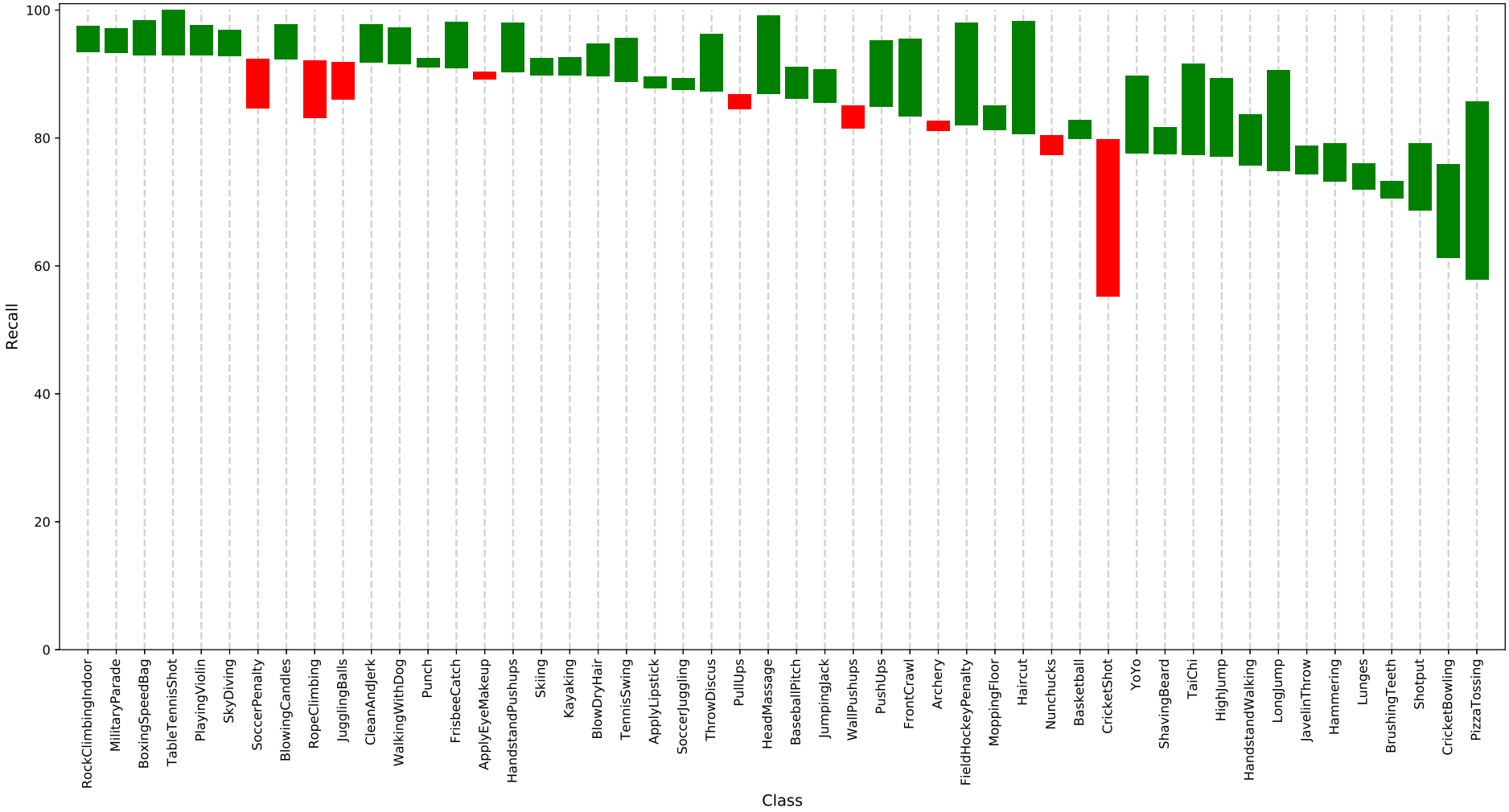


Figure 7.10: Class recall for the combination RGB* + OF + AVR + LVR_1 + LVR_2 compared to the maximum recall among the streams on UCF101. The scores were averaged over all splits. We follow the same order used in Figure 7.6. UCF101 was divided into two groups for illustration purposes, here we show the last 50 classes. Green bars indicate an increase in the recall value from individual stream to the fusion. Red bars indicate a decrease.

