Action Recognition in Videos: Data-efficient approaches for supervised learning of human action classification models for video

César Roberto de Souza

Universitat Autònoma de Barcelona

Action Recognition in Videos: Data-efficient approaches for supervised learning of human action classification models for video

A dissertation submitted by **César Roberto de Souza** at the Departament de Ciències de la Computació, Universitat Autònoma de Barcelona to fulfil the degree of **Doctor of Philosophy** within the Computer Science Doctorate Program.

Bellaterra, March 23, 2018

Co-Director
& Tutor

Dr. Antonio Manuel López Peña
Dept. de Ciències de la Computació & Centre de Visió per Computador
Universitat Autònoma de Barcelona

Co-Director

Dr. Naila Murray
Computer Vision Group
NAVER LABS Europe

A meus pais Caio e Clotilde

Acknowledgements

FIRST and foremost, I thank all my family for the support, effort, and dedication provided to me over all these years, especially to my parents Caio and Clotilde, who have always supported me so much, even in the face of so many difficulties. I am immensely grateful to my brothers Caio, Cristiano, Cassius, and my sister Claudia, whom so often offered me their understanding no matter how far we were.

I have been extremely lucky to have had so many excellent advisors during my PhD course: Eleonora Vig, Adrien Gaidon, Naila Murray, and Antonio López, which without whom this PhD would never had been possible. While some people spend so much time looking for a single good advisor to guide their path, I am thankful I had encountered not one, but four of the very best researchers that could ever have guided me during those three years. I am deeply grateful for their help, insights, understanding, and support during and beyond the context of this PhD.

I offer my thanks to everyone whom were once in Xerox Research Centre Europe, and whom were or still are in NAVER LABS Europe, as the excellence of all colleagues, friends, and professionals I've met there certainly transcends any particular name such company could have.

I also thank my professor and friend Dr. Ednaldo Brigante Pizzolato for giving me the very first opportunity to conduct research while still in Brazil, and helping me walk the first steps that would eventually bring me here. I thank Victor Ciriza for then giving me the opportunity that would change my life forever.

I thank everyone from the best engineering team and the very first friends I met when I arrived in Grenoble: Luis Ulloa, Arturo Mondragon, Christelle Loïodice, Michael and Christine Niemaz, Frederic Roulland, and Victor Ciriza. Working with these fantastic people made me strive to continue giving my best during the years I worked as a Research Engineer, both professionally and personally, eventually leading me to start this PhD.

I am also grateful for having the opportunity to meet so many outstanding people in the Centre de Visió per Computador in the multiple times I've stayed in Barcelona. I thank everyone from the astounding staff, who would help and offer their support every time I needed; and the amazing friends, who were always there to both talk and listen, about everything and anything.

I thank God for placing all these people in my life.

Abstract

In this dissertation, we explore different ways to perform human action recognition in video clips. We focus on data efficiency, proposing new approaches that alleviate the need for laborious and time-consuming manual data annotation. In the first part of this dissertation, we start by analyzing previous state-of-the-art models, comparing their differences and similarities in order to pinpoint where their real strengths comes from. Leveraging this information, we then proceed to boost the classification accuracy of shallow models to levels that rival deep neural networks. We introduce hybrid video classification architectures based on carefully designed unsupervised representations of handcrafted spatiotemporal features classified by supervised deep networks. We show in our experiments that our hybrid model combine the best of both worlds: it is data efficient (trained on 150 to $10,000$ short clips) and yet improved significantly on the state of the art, including deep models trained on millions of manually labeled images and videos. In the second part of this research, we investigate the generation of synthetic training data for action recognition, as it has recently shown promising results for a variety of other computer vision tasks. We propose an interpretable parametric generative model of human action videos that relies on procedural generation and other computer graphics techniques of modern game engines. We generate a diverse, realistic, and physically plausible dataset of human action videos, called PHAV for "Procedural Human Action Videos". It contains a total of $39,982$ videos, with more than $1,000$ examples for each action of 35 categories. Our approach is not limited to existing motion capture sequences, and we procedurally define 14 synthetic actions. We then introduce deep multi-task representation learning architectures to mix synthetic and real videos, even if the action categories differ. Our experiments on the UCF-101 and HMDB-51 benchmarks suggest that combining our large set of synthetic videos with small real-world datasets can boost recognition performance, outperforming fine-tuning state-of-the-art unsupervised generative models of videos.

Keywords: *computer vision, action recognition, machine learning, applied mathematics*

Resumen

En esta disertación, exploramos diferentes formas de realizar reconocimiento de acciones humanas en vídeos. Nos enfocamos en la eficiencia de los datos, proponiendo nuevos enfoques que alivian la necesidad de anotarlos manualmente, tarea muy laboriosa y subjetiva, sujeta a errores. En la primera parte de esta disertación, comenzamos analizando modelos anteriores de vanguardia, comparando sus diferencias y similitudes con el fin de identificar de dónde vienen sus verdaderas fortalezas. Aprovechando esta información, procedemos a aumentar la precisión de la clasificación basada en modelos diseñados por un experto a niveles que rivalizan con las redes neuronales profundas. Presentamos arquitecturas híbridas de clasificación de vídeo basadas en representaciones espaciotemporales generales y no supervisadas, cuidadosamente diseñadas como características de entrada a redes neuronales profundas supervisadas. Los experimentos que presentamos muestran que nuestro modelo híbrido combina lo mejor de ambos mundos: es eficiente en datos (entrenado en 150 a $10,000$ vídeos cortos) y mejora significativamente en el estado del arte, incluyendo modelos profundos entrenados en millones de imágenes etiquetadas manualmente y videos. En la segunda parte de esta tesis, investigamos la generación de datos sintéticos de entrenamiento para el reconocimiento de acciones, ya que recientemente este paradigma ha mostrado resultados prometedores en muchas otras tareas de visión por computador. Basándonos en técnicas de gráficos por computador, proponemos un modelo paramétrico e interpretable para generar vídeos de acciones humanas. Los vídeos que generamos son diversos, realistas y físicamente plausibles; llamamos PHAV (de "Procedural Human Action Videos") al conjunto de vídeos. PHAV contiene un total de $39,982$ videos, con más de $1,000$ ejemplos para cada acción, contemplando 35 acciones diferentes. Nuestro enfoque no se limita a las secuencias de captura de movimiento existentes, ya que también definimos procedimentalmente 14 acciones sintéticas. Luego presentamos arquitecturas profundas para el aprendizaje de representaciones de tareas múltiples que mezclan vídeos sintéticos y reales, incluso si las categorías de acción son diferentes. Nuestros experimentos en los conjuntos de datos UCF-101 y HMDB-51 sugieren que la combinación de PHAV con pequeños conjuntos de datos del mundo real puede aumentar la precisión del reconocimiento, superando el estado del arte de los modelos no supervisados de generación de vídeos.