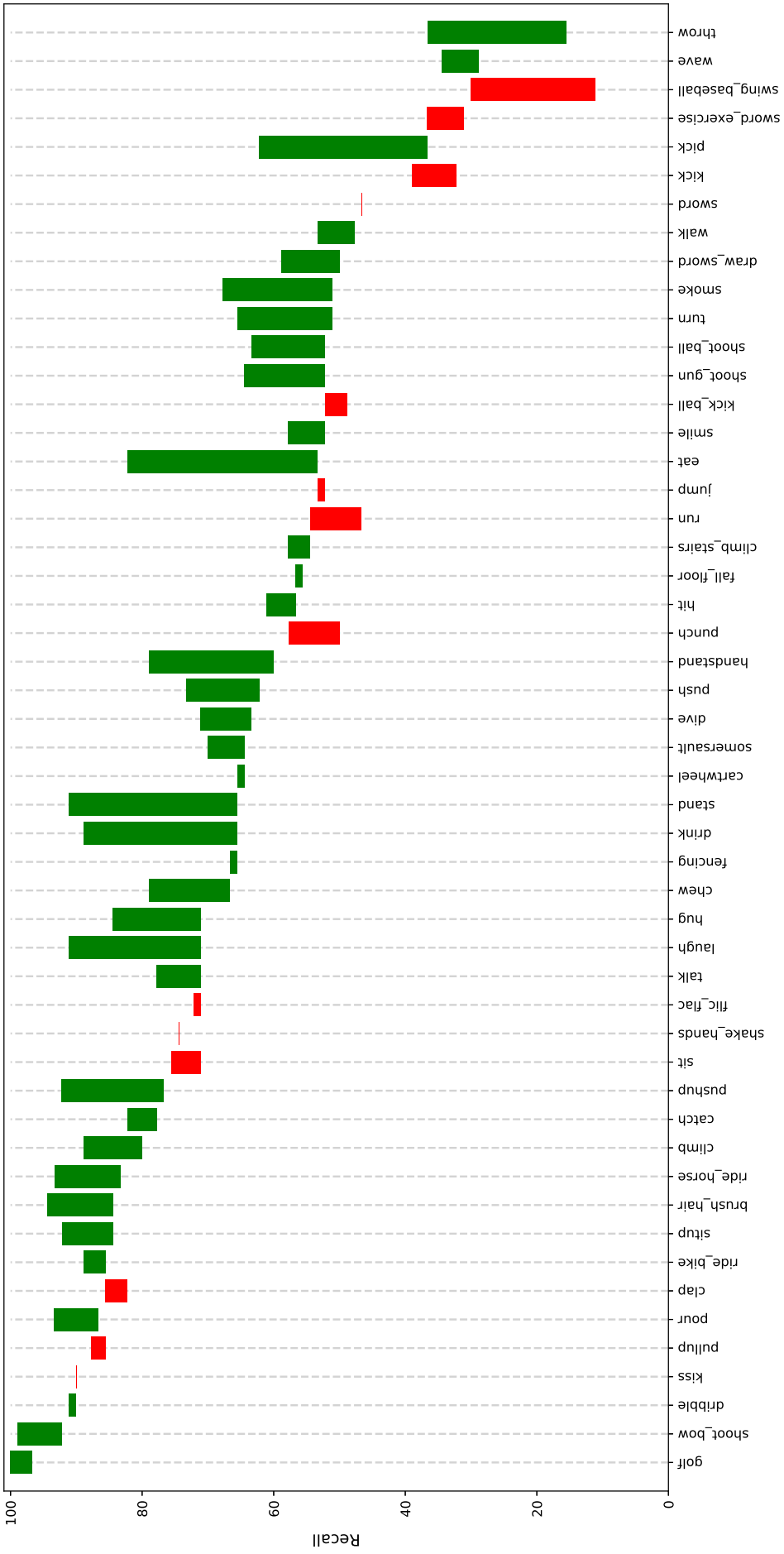


Figure 7.11: Class recall for the combination RGB* + OF + AVR + LVR_0 + LVR_1 + LVR_2 compared to the maximum recall among the streams on HMDB51. The scores were averaged over all splits. We follow the same order used in Figure 7.7. Green bars indicate an increase in the recall value from individual stream to the fusion. Red bars indicate a decrease.

Chapter 8

Conclusions and Future Work

Human action recognition is a challenging and attractive problem due to the wide range of possible applications. Although much effort has been made in this research field, there is no generic methodology for solving the problem and many questions remain open. The perception of the problem itself evolves as new datasets are released.

Throughout this text, we presented our research achievements for HAR in videos. The central issue of any application involving video analysis is the definition of a proper spatio-temporal representation that describes the event of interest. We explored deep learning strategies for this task that learns complex visual patterns from data. To minimize the high training cost of video-based deep networks, we follow the trend of exploring non-trainable elements from traditional methods in image-based ones. Thus, we used handcrafted inputs that encode the input video in an image form. Our proposed architecture is based on the multi-stream architecture [113], exploring complementary image modalities. In addition to the original spatial and temporal streams, here we introduced new ones that work with visual rhythms. Visual rhythms handle different video lengths and encode long-term information.

Our first approach, the Adaptive Visual Rhythm (AVR), was part of a collaborative project. The corresponding stream receives the horizontal-mean or vertical-mean rhythm as input, which represents the movement of objects by means changes in intensity over time. We proposed a method to adaptively decide the best direction for each video. We compared the accuracies of this stream with (AVR-K) and without (AVR) a pre-training on the large Kinetics. Although the AVR-K significantly outperformed the AVR in individual results, the increase was not reflected in the fusion. The accuracy rates were approximately 0.2% higher with the AVR-K in the combination, despite increases of 1.94% (UCF101) and 9.28% (HMDB51) in individual results.

The handcrafted AVR evolved into a learnable one (Learnable Visual Rhythm) composed of a feature-extractor and classification CNNs. We showed that the LVR achieves higher scores in both the individual and the combined scenarios. We also showed a comparison of the proposed methods against state-of-the-art approaches. Although our method achieves competitive results compared to those pre-trained only on ImageNet, we are behind those pre-trained on both ImageNet and Kinetics.

We tested new fusion methods by stacking a meta-classifier on top of the streams. Our initial experiments suggested that the trainable fusion was affected by overfitting.

Thus, we carried out a study on overfitting in the streams and tested an early stopping protocol to reduce it. We showed that the overfitting occurs at early epochs of every stream-training, except in the temporal one. Based on this behavior, we tested different parameters making the early stopping protocol more restricted or more flexible according to the stream.

We showed an extensive analysis of the stream performances regarding the dataset classes. We also assessed how much they contribute to the combinations, using pairwise and non-pairwise measures.

We tested three fusion methods with trainable parameters: (i) weighted average (ii) fully connected layer and (iii) SVM classifier. We selected the SVM based on its results in the split 1 of the datasets. We showed the top-5 combinations using the SVM for the three splits. The best combinations reached higher scores than the best stream for most of the classes, including the most difficult ones. We also showed lower cost combinations that achieved satisfactory results, in particular some that do not include the expensive temporal stream, which may be indicated for scenarios with limited computational resources.

The action recognition datasets usually contain a wide range of classes in different scenarios. Thus, the knowledge acquired by the networks may be useful to related problems. For instance, the HMDB51 contain classes such as "clap" and "wave" that may be interesting for human-computer interaction systems, whereas "punch" and "run" may be used to detect anomalous events. Therefore, a transfer learning strategy may be employed to address these related problems, as proposed by Leite et al. [63].

With the experiments presented here, we are now able to answer the research questions formulated in Chapter 1.

Q1. Can visual rhythm, as a longer-term feature, provide complementary information to characterize the action in a multi-stream scenario?

Answer: Through the experiments described in Sections 7.2 and 7.3, we showed that the AVR and LVR versions were able to provide complementary information for the spatial and temporal streams. The AVR tends to better complement the spatial stream, whereas the LVR achieves higher scores combined with the temporal one. Furthermore, we showed that the two VR-based approaches can even complement each other.

Q2. Does the temporal concatenation of the outputs generated by an image CNN (Learnable VR) present relevant patterns to characterize the action?

Answer: From the experiments reported in Chapter 6, we showed that the LVR versions achieved effective results on both datasets, especially the LVR_1 and LVR_2 that reached scores comparable to the spatial stream.

Q3. Can the Learnable VR outperform our baseline Adaptive VR?

Answer: The LVR_1 and LVR_2 achieved significantly higher results than the AVR. This was not the case for LVR_0. We suspect that the LVR_0 may lack information about the scene structure, since it was extracted from earlier layers of the CNN. These results were discussed in Chapter 6.

Q4. Can the early stopping protocol reduce the overfitting on individual streams and improve the fusion performance?

Answer: The early stopping strategy brought some benefits to the architecture, such as the considerable reduction in the number of training epochs. However, in our experiments, it was not able to reduce overfitting, since it affected the test accuracy, although it reduced the difference between training and test accuracies (Section 7.1). It is necessary to test other strategies to reduce overfitting and evaluate its impact on the fusion performance.

The work presented here can be extended in several ways. In the following, we outline some potential directions for future research:

- Explore other strategies to prevent overfitting, such as the use of CNNs other than the Inception V3.
- Test an end-to-end multi-stream architecture, training all the streams simultaneously.
- Pre-train the LVR with the Kinetics and combine the resulting stream with the I3D architectures.
- Test the data augmentation technique proposed by Tacon et al. [125], which is proper for visual rhythms in LVR.
- Employ a cross-dataset training to test the generalization power of the proposed features.
- Apply the proposed method to other related problems, such as abnormal event detection, hand gesture recognition and fall detection. Since the datasets for these problems are usually smaller than the human action recognition ones, the CNN parameters obtained with the HAR training may be useful to initialize the architectures for related problems.

Bibliography

[1] J. K. Aggarwal and M. S. Ryoo. Human Activity Analysis: A Review. *ACM Computing Surveys*, 43(3), 2011.

[2] A. Akutsu and Y. Tonomura. Video Tomography: An Efficient Method for Camerawork Extraction and Motion Analysis. In *International Conference on Multimedia*, pages 349–356, San Francisco, California, USA, 1994. ACM.

[3] S. M. Amiri, M. T. Pourazad, P. Nasiopoulos, and V. C. Leung. Non-intrusive Human Activity Monitoring in a Smart Home Environment. In *International Conference on e-Health Networking, Applications and Services*, pages 606–610. IEEE, 2013.

[4] F. Baumann, J. Lao, A. Ehlers, and B. Rosenhahn. Motion Binary Patterns for Action Recognition. In *International Conference on Pattern Recognition Applications and Methods*, pages 385–392, 2014.

[5] H. Bilen, B. Fernando, E. Gavves, and A. Vedaldi. Action Recognition with Dynamic Image Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2799–2813, 2017.

[6] P. Bilinski and F. Bremond. Video Covariance Matrix Logarithm for Human Action Recognition in Videos. In *International Joint Conference on Artificial Intelligence*, 2015.

[7] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri. Actions as Space-Time Shapes. In *International Conference on Computer Vision*, pages 1395–1402, Beijing, China, 2005.

[8] Y. Bo, Y. Lu, and W. He. Few-Shot Learning of Video Action Recognition Only Based on Video Contents. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, March 2020.

[9] R. Bolles and H. Baker. Epipolar-Plane Image Analysis: A Technique for Analyzing Motion Sequences. In *IEEE Workshop on Computer Vision, Representation, and Control*, pages 168–178. IEEE, Oct. 1985.

[10] J.-Y. Bouguet. Pyramidal Implementation of the Affine Lucas Kanade Feature Tracker Description of the Algorithm. *Intel Corporation*, 5(1-10):4, 2001.

[11] E. Brill and J. Wu. Classifier Combination for Improved Lexical Disambiguation. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, volume 1, pages 191–195, 1998.

[12] F. A. Caetano, M. B. Vieira, and R. L. Souza da Silva. A Video Descriptor using Orientation Tensors and Shape-based Trajectory Clustering. *International Journal of Image and Graphics*, 16(04), 2016.

[13] J. Carreira, E. Noland, A. Banki-Horvath, C. Hillier, and A. Zisserman. A Short Note about Kinetics-600. *arXiv preprint arXiv:1808.01340*, 2018.

[14] J. Carreira, E. Noland, C. Hillier, and A. Zisserman. A Short Note on the Kinetics-700 Human Action Dataset. *arXiv preprint arXiv:1907.06987*, 2019.

[15] J. Carreira and A. Zisserman. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4724–4733. IEEE, 2017.

[16] J. Cavazza, A. Zunino, M. S. Biagio, and V. Murino. Kernelized Covariance for Action Recognition. In *International Conference on Pattern Recognition*, pages 408–413, 2016.

[17] C.-C. Chang and C.-J. Lin. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):1–27, 2011.

[18] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the Devil in the Details: Delving Deep into Convolutional Nets. In *British Machine Vision Conference*. BMVA Press, 2014.

[19] H. L. Chaves, K. S. Ribeiro, A. S. Brito, H. Tacon, M. B. Vieira, A. S. Cerqueira, S. M. Villela, H. Maia, D. T. Concha, and H. Pedrini. Filter Learning from Deep Descriptors of a Fully Convolutional Siamese Network for Tracking in Videos. In *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, volume 4, pages 685–694. INSTICC, SciTePress, 2020.

[20] E. Chen, X. Bai, L. Gao, H. C. Tinega, and Y. Ding. A Spatiotemporal Heterogeneous Two-stream Network for Action Recognition. *IEEE Access*, 7:57267–57275, 2019.

[21] S. Cheng, J. Yang, Z. Ma, and M. Xie. Action Recognition Based on Spatio-temporal Log-Euclidean Covariance Matrix. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 9(2):95–106, 2016.

[22] V. Choutas, P. Weinzaepfel, J. Revaud, and C. Schmid. PoTion: Pose MoTion Representation for Action Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[23] D. T. Concha. Multi-Stream Convolutional Neural Networks for Action Recognition in Video Sequences Based on Spatio-Temporal Information. Master's thesis, Institute of Computing, University of Campinas, Campinas, Brazil, 2019.

[24] D. T. Concha, H. A. Maia, H. Pedrini, H. Tacon, A. S. Brito, H. L. Chaves, and M. B. Vieira. Multi-Stream Convolutional Neural Networks for Action Recognition in Video Sequences Based on Adaptive Visual Rhythms. In *IEEE International Conference on Machine Learning and Applications*, Orlando, FL, USA, Dec. 2018. IEEE.

[25] C. Cortes and V. Vapnik. Support-Vector Networks. *Machine Learning*, 20, 1995.

[26] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893. IEEE, 2005.

[27] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray. Scaling Egocentric Vision: The Epic-kitchens Dataset. In *European Conference on Computer Vision*, pages 720–736, 2018.

[28] A. Diba, V. Sharma, and L. Van Gool. Deep Temporal Linear Encoding Networks. In *IEEE conference on Computer Vision and Pattern Recognition*, pages 2329–2338, 2017.

[29] B. G. Fabian Caba Heilbron, Victor Escorcia and J. C. Niebles. ActivityNet: A Large-Scale Video Benchmark for Human Activity Understanding. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–970, 2015.

[30] F. A. Faria. *A Framework for Pattern Classifier Selection and Fusion*. PhD thesis, Institute of Computing, University of Campinas, Campinas, Brazil, 2014.