Palabras clave: *visión artificial, reconocimiento de acciones, aprendizaje automático, matemáticas aplicadas.*

Resum

En aquesta dissertació, explorem diferents maneres de reconèixer accions humanes en vídeo. Ens centrem sobretot en l'eficiència amb les dades, investigant i proposant nous mètodes que permetin evitar la laboriosa i lenta anotació de dades de forma manual. A la primera part d'aquesta dissertació, comencem analitzem els millors models preexistents, comparant les seves diferències i similituds amb la finalitat d'identificar d'on provenen els seus punts forts. Aprofitant aquesta informació, procedim a millorar el rendiment en classificació d'aquests models senzills a nivells que podrien competir amb xarxes neuronals profundes mitjançant la introducció d'arquitectures híbrides de classificació de vídeo. Aquestes arquitectures estan basades en representacions no supervisades dissenyades amb característiques espai-temporals degudament escollides a mà i després classificades per xarxes neuronals profundes supervisades. En els nostres experiments mostrem que el model híbrid que proposem combina el millor d'ambdós mons: per una banda és més eficient amb les dades (entrenat entre 150–10,000 fragments de vídeos curts); i per l'altra, millora significativament els resultats dels models existents, incloent models profunds entrenats en milions d'imatges i vídeos etiquetats manualment. A la segona part de la dissertació, investiguem la generació de dades d'entrenament sintètiques per al reconeixement d'accions, ja que recentment s'han mostrat resultats prometedors en una varietat d'altres tasques en visió per computador. Proposem un model generatiu paramètric interpretable de vídeos d'acció humana que es basa en la generació procedimental i altres tècniques de gràfics per computador existents en els motors dels videojocs moderns. Generem un conjunt sintètic de vídeos d'accions humanes diverses, realistes i físicament plausibles, anomenats PHAV ("Procedural Human Action Videos"). Aquest conjunt de dades conté un total de 39,982 vídeos, amb més de 1,000 exemples per cadascuna de les 35 categories d'acció. La nostra proposta no es limita a les seqüències de captura de moviment existents, i definim procedimental 14 accions sintètiques. Després, presentem arquitectures profundes d'aprenentatge de representacions multi-tasca per fusionar vídeos sintètics i reals, fins i tot quan les categories d'acció difereixen. Els nostres experiments en comparats amb els altres mitjançant els punts de referència UCF-101 i HMDB-51 suggereixen que la combinació del gran conjunt de vídeos sintètics que proposem amb petits conjunts de dades del món real pot millorar el rendiment, superant els models generatius de vídeo no supervisats recentment desenvolupats.

Paraules clau: *visió artificial, reconeixement d'accions, aprenentatge automàtic, matemàtiques aplicades.*

Contents

List of Figures

List of Tables

Acronyms

2SN	Two-Stream Networks
BN	Batch Normalization
BoW	Bag-of-Words
C3D	Convolutional 3D Network
CIFAR	Canadian Institute for Advanced Research
CNN	Convolutional Neural Network
DAFS	Data Augmentation by Feature Stacking
DN	Double Normalization
DNN	Deep Neural Network
DO	Drop-Out
DT	Dense Trajectories
EM	Expectation Maximization
FPS	Frames Per Second
FV	Fisher Vector
GMM	Gaussian Mixture Model
GRU	Gated Recurrent Unit
HMDB	Human Motion Data Base
HOF	Histogram of Optical Flow
HOG	Histogram of Oriented Gradients

HPTSN	Human Parsing Temporal Segment Network
iDT	Improved Dense Trajectories
I3D	Inflated 3D Convolutional Networks
i.i.d	Independent and Identically Distributed
ILSVRC	ImageNet Large-Scale Visual Recognition Challenge
ISCC-NBS	Inter-Society Color Council – National Bureau of Standards
JHMDB	Join-annotated Human Motion Data Base
KLT	Kanade–Lucas–Tomasi
LSTM	Long Short Term Memory
LRCN	Long-term Recurrent Convolutional Network
mAcc	mean Accuracy
mAP	mean Average Precision
MAP	Maximum A Posteriori
MBH	Motion Boundary Histograms
MBHx	Motion Boundary Histogram (along the x-axis)
MBHy	Motion Boundary Histogram (along the y-axis)
MIFS	Multi-skIp Feature Stacking
ML	Maximum Likelihood
OF	Optical Flow
PC	Parallel Coordinates
PCA	Principal Component Analysis
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SA	Soft-Assignment

SFV	Spatial Fisher Vector
SIFT	Scale-Invariant Feature Transform
STP	Spatio-Temporal Pyramids
SURF	Speeded-Up Robust Features
TDD	Trajectory-pooled Deep-convolutional Descriptors
TREC	Text REtrieval Conference
TRECVID	TREC Video Retrieval Evaluation
TSN	Temporal Segment Networks
TVHI	TV Human Interactions
TV-L1	Total Variation-L^1
SVM	Support Vector Machine
UCF	University of Central Florida
VGAN	Video Generative Adversarial Networks
VLAD	Vector of Linearly Agregated Descriptors
VOC	Visual Object Classes
VQ	Vector Quantization

1 Introduction

In order to carry a positive action we must develop here a positive vision.