[31] G. Farnebäck. Two-Frame Motion Estimation Based on Polynomial Expansion. In *Scandinavian Conference on Image Analysis*, pages 363–370. Springer, 2003.

[32] C. Feichtenhofer, H. Fan, J. Malik, and K. He. SlowFast Networks for Video Recognition. In *IEEE International Conference on Computer Vision*, pages 6202–6211, 2019.

[33] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional Two-stream Network Fusion for Video Action Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1933–1941, 2016.

[34] A. M. Figueiredo, H. A. Maia, F. L. Oliveira, V. F. Mota, and M. B. Vieira. A Video Tensor Self-Descriptor Based on Block Matching. In *International Conference on Computational Science and Its Applications*, pages 401–414. Springer, 2014.

[35] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[36] I. Gori, J. K. Aggarwal, L. Matthies, and M. S. Ryoo. Multitype Activity Recognition in Robot-Centric Scenarios. *IEEE Robotics and Automation Letters*, 1(1):593–600, Jan. 2016.

[37] R. Goyal, S. E. Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, F. Hoppe, C. Thurau, I. Bax, and R. Memisevic. The "Something Something" Video Database for Learning and Evaluating Visual Common Sense. In *IEEE International Conference on Computer Vision*, Oct 2017.

[38] C. Gu, C. Sun, D. A. Ross, C. Vondrick, C. Pantofaru, Y. Li, S. Vijayanarasimhan, G. Toderici, S. Ricco, R. Sukthankar, C. Schmid, and J. Malik. AVA: A Video Dataset of Spatio-Temporally Localized Atomic Visual Actions. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2018.

[39] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen. Recent Advances in Convolutional Neural Networks. *Pattern Recognition*, 77:354–377, 2017.

[40] K. Guo, P. Ishwar, and J. Konrad. Action Recognition from Video Using Feature Covariance Matrices. *IEEE Transactions on Image Processing*, 22(6):2479–2494, June 2013.

[41] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[42] S. Herath, M. Harandi, and F. Porikli. Going Deeper into Action Recognition: A Survey. *Image and Vision Computing*, 2017.

[43] B. K. Horn and B. G. Schunck. Determining Optical Flow. *Artificial Intelligence*, 17(1-3):185–203, 1981.

[44] M. Ilyes Lakhal, A. Clapés, S. Escalera, O. Lanz, and A. Cavallaro. Residual Stacked RNNs for Action Recognition. In *European Conference on Computer Vision*, pages 534–548, 2018.

[45] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating Local Descriptors into a Compact Image Representation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3304–3311. IEEE, 2010.

[46] S. Ji, W. Xu, M. Yang, and K. Yu. 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, 2013.

[47] R. Kahani, A. Talebpour, and A. Mahmoudi-Aznaveh. A Correlation Based Feature Representation for First-Person Activity Recognition. *Multimedia Tools and Applications*, 78(15):21673–21694, 2019.

[48] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-Scale Video Classification with Convolutional Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[49] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman. The Kinetics Human Action Video Dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[50] V. Kellokumpu, G. Zhao, and M. Pietikäinen. Human Activity Recognition using a Dynamic Texture Based Method. In *British Machine Vision Conference*, volume 1, page 2, 2008.

[51] H. Kim, J. Lee, J.-H. Yang, S. Sull, W. M. Kim, and S. M.-H. Song. Visual Rhythm and Shot Verification. *Multimedia Tools and Applications*, 15(3):227–245, 2001.

[52] J. Kim, S. Cha, D. Wee, S. Bae, and J. Kim. Regularization on Spatio-Temporally Smoothed Feature for Action Recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12103–12112, 2020.

[53] A. Klaser, M. Marszałek, and C. Schmid. A Spatio-Temporal Descriptor Based on 3D-Gradients. In *British Machine Vision Conference*, pages 275–1. British Machine Vision Association, 2008.

[54] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[55] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: A Large Video Database for Human Motion Recognition. In *International Conference on Computer Vision*, 2011.

[56] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley & Sons, 2004.

[57] B. Kwolek and M. Kepski. Human Fall Detection on Embedded Platform Using Depth Maps and Wireless Accelerometer. *Computer Methods and Programs in Biomedicine*, 117(3):489–501, 2014.

[58] H. Kwon, Y. Kim, J. S. Lee, and M. Cho. First Person Action Recognition via Two-stream Convnet with Long-term Fusion Pooling. *Pattern Recognition Letters*, 112:161–167, 2018.

[59] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning Realistic Human Actions from Movies. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[60] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov. 1998.

[61] Y. LeCun, K. Kavukcuoglu, and C. Farabet. Convolutional Networks and Applications in Vision. In *IEEE International Symposium on Circuits and Systems*, pages 253–256. IEEE, 2010.

[62] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. In *International Conference on Machine Learning*, pages 609–616, 2009.

[63] G. Leite, G. Silva, and H. Pedrini. Fall Detection in Video Sequences Based on a Three-Stream Convolutional Neural Network. In *IEEE International Conference On Machine Learning And Applications*, pages 191–195. IEEE, 2019.

[64] H. Li, J. Chen, and R. Hu. Multiple Feature Fusion in Convolutional Neural Networks for Action Recognition. *Wuhan University Journal of Natural Sciences*, 22(1):73–78, 2017.

[65] Y. Li, J. Gao, Q. Li, and W. Fan. Ensemble Learning. In *Data Classification: Algorithms and Applications*, pages 483–509. Chapman and Hall/CRC, 1st edition, 2014.

[66] Y. Li, B. Ji, X. Shi, J. Zhang, B. Kang, and L. Wang. TEA: Temporal Excitation and Aggregation for Action Recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 909–918, 2020.

[67] J. Lin, C. Gan, and S. Han. TSM: Temporal Shift Module for Efficient Video Understanding. In *IEEE International Conference on Computer Vision*, pages 7083–7093, 2019.

[68] J. Liu, J. Luo, and M. Shah. Recognizing Realistic Actions from Videos "In the Wild". In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1996–2003. IEEE, 2009.

[69] B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *International Joint Conference on Artificial Intelligence*, volume 2, pages 674–679. Vancouver, BC, Canada, 1981.

[70] H. A. Maia, D. T. Concha, H. Pedrini, H. Tacon, A. S. Brito, H. L. Chaves, M. B. Vieira, and S. M. Villela. Action Recognition in Videos Using Multi-Stream Convolutional Neural Networks. In *Deep Learning Applications*, pages 95–111. Springer, 2020.

[71] H. A. Maia, A. M. O. Figueiredo, F. L. M. Oliveira, V. F. Mota, and M. B. Vieira. A Video Tensor Self-Descriptor Based on Variable Size Block Matching. *Journal of Mobile Multimedia*, 11(1&2):090–102, 2015.

[72] H. A. Maia, M. R. Souza, A. Santos, H. Pedrini, H. Tacon, A. S. Brito, H. L. Chaves, M. B. Vieira, and S. M. Villela. Learnable Visual Rhythms Based on the Stacking of Convolutional Neural Networks for Action Recognition. In *IEEE International*

Conference on Machine Learning and Applications, pages 1794–1799, Boca Raton, FL, USA, Dec. 2019. IEEE.

[73] M. Marszałek, I. Laptev, and C. Schmid. Actions in Context. In *IEEE Conference on Computer Vision & Pattern Recognition*, 2009.

[74] G. A. Miller. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[75] T. B. Moeslund, A. Hilton, and V. Krüger. A Survey of Advances in Vision-based Human Motion Capture and Analysis. *Computer Vision and Image Understanding*, 104(2-3):90–126, 2006.

[76] M. Monfort, A. Andonian, B. Zhou, K. Ramakrishnan, S. A. Bargal, T. Yan, L. Brown, Q. Fan, D. Gutfreund, and C. Vondrick. Moments in Time Dataset: One Million Videos for Event Understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):502–508, 2019.

[77] T. P. Moreira, D. Menotti, and H. Pedrini. Video Action Recognition Based on Visual Rhythm Representation. *Journal of Visual Communication and Image Representation*, page 102771, 2020.

[78] V. F. Mota, E. A. Perez, L. M. Maciel, M. B. Vieira, and P. H. Gosselin. A Tensor Motion Descriptor Based on Histograms of Gradients and Optical Flow. *Pattern Recognition Letters*, 39:85–91, 2014.

[79] A. Nagrani, C. Sun, D. Ross, R. Sukthankar, C. Schmid, and A. Zisserman. Speech2Action: Cross-modal Supervision for Action Recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10317–10326, 2020.

[80] V. Nair and G. E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *International Conference on Machine Learning*, page 807–814, Madison, WI, USA, 2010. Omnipress.

[81] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond Short Snippets: Deep Networks for Video Classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4694–4702, 2015.

[82] C.-W. Ngo, T.-C. Pong, and R. T. Chin. Camera Break Detection by Partitioning of 2D Spatio-Temporal Images in MPEG Domain. In *IEEE International Conference on Multimedia Computing and Systems*, volume 1, pages 750–755. IEEE, 1999.

[83] C.-W. Ngo, T.-C. Pong, and R. T. Chin. Detection of Gradual Transitions through Temporal Slice Analysis. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 36–41. IEEE, 1999.

[84] C.-W. Ngo, T.-C. Pong, and H.-J. Zhang. Motion Analysis and Segmentation through Spatio-Temporal Slices Processing. *IEEE Transactions on Image Processing*, 12(3):341–355, 2003.

[85] U. D. of Labor. American Time Use Survey. `https://www.bls.gov/tus/`, 2020 (Last access: September 01, 2020).

[86] T. Ojala, M. Pietikäinen, and D. Harwood. A Comparative Study of Texture Measures with Classification Based on Featured Distributions. *Pattern Recognition*, 29(1):51–59, 1996.

[87] E. Park, X. Han, T. L. Berg, and A. C. Berg. Combining Multiple Sources of Knowledge in Deep CNNs for Action Recognition. In *IEEE Winter Conference on Applications of Computer Vision*, pages 1–8. IEEE, 2016.

[88] C. I. Patel, S. Garg, T. Zaveri, A. Banerjee, and R. Patel. Human Action Recognition using Fusion of Features for Unconstrained Video Sequences. *Computers & Electrical Engineering*, 2016.

[89] S.-L. Peng. Temporal Slice Analysis of Image Sequences. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 283–288. IEEE, 1991.

[90] X. Peng, L. Wang, X. Wang, and Y. Qiao. Bag of Visual Words and Fusion Methods for Action Recognition: Comprehensive Study and Good Practice. *Computer Vision and Image Understanding*, 150:109–125, 2016.

[91] E. A. Perez, V. F. Mota, L. M. Maciel, D. Sad, and M. B. Vieira. Combining Gradient Histograms using Orientation Tensors for Human Action Recognition. In *International Conference on Pattern Recognition*, pages 3460–3463. IEEE, 2012.

[92] F. Perronnin and C. Dance. Fisher Kernels on Visual Vocabularies for Image Categorization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[93] H.-H. Phan, N.-S. Vu, V.-L. Nguyen, and M. Quoy. Motion of Oriented Magnitudes Patterns for Human Action Recognition. In *International Symposium on Visual Computing*, pages 168–177. Springer, 2016.

[94] R. Poppe. A Survey on Vision-based Human Action Recognition. *Image and Vision Computing*, 28(6):976–990, 2010.

[95] L. Prechelt. Early Stopping - But When? In *Neural Networks: Tricks of the Trade*, pages 55–69. Springer, 1998.

[96] S. Raschka. *Python Machine Learning*. Packt Publishing, 2015.

[97] M. Ravanbakhsh, H. Mousavi, M. Rastegari, V. Murino, and L. S. Davis. Action Recognition with Image based CNN Features. *arXiv preprint arXiv:1512.03980*, 2015.

[98] K. K. Reddy and M. Shah. Recognizing 50 Human Action Categories of Web Videos. *Machine Vision and Applications*, 24(5):971–981, 2012.

[99] Y. Ricquebourg and P. Bouthemy. Real-Time Tracking of Moving Persons by Exploiting Spatio-Temporal Image Slices. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 22(8):797–808, 2000.

[100] M. D. Rodriguez, J. Ahmed, and M. Shah. Action Mach a Spatio-temporal Maximum Average Correlation Height Filter for Action Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[101] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[102] M. S. Ryoo and L. Matthies. First-Person Activity Recognition: Feature, Temporal Structure, and Prediction. *International Journal of Computer Vision*, 119(3):307–328, Sept. 2016.

[103] D. Sad, V. F. Mota, L. M. Maciel, M. B. Vieira, and A. A. Araujo. A Tensor Motion Descriptor Based on Multiple Gradient Estimators. In *Conference on Graphics, Patterns and Images*, pages 70–74. IEEE, 2013.

[104] H. Sagan. *Space-Filling Curves*. Springer Science & Business Media, 2012.

[105] A. Sanin, C. Sanderson, M. T. Harandi, and B. C. Lovell. Spatio-Temporal Covariance Descriptors for Action and Gesture Recognition. In *IEEE Workshop on Applications of Computer Vision*, pages 103–110. IEEE, 2013.