Dalai Lama

UNDERSTANDING HUMAN BEHAVIOR IN VIDEOS remains a key problem in computer vision. Not only solving, but also *moving* towards solving this problem brings advances in diverse fields, from human computer interaction to security surveillance, passing through home care, autonomous driving, robotics, and even shopping. In a time where ubiquitous computing is becoming the norm, lowering the interaction barrier between humans and computers by having machines understand and possibly *respond* to our actions has not only the potential to revolutionize the way humans interact with technology, but to also change our expectations towards what computers should provide us – in the same way touch interfaces may one day become more natural than flipping pages of a book [79].

For a practical definition, human action recognition in videos can be described as a sub-task of video classification. Whereas in video classification the objective may be to classify a single video into one or more possible class labels, in human action recognition the goal is simpler: each video to be classified should contain at least one person, and this person should be performing at least one action (*e.g.,* riding a bycicle). There are, however, no limitations on the level of granularity expected from the action to be recognized. Actions are hierarchical in nature, and can often be described as combinations of simpler, atomic motions of specific body parts (*e.g.,* raising the left leg, then moving the leg forward) [53, 55, 56, 154].

Methods for video classification therefore need to capture information about both appearance and motion. Achieving accurate representations for both require either carefully handcrafting features with prior knowledge (*e.g.,* the dense trajectories approach of [202]) or end-to-end deep learning of high capacity models from a large amount of labeled data (*e.g.,* the two-stream architectures of [22, 170, 208]).

1.1 Context

Even though at the current date of writing the state of the art in action recognition has now shifted towards deep learning, for a long period of time while deep neural networks had already conquered the landscape in image recognition, the most succesfull approaches in action recognition and video classification in general were still based on handcrafted features using relatively simple shallow classifiers [43, 51, 101, 140, 201, 202, 203, 204, 205].

At the time, there was no clear indication on why deep neural networks for video were not performing as well as their counterparts for image classification. The dominance of methods based on handcrafted features and the constant confirmation that features learned by neural networks were in fact orthogonal to them [49, 91, 187, 207] did not help clarify whether a radically new approach was needed, or whether the problem could have been tackled with existing deep architectures by just gathering enough additional training data. It was evident, however, that the video datasets at the time contained much less samples than their image counterparts, and would need to be expanded before better conclusions could be drawn.

Still, the first remarkable effort in gathering a seemingly large enough dataset from which weights could be learned from scratch by deep learning (*e.g.,* [91]) revealed a complete new set of problems associated with gathering, maintaining and distributing such large amounts of videos. It was soon discovered that distributing videos through a video sharing platform (*e.g.,* YouTube) [73, 91, 92] would eventually lead to the loss of many of those videos due to copyright violations, privacy requests, censorship, regional filters, and simply from users pulling out videos from the platform. Besides, the problem of having to download, store and process terabytes of data – not to mention also be able to learn deep neural networks from them – also meant that many researchers would have been excluded from the field in case they did not have the right hardware and infra-structure at their disposal.

The aforementioned issues notwithstanding, the classification accuracy for the first networks trained on those large datasets could hardly surpass existing handcrafted methods of the time (in fact, it had been shown later, including by works that originated from the research in this thesis [43], that methods based on handcrafted features could perform significantly better than deep methods that had been pre-trained on millions of samples [91, 176, 187, 221]). This was a strong indication that there was something to be learned from handcrafted features even though deep learning approaches were more promising given their success in so many other areas. And at the same time, it also served as a reminder that having datasets with millions of samples could also not be enough for learning deep neural networks from scratch successfully – it might have been necessary to move further.

1.2 Objectives and scope

Given the issues raised in the previous section, the objective of this research is to explore and develop new approaches for creating state-of-the-art computer vision models for action recognition that can learn from a limited number of real training samples. The aim is to reduce the dependency on large manually annotated collections of data, helping the research in action recognition become more feasible for those with limited computing resources, or for those with limited personnel for collecting and annotating data for new action recognition tasks.

Given the success of handcrafted models and the observable complementarity between their features and deep-learned features, one of our objectives is to investigate whether it would be possible to keep the best of both worlds and still design an architecture that can be deep, data-efficient and state of-the-art at the same time. Yet, foreseeing that deep learning methods would eventually overtake the field, another objective was to determine whether it should have been possible to move further from the traditional workflow of having to download, store and handle large collections of data when training models for video – and to determine which novel architectures would have been necessary to make this move possible. Thus said, the main research questions pursued in this thesis therefore are:

- Would a hybrid architecture, unifying handcrafted features and deep neural networks perform better against pure deep learning when there is less data, even when those pure deep learning methods had been pre-trained on large collections of images or videos beforehand?

- Would it be possible to avoid having to collect and annotate – or even download and store – large collections of videos in order to train deep learning systems by instead crafting a system that could generate fully annotated synthetic human action videos on-the-fly?

- Would those synthetically generated videos be effective in helping learn deep neural networks since these models need much more data to be trained?

The above questions aside, at the same time we will be limiting the scope of this thesis to keep the research feasible. In particular, in this work we will be focusing exclusively on visual features, meaning audio features will not be addressed in the main course of the dissertation. However, the use of audio clues has been shown to be beneficial in multiple works [131, 132, 151, 213] and one should therefore expect that the results presented in this thesis could be enhanced when incorporating audio features (an example on how audio features could be combined with models presented in this thesis can be seen in appendix A).

1.3 Contributions

The main contribution of this thesis shall be the investigation of the different ways to achieve data-efficient models for human action recognition. Most specifically, in this work we contribute with:

- Methods for significantly boosting the accuracy performance of previous state-of-the-art models based on handcrafted features, which even with a simple SVM classifier are already on par with the state of the art. We provide experimental evidence that showing *sympathy for the details* (*e.g.,* spatiotemporal structure, normalization) and doing *data augmentation by feature stacking* (instead of duplicating training samples) are effective techniques for enhancing the performance of those methods. We demonstrate that the optimal design decisions we present are general, measuring their impact across multiple datasets with unique characteristics (Section 3.2).

- A *data efficient hybrid architecture* that combines unsupervised representation layers with a deep network of multiple fully connected layers. We show that *supervised mid-to-end learning of a dimensionality reduction layer together with non-linear classification layers* yields an excellent compromise between recognition accuracy, model complexity, and transferability of model parameters across datasets thanks to a reduced risk of training data overfitting and to modern optimization techniques (Section 3.3).