[106] A. C. S. Santos. *Spatio-Temporal Representation Based on Autoencoder for Video Action Recognition*. PhD thesis, Institute of Computing, University of Campinas, Campinas, Brazil, 2019.

[107] A. C. S. Santos, H. A. Maia, M. R. Souza, M. B. Vieira, and H. Pedrini. Fuzzy Fusion for Two-stream Action Recognition. In *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, volume 5, pages 117–123. INSTICC, SciTePress, 2020.

[108] A. B. Sargano, P. Angelov, and Z. Habib. A Comprehensive Review on Handcrafted and Learning-based Action Representation Approaches for Human Activity Recognition. *Applied Sciences*, 7(1):110, 2017.

[109] C. Schuldt, I. Laptev, and B. Caputo. Recognizing Human Actions: A Local SVM Approach. In *International Conference on Pattern Recognition*, volume 3, pages 32–36. IEEE, 2004.

[110] F. Shi, R. Laganiere, and E. Petriu. Gradient Boundary Histograms for Action Recognition. In *IEEE Winter Conference on Applications of Computer Vision*, pages 1107–1114. IEEE, 2015.

[111] J. Shi and C. Tomasi. Good Features to Track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600. IEEE, 1994.

[112] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta. Hollywood in Homes: Crowdsourcing Data Collection for Activity Understanding. In *European Conference on Computer Vision*, pages 510–526. Springer, 2016.

[113] K. Simonyan and A. Zisserman. Two-Stream Convolutional Networks for Action Recognition in Videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014.

[114] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Y. Bengio and Y. LeCun, editors, *International Conference on Learning Representations*, 2015.

[115] A. C. Sintes. *Learning to Recognize Human Actions: from Hand-crafted to Deep-learning Based Visual Representations*. PhD thesis, Departament de Matemàtiques i Informàtica, Universitat de Barcelona, Barcelona, Spain, 2018.

[116] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A Dataset of 101 Human Actions Classes from Videos in the Wild. *arXiv preprint arXiv:1212.0402*, 2012.

[117] M. R. Souza. Digital Video Stabilization: Algorithms and Evaluation. Master's thesis, Institute of Computing, University of Campinas, Campinas, Brazil, 2018.

[118] M. R. Souza, H. A. Maia, M. B. Vieira, and H. Pedrini. Survey on Visual Rhythms: A Spatio-Temporal Representation for Video Sequences. *Neurocomputing*, 402:409–422, 2020.

[119] Statista. Hours of Video Uploaded to YouTube Every Minute as of May 2019. `https://www.statista.com/statistics/259477/hours-of-video-uploaded-to-youtube-every-minute/`, 2020 (Last access: March 25, 2020).

[120] W. Sultani, C. Chen, and M. Shah. Real-world Anomaly Detection in Surveillance Videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6479–6488, 2018.

[121] L. Sun, K. Jia, K. Chen, D. Y. Yeung, B. E. Shi, and S. Savarese. Lattice Long Short-Term Memory for Human Action Recognition. In *IEEE International Conference on Computer Vision*, pages 2166–2175, 2017.

[122] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the Importance of Initialization and Momentum in Deep Learning. In *International Conference on Machine Learning*, pages 1139–1147, 2013.

[123] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception Architecture for Computer Vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[124] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer Science & Business Media, 2010.

[125] H. Tacon, A. S. Brito, H. L. Chaves, M. B. Vieira, S. M. Villela, H. Almeida Maia, D. T. Concha, and H. Pedrini. Human Action Recognition Using Convolutional Neural Networks with Symmetric Time Extension of Visual Rhythms. In *International Conference on Computational Science and Its Applications*, pages 351–366. Springer, 2019.

[126] H. Tacon, A. S. Brito, H. L. Chaves, M. B. Vieira, S. M. Villela, H. Maia, D. T. Concha, and H. Pedrini. Multi-stream Architecture with Symmetric Extended Visual Rhythms for Deep Learning Human Action Recognition. In *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, volume 5, pages 351–358. INSTICC, SciTePress, 2020.

[127] M. Tan and Q. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In K. Chaudhuri and R. Salakhutdinov, editors, *International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[128] B. S. Torres and H. Pedrini. Detection of Complex Video Events through Visual Rhythm. *The Visual Computer*, pages 1–21, 2016.

[129] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning Spatiotemporal Features with 3D Convolutional Networks. In *IEEE International Conference on Computer Vision*, pages 4489–4497, 2015.

[130] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A Closer Look at Spatiotemporal Convolutions for Action Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.

[131] O. Tuzel, F. Porikli, and P. Meer. Region Covariance: A Fast Descriptor for Detection and Classification. In *European Conference on Computer Vision*, pages 589–600. Springer, 2006.

[132] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik. Action Recognition in Video Sequences using Deep Bi-Directional LSTM With CNN Features. *IEEE Access*, 6:1155–1166, 2018.

[133] F. B. Valio, H. Pedrini, and N. J. Leite. Fast Rotation-Invariant Video Caption Detection Based on Visual Rhythm. In *Iberoamerican Congress on Pattern Recognition*, pages 157–164. Springer, 2011.

[134] G. Varol, I. Laptev, and C. Schmid. Long-Term Temporal Convolutions for Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6):1510–1517, 2018.

[135] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Action Recognition by Dense Trajectories. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3169–3176. IEEE, 2011.

[136] H. Wang and C. Schmid. Action Recognition with Improved Trajectories. In *IEEE International Conference on Computer Vision*, pages 3551–3558. IEEE, 2013.

[137] H. Wang and C. Schmid. LEAR-INRIA Submission for the THUMOS Workshop. In *Workshop on Action Recognition with a Large Number of Classes*, 2013.

[138] H. Wang, Y. Yang, E. Yang, and C. Deng. Exploring Hybrid Spatio-Temporal Convolutional Networks for Human Action Recognition. *Multimedia Tools and Applications*, 76(13):15065–15081, 2017.

[139] J. Wang, A. Cherian, F. Porikli, and S. Gould. Video Representation Learning Using Discriminative Pooling. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1149–1158, 2018.

[140] L. Wang, Y. Qiao, and X. Tang. Action Recognition with Trajectory-Pooled Deep-Convolutional Descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4305–4314, 2015.

[141] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards Good Practices for very Deep Two-Stream Convnets. *arXiv preprint arXiv:1507.02159*, 2015.

[142] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal Segment Networks: Towards Good Practices for Deep Action Recognition. In *European Conference on Computer Vision*, pages 20–36. Springer, 2016.