- A *parametric generative model of human action videos* relying on physics simulations, scene composition rules, and procedural animation techniques like *ragdoll physics* that provide a much stronger prior than just viewing videos as tensors or sequences of frames (*e.g.,* as in [198]). We show how to procedurally generate physically plausible variations of different types of action categories originally based on MOCAP datasets through animation blending, physics-based navigation, or entirely from scratch using programmatically defined behaviors. We demonstrate an approach to obtain naturalistic actor-centric randomized camera paths to film the generated actions, with care for physical interactions of the camera. Furthermore, our manually designed generative model has *interpretable parameters* that allow to either randomly sample or precisely control discrete and continuous scene (weather, lighting, environment, time of day, etc), actor, and action variations to generate large amounts of diverse, physically plausible human action videos (Section 4.3).

- A *quantitative experimental validation* for our generative model using a modern and accessible game engine (Unity®Pro), demonstrating it can be used to synthesize a labeled dataset of 39,982 videos, with more than 1,000 examples for each of 35 action categories: 21 grounded in MOCAP data, and 14 entirely synthetic ones defined procedurally (Sections 4.4), which we then proceed to use to improve the state of the art in action recognition (Chapter 5).

- A *new dataset for human action recognition* originating from our quantitative experimental validation, called *PHAV* for "Procedural Human Action Videos". Our dataset is publicly available for download, and provides five ground-truth modalities which can be explored to train and improve models not only for action recognition but also semantic segmentation, optical flow, and depth estimation (Section 4.4).

- Two *new deep learning architectures* that are able to learn from both synthetic and real-world training samples at the same time, even if their action categories differ (Section 5.2), and even if certain ground-truth modalities are available only for the synthetic domain (Section 5.3).

1.4 Outline

This PhD thesis is organized in 6 chapters and one appendix. After this introduction, in Chapter 2 we present works in human action recognition as well as other literature related to the topics addressed in this thesis. In Chapter 3, we analyze the different models presented in the related works to build a strong baseline model. While this model could already rival other state-of-the-art models due to a careful application of good practices and engineering during its design, we go further and present a novel model for action recognition based on a hybrid shallow-deep architecture which could attain state-of-the-art performance in multiple datasets using less data than pure deep-learning approaches. In Chapter 4 we begin to explore a different way to achieve data-efficiency, now based on data generation. We demonstrate how it is possible to leverage a game engine and computer graphics to create a human action video generator that can be used to create a virtually infinite amount of strongly supervised data for training deep neural networks for action recognition and other tasks. In Chapter 5 we then explore different approaches for using this synthetic data to finally train those networks, presenting novel deep architectures that can learn from synthetic and real-world data at the same time while leveraging the different modalities offered by our virtual dataset. Finally, in Chapter 6 we present our conclusions along with a summary of scientific contributions, articles and discussion for future works that can originate from this thesis.

2 Human action recognition in videos

> The improvement of understanding is for two ends: first, our own increase of knowledge; secondly, to enable us to deliver that knowledge to others.
>
> John Locke

In this chapter we present relevant works in the literature that are related to this thesis. After a brief introduction, we present an overview of related datasets for action recognition. Afterwards, we present an aggregated overview of various methods according to whether they are based on handcrafted features, deep features, shallow classifiers, and deep classifiers. We then identify the most prominent works in the handcrafted and deep-learning camps, performing a detailed analysis and comparison of their characteristics and design choices. This analysis will later be used to motivate and derive our strong baseline for action recognition in Section 3.2, our new hybrid architecture in Section 3.3 and in the design of our multi-task architectures in Sections 5.2 and 5.3.

2.1 Introduction

ONE OF THE EARLIEST WORKS in action recognition is [188], in which the authors devised a dynamic scene model aiming to understand a line-drawn, simple cartoon movie[1]. While being severely limited by the technology existent at their time, their system could correctly identify the main humanoid actor in the movie, find elements of the surrounding environment, identify motions, and infer the actions being performed. Part of their approach was based on the extraction of image feature

[1] Interestingly enough, this work was also one of the firsts showing the use of synthetic data for exploring action recognition. We will be revisiting the topic of synthetic data later in this chapter in Section 2.4 and also in Chapters 4 and 5.

points from consecutive video frames, an approach that would continue being prevalent for a long time in the literature. However, their recognition system was based on fixed ad-hoc rules, with no machine or statistical learning involved.

Current methods for action recognition are largely based on the automatic learning of parametrized models from collections of videos using machine learning techniques (*cf.* [199] for a recent survey). These collections, or datasets for action recognition, can be represented as a set of n tuples $\mathbf{X} = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{n}$, where $\boldsymbol{x_i}$ usually denotes a video volume in $\mathbb{R}^3$, and $y_i \in \Omega$ its associated action label, where Ω denotes the space of all possible labels a single video can have. The goal of an action recognition system is then to estimate a classification function on the form $F(\boldsymbol{x}) : \mathbb{R}^3 \to \Omega$ using the dataset $\mathbf{X}$.

In the next sections we will see datasets that can be used for this purpose and explore different action recognition systems that can be trained in this way.

2.1.1 Datasets for action and related tasks

The number of action recognition datasets has been growing steadily in recent years, with at least one new dataset being introduced every year since 2005 (*cf.* recent surveys in [5, 26, 71]).

These datasets can vary widely in a number of characteristics. Not only do they differ in the number of action categories and videos they contain (*cf.* Table 2.1), but also in the average length of their clips and their resolution (*cf.* Table 2.2), whether videos are trimmed (*i.e.* they are short clips containing only the action of interest) or untrimmed (*i.e.* actions can happen anywhere in a long video not necessarily focused in the action), whether they are multi-class (*i.e.* a video can belong only to a single class) or multi-label (*i.e.* a clip can belong to multiple classes at the same time), and in the different data modalities and ground-truth annotations they provide. Each dataset is normally also associated with a particular evaluation protocol that needs to be followed when reporting results in the literature to ensure performance numbers are always comparable across different works.

Below we show a list of the datasets used throughout this dissertation, along with their main characteristics and their associated evaluation protocols:

- The **UCF-101** [173] dataset contains 13,320 video clips distributed over 101 distinct classes. This is the same dataset used in the THUMOS'13 challenge [88]. The performance in this dataset is measured as the average accuracy (mAcc) over three fixed train/test splits.

- The **HMDB-51** [99] dataset contains 6,766 videos distributed over 51 distinct action categories. Each class in this dataset contains at least 100 videos, with high intra-class variability. As is the case for UCF-101, the evaluation protocol to be followed is again the mAcc over three fixed train/test splits [99] provided by the dataset authors.

- The **JHMDB** [84] dataset is a subset of the HMDB-51 dataset in which human joints have been manually annotated using a puppet model. This dataset contains 928 video clips from HMDB-51 divided into 21 classes. The evaluation protocol is the same as HMDB-51, using the same data splits.

- The **Hollywood2** [114] dataset contains 1,707 videos extracted from 69 Hollywood movies, distributed over 12 overlapping action classes. This is a multi-label dataset where one video can have multiple class labels (*e.g., HandShake* and *HugPerson*). Results are reported using the mean average precision (mAP) using a fixed train-test split also provided by the dataset authors.

- The **Olympic Sports** [129] dataset contains 783 videos of athletes performing 16 different sport actions, with 50 sequences per class. Some actions include interactions with objects, such as *Throwing*, *Bowling*, and *Weightlifting*. The most common evaluation metric for this dataset is the mAP over the train/test splits released together with this dataset. However, some works have also reported the mAcc instead of the mAP in the literature (*e.g.,* [18, 186]) and it may be necessary to report both metrics in order to keep results always comparable (*e.g.,* [56, 110]).

- The **High-Five** (a.k.a. TV Human Interactions, or TVHI) [138] dataset contains 300 videos from 23 different TV shows distributed over four different human interactions and a negative (no-interaction) class. The most common evaluation metric for this dataset [56, 137, 138, 203] is the mAP for the positive classes (mAP+) through 2-fold cross-validation using the train/test splits provided by the dataset authors.

- The **ActivityNet** [73] dataset (release 1.3) contains 19,994 untrimmed videos distributed over 200 activity classes. It contains at least 100 videos per action category, with an average of 1.54 activity instances per video. Since the testing classes are withheld for the challenge purposes, the common evaluation protocol to compare against other works in the literature is the mAP for the validation set of this dataset. The videos contained in this dataset are untrimmed, presenting the longest clip durations among all datasets considered in this thesis.

A summary of the main characteristics of the above datasets is shown in Tables 2.1, 2.2, and 2.2. A comparison of the average length of the video clips contained in each dataset is shown in Figure 2.1, and example frames are shown in Figure 2.2.

Table 2.1 – Statistics for action recognition datasets according to their organization.

		Number of videos (with aggregate statistics for a single split)							
			Training set			Validation set			Test set
Dataset	Classes	Total	Total	Per class (s.d.)	Range	Total	Per class (s.d.)	Range	Total
UCF-101	101	13,320	9,537	94.42 (13.38)	72-121	3,783	37.45 (5.71)	28-49	-
HMDB-51	51	6,766	3,570	70.00 (0.00)	70-70	1,530	30.00 (0.00)	30-30	-
JHMDB	21	929	660	31.42 (4.71)	25-39	268	12.76 (1.74)	11-16	-
Hollywood2	12	1,707	901	75.08 (36.74)	24-135	972	81.00 (36.77)	33-146	-
High-Five	5 (4+1)*	300	150	30.00 (10.00)	25-50	150	30.00 (10.00)	25-50	-
Olympic Sports	16	783	649	40.56 (11.02)	17-56	134	8.37 (2.12)	4-11	-
ActivityNet (1.3)	200	19,746†	9,902	49.51 (10.16)	28-87	4,856	24.28 (5.75)	9-42	4,988

Averages are per class considering only the first split of each dataset (when applicable). *High-Five contains four action classes and one non-action class. †Videos that could be downloaded from southeastern France in April 2016.

Table 2.2 – Statistics for action recognition datasets according to their contents.

	Width		Height		Frames per second	
Dataset	Mean (s.d.)	Range	Mean (s.d.)	Range	Mean (s.d.)	Range
UCF-101	240.99 (0.24)	320-400	320.02 (1.38)	226-240	25.90 (1.94)	25.00-29.97
HMDB-51	366.81 (77.61)	176-592	240.00 (0.00)	240-240	30.00 (0.00)	30.00-30.00
JHMDB	320.00 (0.00)	320-320	240.00 (0.00)	240-240	30.00 (0.00)	30.00-30.00
Hollywood2	609.24 (65.30)	480-720	338.31 (77.89)	224-576	24.75 (1.10)	23.98-29.97
High-Five	607.36 (47.75)	400-720	356.37 (29.72)	288-576	24.10 (0.33)	23.98-25.00
Olympic Sports	509.38 (177.07)	192-1280	361.70 (99.35)	144-720	-	-
ActivityNet (1.3)	845.28 (405.76)	128-1,280	516.44 (197.66)	96-720	27.68 (4.04)	6.00-30.00

Averages are among all videos in the dataset (and not per-class as in Table 2.1).

Table 2.3 – Number of frames for popular action recognition and image datasets.

	Number of frames			
Dataset	Total	Mean (s.d.)	Range	Year
UCF-101	2,484,199	186.50 (97.76)	29-1,776	2012 [173]
HMDB-51	639,307	94.488 (68.10)	19-1,063	2011 [99]
JHMDB	38,152	41.112 (8.84)	18-48	2013 [84]
Hollywood2	487,556	285.62 (273.53)	59-3,116	2009 [114]
High-Five	28,283	94.276 (64.04)	23-650	2010 [138]
Olympic Sports	163,902	180.18 (152.94)	24-1,184	2010 [129]
ActivityNet (1.3)	63,840,021	3,236.17 (1,940.30)	28-29,222	2015 [73]
ImageNet	14,197,122	-	-	2009 [87]
PASCAL-VOC	10,103	-	-	2010 [48]
PASCAL-VOC (humans)	3,589	-	-	2010 [48]

Averages are among all videos in each dataset. Datasets for image-related tasks (image classification and semantic segmentation) are shown in the bottom for comparison purposes.