[143] L. Wang, J. Zhang, L. Zhou, C. Tang, and W. Li. Beyond Covariance: Feature Representation with Nonlinear Kernel Matrices. In *IEEE International Conference on Computer Vision*, pages 4570–4578, 2015.

[144] Y. Wang, M. Long, J. Wang, and P. S. Yu. Spatiotemporal Pyramid Network for Video Action Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2097–2106. IEEE, 2017.

[145] C. Yang, Y. Xu, J. Shi, B. Dai, and B. Zhou. Temporal Pyramid Network for Action Recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 591–600, 2020.

[146] Z. Yang, L. Zhu, Y. Wu, and Y. Yang. Gated Channel Transformation for Visual Recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11794–11803, 2020.

[147] H. Ye, Z. Wu, R.-W. Zhao, X. Wang, Y.-G. Jiang, and X. Xue. Evaluating Two-stream CNN for Video Classification. In *ACM on International Conference on Multimedia Retrieval*, pages 435–442, 2015.

[148] L. Yeffet and L. Wolf. Local Trinary Patterns for Human Action Recognition. In *IEEE International Conference on Computer Vision*, pages 492–497. IEEE, 2009.

[149] S. Yu, Y. Cheng, L. Xie, Z. Luo, M. Huang, and S. Li. A Novel Recurrent Hybrid Network for Feature Fusion in Action Recognition. *Journal of Visual Communication and Image Representation*, 49:192–203, 2017.

[150] Y. Yun, I. Y.-H. Gu, and H. Aghajan. Riemannian Manifold-based Support Vector Machine for Human Activity Classification in Images. In *IEEE International Conference on Image Processing*, pages 3466–3469. IEEE, 2013.

[151] C. Zach, T. Pock, and H. Bischof. A Duality Based Approach for Realtime TV-L1 Optical Flow. In *Joint Pattern Recognition Symposium*, pages 214–223. Springer, 2007.

[152] G. Zhao and M. Pietikainen. Dynamic Texture Recognition using Local Binary Patterns with an Application to Facial Expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6), 2007.

[153] F. Zheng, S. Li, H. Wu, and J. Feng. Anchor Shot Detection with Diverse Style Backgrounds Based on Spatial-Temporal Slice Analysis. In *International Conference on Multimedia Modeling*, pages 676–682. Springer, 2010.

[154] B. Zhou, A. Andonian, A. Oliva, and A. Torralba. Temporal Relational Reasoning in Videos. In *European Conference on Computer Vision*, pages 803–818, 2018.

[155] Z.-H. Zhou. *Ensemble Methods: Foundations and Algorithms*. CRC press, 2012.

[156] F. Zhu, L. Shao, J. Xie, and Y. Fang. From Handcrafted to Learned Representations for Human Action Recognition: A Survey. *Image and Vision Computing*, 55:42–52, 2016.

[157] J. Zhu, Z. Zhu, and W. Zou. End-to-End Video-level Representation Learning for Action Recognition. In *International Conference on Pattern Recognition*, pages 645–650. IEEE, 2018.

[158] J. Zhu, W. Zou, and Z. Zhu. Two-stream Gated Fusion Convnets for Action Recognition. In *International Conference on Pattern Recognition*, pages 597–602. IEEE, 2018.

[159] W. Zhu, J. Hu, G. Sun, X. Cao, and Y. Qiao. A Key Volume Mining Deep Framework for Action Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1991–1999. IEEE, 2016.

[160] Y. Zhu. PyTorch Implementation of Popular Two-Stream Frameworks for Video Action Recognition. https://github.com/bryanyzhu/two-stream-pytorch, 2018 (Last access: September 24, 2020).

Appendix A

Diversity Measures - Demonstration

In Section 2.9, we argued that four of the pairwise diversity metrics (disagreement, Q-statistic, correlation coefficient and kappa-statistic) cannot capture disparities between b and c. We mentioned a toy example where all of them achieve the perfect (or almost perfect) score. Here, we demonstrate this fact. Considering the situation given in Table A.1, we show the formulas and resulting score for each measure.

Table A.1: Toy example: c_j gives an almost perfect set of predictions whereas c_i fails in most samples.

	Hit c_i	Miss c_i
Hit c_j	a	b
Miss c_j	c	d

	Hit c_i	Miss c_i
Hit c_j	0	$n_s - 1$
Miss c_j	1	0

- Disagreement ($DM(c_i, c_j)$):

$$DM(c_i, c_j) = \frac{b + c}{n_s} = \frac{(n_s - 1) + 1}{n_s} = 1$$

- Q-statistic ($Q(c_i, c_j)$):

$$Q(c_i, c_j) = \frac{a \cdot d - b \cdot c}{a \cdot d + b \cdot c} = \frac{0 \cdot 0 - (n_s - 1) \cdot 1}{0 \cdot 0 + (n_s - 1) \cdot 1} = -1$$

- Correlation coefficient ($\rho(c_i, c_j)$):

$$\begin{aligned}\rho(c_i, c_j) &= \frac{a \cdot d - b \cdot c}{\mid \sqrt{(a + b) \cdot (a + c) \cdot (b + d) \cdot (c + d)} \mid} \\ &= \frac{0 \cdot 0 - (n_s - 1) \cdot 1}{\mid \sqrt{(0 + (n_s - 1)) \cdot (0 + 1) \cdot ((n_s - 1) + 0) \cdot (1 + 0)} \mid} \\ &= \frac{-(n_s - 1)}{\mid \sqrt{(n_s - 1)^2} \mid} = -1\end{aligned}$$

- Kappa-statistic ($\kappa_p(c_i, c_j)$):

$$\kappa_p(c_i, c_j) = \frac{\theta_1 - \theta_2}{1 - \theta_2}$$

$$(1) \qquad \theta_1 = \frac{a+d}{n_s} = \frac{0+0}{n_s} = 0$$

$$\begin{aligned}(2) \qquad \theta_2 &= \frac{(a+b)\cdot(a+c) + (b+d)\cdot(c+d)}{n_s^2} \\ &= \frac{(0+(n_s-1))\cdot(0+1) + ((n_s-1)+0)\cdot(1+0)}{n_s^2} \\ &= \frac{2\cdot(n_s-1)}{n_s^2}\end{aligned}$$

$$\begin{aligned}\kappa_p(c_i, c_j) &= \frac{0 - \frac{2\cdot(n_s-1)}{n_s^2}}{\frac{n_s^2}{n_s^2} - \frac{2\cdot(n_s-1)}{n_s^2}} \\ &= \frac{-2\cdot(n_s-1)}{n_s^2 - 2\cdot(n_s-1)} \approx 0\end{aligned}$$

Appendix B

Reference Lists

In this appendix, we provide the reference lists containing the indices used in this text. Tables B.1 and B.2 contain the list of HMDB51 and UCF101 classes, respectively. Table B.3 contains the list of every r-combination using the six streams, the spatial RGB*, the temporal OF and the four VR-based versions AVR, LVR_0, LVR_1 and LVR_2, from Chapter 7.