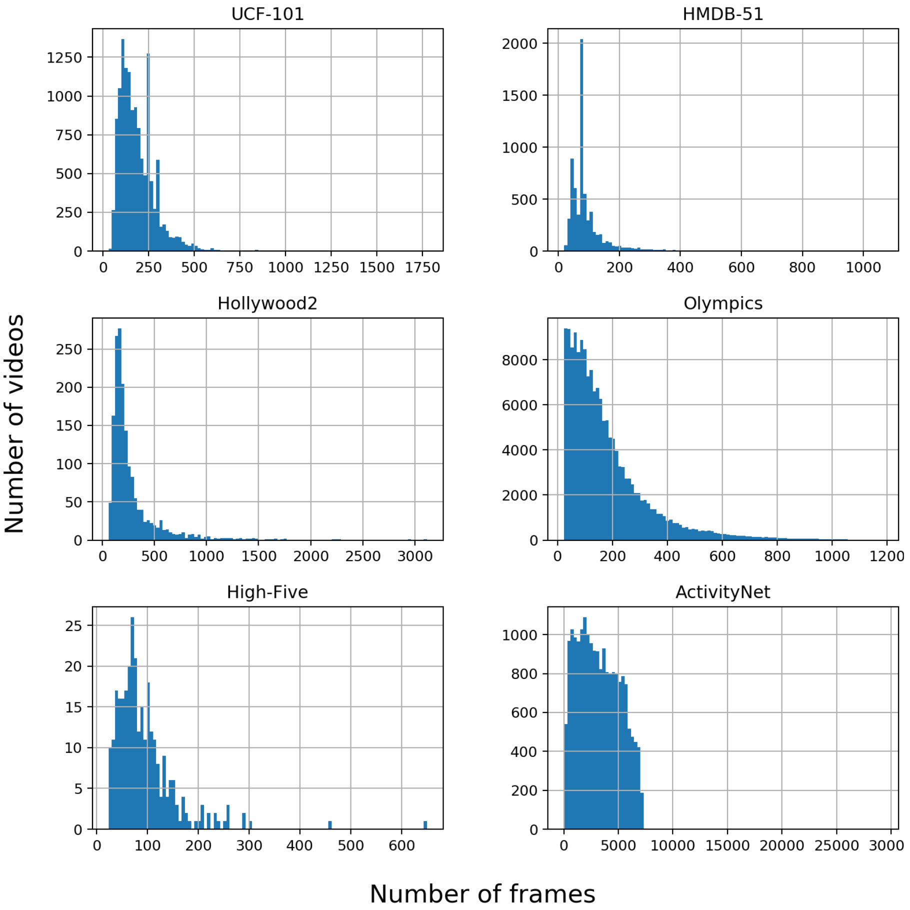


Figure 2.1 – Histograms showing the frame count distribution for common action recognition datasets. ActivityNet contains videos with the largest number of frames, but also the largest variance (*cf.* Table 2.3 for the actual numbers).

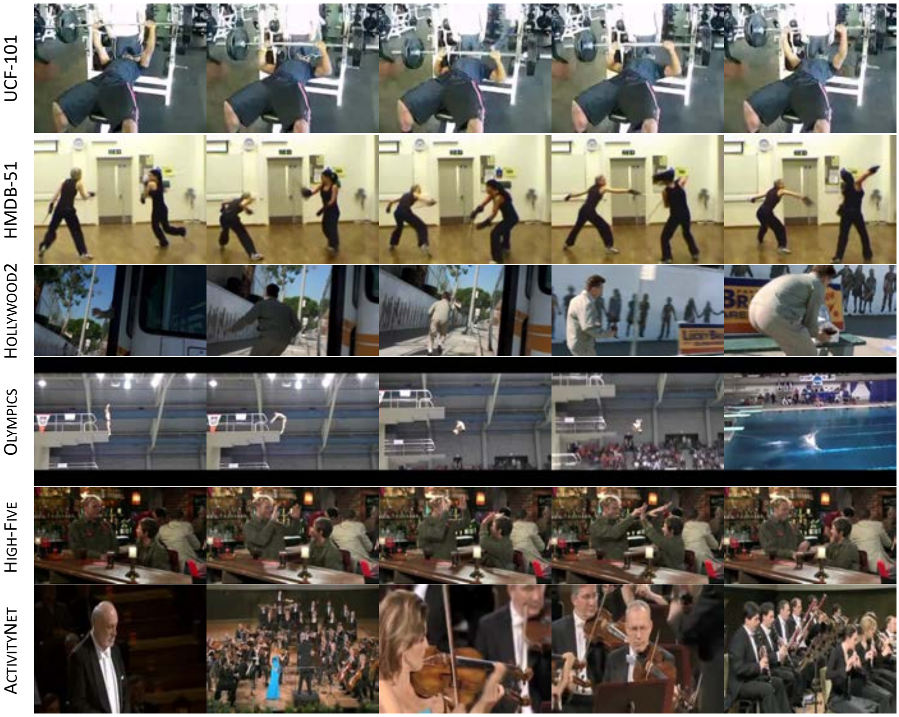


Figure 2.2 – Example frames from multiple action recognition datasets. The actions for each dataset from top to bottom are: *Bench press, Fencing, Run, Platform diving, High-five, Playing violin.* As Hollywood2 and ActivityNet contain the longest videos (*cf.* Table 2.3), they also contain the most complex action clips involving multiple actions per clip, multiple camera angles, and multiple camera shots.

Other noteworthy datasets include:

- The **Sports-1M** [91] dataset is a large-scale, multi-label dataset containing 1.13 million videos divided among 487 sport categories. It contains from 1000 to 3000 videos per class where approximately 5% of the videos contain more than one action class. This dataset was among the first large datasets to be exclusively distributed using YouTube links, and therefore one of the first to suffer from the problem of missing videos – where links would become unavailable over time due to region restrictions, copyright violations, user deletions, among other reasons. The dataset would take over 1.7TB on disk.

- The **YouTube-8M** [1] dataset contains about 7 million video URLs and 450,000 hours of video, divided among 4716 classes with an average of 3.4 labels per video. However, the classes in this dataset are not restricted to action classes, being therefore a general dataset for video classification. This dataset also comes with 3.2 billion pre-computed audio-visual features that have been extracted using deep neural networks, using Inception-v3 [184] for the image features, and VGG-acoustic [74] for the audio features.

- The **Kinetics** [92] dataset is the most recent dataset introduced at the time of writing of this thesis. This dataset contains more than 240k training videos divided among 400 action classes, resulting in a dataset with at least 400 clips for each class. As the two other datasets above, this dataset is also available to download through a collection of YouTube links.

In this thesis we will refer to three additional datasets which are not particular to action recognition, but which will be used later to provide external training data for the models we describe in section 5. These are:

- The **ImageNet** [87] dataset has been the de-facto dataset for training models for image classification in the past decade. This dataset is comprised of 456,567 training images, with at least 20,121 validation images and 40,152 test images (depending on the year of the challenge) divided among at least 1,000 object categories. This dataset has been used to train image classification models whose weights have been used to bootstrap multiple action classification models such as [170, 207, 208].

- The **Pascal-VOC** [48] dataset was part of the PASCAL Visual Object Classes (VOC) Challenge 2010. This dataset contains 10,103 images divided among 20 object classes, containing 23,374 ROI annotated objects and 4,203 segmentations. This dataset can be used to train models for semantic segmentation as it contains the pixel boundaries surrounding each object within its images (*cf.* Figure 2.3).

- The **PASCAL-Part** [32] dataset is a set of additional part annotations for the PASCAL VOC 2010 challenge [48]. This dataset contains 19,740 images with 193 extra part annotations divided among the 20 object classes of the original challenge, including human body parts (*cf.* Figure 2.4).

A comparison between these datasets and the aforementioned action recognition datasets is shown in Table 2.3.

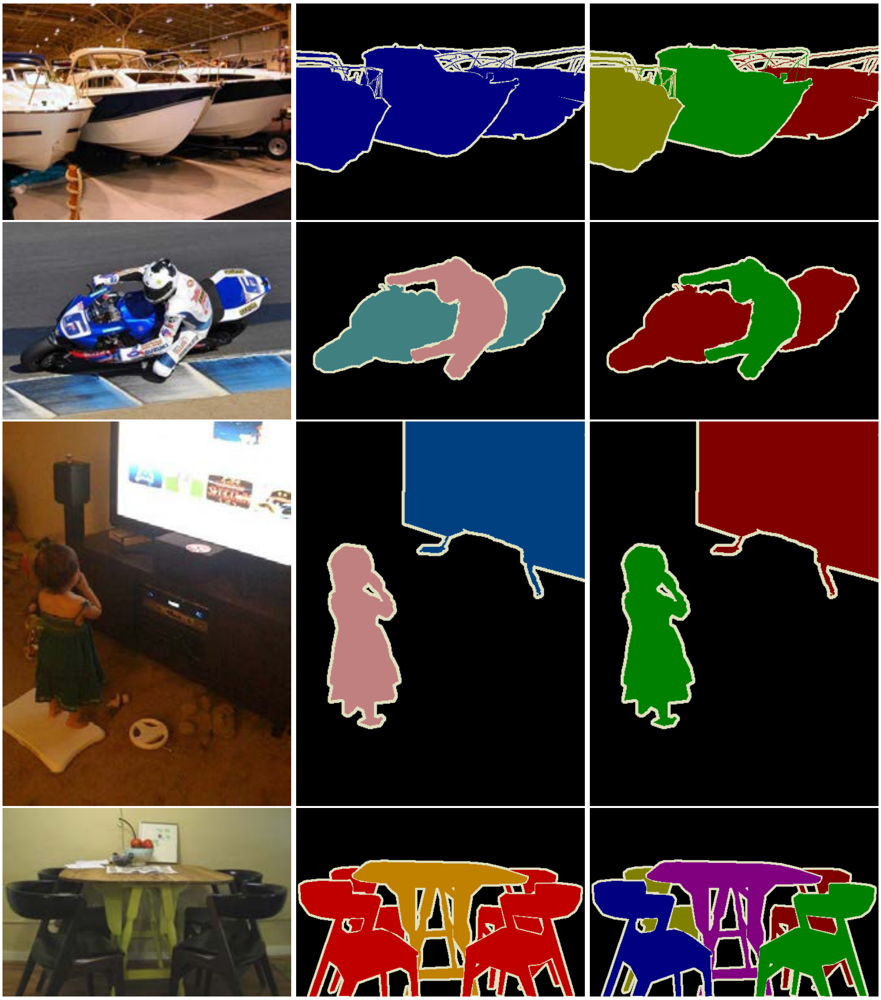


Figure 2.3 – Example images from the PASCAL VOC 2010 dataset and their pixel-level annotations. Left: original image. Middle: class-level pixel-wise annotations. Right: instance-level pixel-wise annotations.

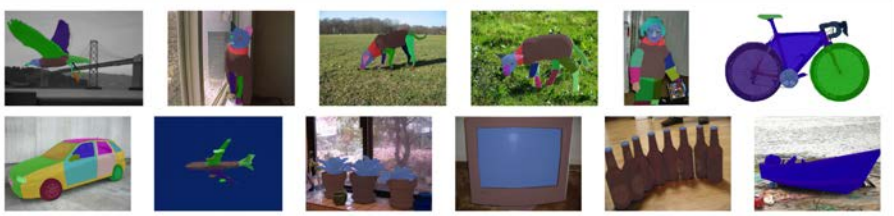


Figure 2.4 – Example images from the PASCAL VOC 2010 dataset with part annotations from PASCAL-Part.

2.1.2 Overview of related works in the literature

In this section we will present an overview of related works that adopt a machine learning-based approach for action recognition, using the datasets mentioned in the previous section for model training and evaluation. We organize these works into four broad categories based on whether they involve *handcrafted vs. deep-based* video features, and a *shallow vs. deep* classifier. An overview of related works according to this organization is shown in Table 2.4.