Table B.1: The complete list of HMDB51 classes with the corresponding indices.

HMDB51					
0	brush_hair	17	hit	34	shoot_ball
1	cartwheel	18	hug	35	shoot_bow
2	catch	19	jump	36	shoot_gun
3	chew	20	kick	37	sit
4	clap	21	kick_ball	38	situp
5	climb	22	kiss	39	smile
6	climb_stairs	23	laugh	40	smoke
7	dive	24	pick	41	somersault
8	draw_sword	25	pour	42	stand
9	dribble	26	pullup	43	swing_baseball
10	drink	27	punch	44	sword
11	eat	28	push	45	sword_exercise
12	fall_floor	29	pushup	46	talk
13	fencing	30	ride_bike	47	throw
14	flic_flac	31	ride_horse	48	turn
15	golf	32	run	49	walk
16	handstand	33	shake_hands	50	wave

Table B.2: The complete list of UCF101 classes with the corresponding indices.

UCF101					
0	ApplyEyeMakeup	34	Hammering	68	PommelHorse
1	ApplyLipstick	35	HammerThrow	69	PullUps
2	Archery	36	HandstandPushups	70	Punch
3	BabyCrawling	37	HandstandWalking	71	PushUps
4	BalanceBeam	38	HeadMassage	72	Rafting
5	BandMarching	39	HighJump	73	RockClimbingIndoor
6	BaseballPitch	40	HorseRace	74	RopeClimbing
7	Basketball	41	HorseRiding	75	Rowing
8	BasketballDunk	42	HulaHoop	76	SalsaSpin
9	BenchPress	43	IceDancing	77	ShavingBeard
10	Biking	44	JavelinThrow	78	Shotput
11	Billiards	45	JugglingBalls	79	SkateBoarding
12	BlowDryHair	46	JumpingJack	80	Skiing
13	BlowingCandles	47	JumpRope	81	Skijet
14	BodyWeightSquats	48	Kayaking	82	SkyDiving
15	Bowling	49	Knitting	83	SoccerJuggling
16	BoxingPunchingBag	50	LongJump	84	SoccerPenalty
17	BoxingSpeedBag	51	Lunges	85	StillRings
18	BreastStroke	52	MilitaryParade	86	SumoWrestling
19	BrushingTeeth	53	Mixing	87	Surfing
20	CleanAndJerk	54	MoppingFloor	88	Swing
21	CliffDiving	55	Nunchucks	89	TableTennisShot
22	CricketBowling	56	ParallelBars	90	TaiChi
23	CricketShot	57	PizzaTossing	91	TennisSwing
24	CuttingInKitchen	58	PlayingCello	92	ThrowDiscus
25	Diving	59	PlayingDaf	93	TrampolineJumping
26	Drumming	60	PlayingDhol	94	Typing
27	Fencing	61	PlayingFlute	95	UnevenBars
28	FieldHockeyPenalty	62	PlayingGuitar	96	VolleyballSpiking
29	FloorGymnastics	63	PlayingPiano	97	WalkingWithDog
30	FrisbeeCatch	64	PlayingSitar	98	WallPushups
31	FrontCrawl	65	PlayingTabla	99	WritingOnBoard
32	GolfSwing	66	PlayingViolin	100	YoYo
33	Haircut	67	PoleVault		

Table B.3: List of every r-combination ($r \in \{2, 3, 4, 5, 6\}$) with the corresponding indices using the six streams.

	2-combinations
1	RGB* + OF
2	RGB* + AVR
3	RGB* + LVR_0
4	RGB* + LVR_1
5	RGB* + LVR_2
6	OF + AVR
7	OF + LVR_0
8	OF + LVR_1
9	OF + LVR_2
10	AVR + LVR_0
11	AVR + LVR_1
12	AVR + LVR_2
13	LVR_0 + LVR_1
14	LVR_0 + LVR_2
15	LVR_1 + LVR_2
	3-combinations
16	RGB* + OF + AVR
17	RGB* + OF + LVR_0
18	RGB* + OF + LVR_1
19	RGB* + OF + LVR_2
20	RGB* + AVR + LVR_0
21	RGB* + AVR + LVR_1
22	RGB* + AVR + LVR_2
23	RGB* + LVR_0 + LVR_1
24	RGB* + LVR_0 + LVR_2
25	RGB* + LVR_1 + LVR_2
26	OF + AVR + LVR_0
27	OF + AVR + LVR_1
28	OF + AVR + LVR_2
29	OF + LVR_0 + LVR_1
30	OF + LVR_0 + LVR_2
31	OF + LVR_1 + LVR_2
32	AVR + LVR_0 + LVR_1
33	AVR + LVR_0 + LVR_2
34	AVR + LVR_1 + LVR_2
35	LVR_0 + LVR_1 + LVR_2

	4-combinations
36	RGB* + OF + AVR + LVR_0
37	RGB* + OF + AVR + LVR_1
38	RGB* + OF + AVR + LVR_2
39	RGB* + OF + LVR_0 + LVR_1
40	RGB* + OF + LVR_0 + LVR_2
41	RGB* + OF + LVR_1 + LVR_2
42	RGB* + AVR + LVR_0 + LVR_1
43	RGB* + AVR + LVR_0 + LVR_2
44	RGB* + AVR + LVR_1 + LVR_2
45	RGB* + LVR_0 + LVR_1 + LVR_2
46	OF + AVR + LVR_0 + LVR_1
47	OF + AVR + LVR_0 + LVR_2
48	OF + AVR + LVR_1 + LVR_2
49	OF + LVR_0 + LVR_1 + LVR_2
50	AVR + LVR_0 + LVR_1 + LVR_2
	5-combinations
51	RGB* + OF + AVR + LVR_0 + LVR_1
52	RGB* + OF + AVR + LVR_0 + LVR_2
53	RGB* + OF + AVR + LVR_1 + LVR_2
54	RGB* + OF + LVR_0 + LVR_1 + LVR_2
55	RGB* + AVR + LVR_0 + LVR_1 + LVR_2
56	OF + AVR + LVR_0 + LVR_1 + LVR_2
	6-combinations
57	RGB* + OF + AVR + LVR_0 + LVR_1 + LVR_2