Table 2.4 – Categorization of related recent action recognition methods.

	Classifier	
Features	Shallow	Deep
Handcrafted	[51, 56, 75, 101, 140, 201, 202, 203, 204]	[10], Chapter 3
Deep-based	[187, 207, 214, 221]	[11, 22, 45, 49, 59, 85, 91, 127, 170, 176, 182, 208, 211, 213], Chapter 5

Handcrafted features, shallow classifier. A significant part of the progress on action recognition has been driven by the development of local handcrafted spatiotemporal features encoded as bag-of-words representations classified by "shallow" classifiers such as SVMs [51, 56, 75, 101, 140, 201, 202, 203, 204]. Most successful approaches use *improved Dense Trajectories (iDT)* [204] to aggregate local appearance and motion descriptors into a video-level representation through the Fisher Vector (FV) encoding [147, 150]. To create an iDT representation, local descriptors such as

Histogram of Oriented Gradients (HOG) [38], Histograms of Optical Flow (HOF) [39], and Motion Boundary Histograms (MBH) [201] are extracted along dense point trajectories obtained from optical flow fields. There are several recent improvements to iDT, for instance, using motion compensation [54, 56, 80, 203] and stacking of FVs to obtain a multi-layer encoding similar to mid-level representations [141]. To include global spatiotemporal location information, Wang *et al.* [203] compute FVs on a Spatio-Temporal Pyramid (STP) [104] and use Spatial Fisher Vectors (SFV) [96]. Fernando *et al.* [51] model the global temporal evolution over the entire video using ranking machines learned on time-varying average FVs. Another recent improvement is the Multi-skIp Feature Stacking (MIFS) technique [101], which stacks features extracted at multiple frame-skips for better invariance to speed variations. An extensive study of the different steps of this general iDT pipeline and various feature fusion methods is provided in [140].

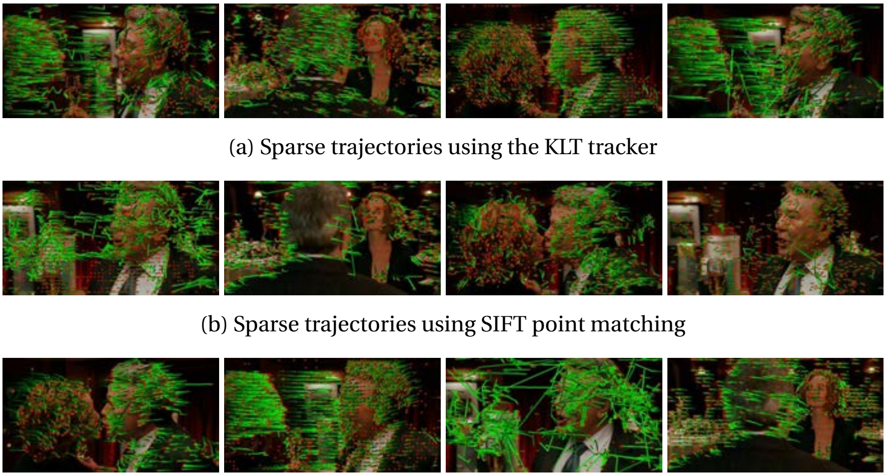
End-to-end learning: deep-based features, deep classifier. The seminal supervised deep learning approach of Krizhevsky *et al.* [98] has enabled impressive performance improvements on large scale image classification benchmarks, such as ImageNet [160], using Convolutional Neural Networks (CNN) [107]. Consequently, several approaches explored deep architectures for action recognition. While earlier works in this direction resorted to unsupervised learning of 3D spatiotemporal features [106], supervised end-to-end learning has recently gained popularity [11, 22, 45, 49, 59, 85, 91, 127, 170, 176, 182, 208, 211, 213]. Karpathy *et al.* [91] studied several architectures and fusion schemes to extend 2D CNNs to the time domain. Although trained on the very large Sports-1M dataset, their 3D networks performed only marginally better than single-frame models. To overcome the difficulty of learning spatiotemporal features jointly, Simonyan *et al.* 's Two-Stream Networks (2SN) architecture [170] is composed of two CNNs trained independently, one for appearance modeling on RGB input, and another for temporal modeling on stacked optical flow. This architecture was later used in Wang *et al.* 's Temporal Segment Networks (TSN) [208], and then merged with 3D convolutional networks in Carreira *et al.* 's Inflated 3D (I3D) Networks [22]. Sun *et al.* [182] factorize 3D CNNs into learning 2D spatial kernels, followed by 1D temporal ones. Alternatively, other recent works use recurrent neural networks (RNN) in conjunction with CNNs to encode the temporal evolution of actions [11, 45, 127]. Donahue *et al.* 's LRCN [45] processes video frames through the 2SN whose outputs are fed into a stack of LSTMs. Ng *et al.* [127] proposes different temporal feature-pooling architectures to combine information over longer time periods. Most recently, Ballas *et al.* [11] makes use of convolutional features from different layers of a pre-trained net as input to a GRU-RNN. In a different line of work, [176] also investigated the unsupervised learning of video representations using LSTMs. Overall, due to the difficulty of training 3D-CNNs and the need for vast amounts of training videos (*e.g.,* Sports-1M [91]), end-to-end methods used to report only marginal improvements over traditional baselines. Our experiments show that the iDT-FV often outperformed these approaches. It was only very recently that end-to-end methods had been shown to offer better performance than other methods, thanks to the availability of even larger amounts of training videos (*e.g.,* Kinetics [92]). Carreira *et al.* have shown that pre-training the I3D [22] architecture on the very large-scale Kinetics dataset (and therefore on hundred thousands of extra real manually annotated training videos) could bring large improvements over the state of the art. However, this architecture presented either comparable (*e.g.,* UCF-101) or worse (*e.g.,* HMDB-51) performance to the iDT-FV when trained directly on target datasets.

Deep-based features, shallow classifier. Several works [187, 207, 214, 221] explore the encoding of general-purpose deep-learned features in combination with "shallow" classifiers, transferring ideas from the iDT-FV algorithm. Zha *et al.* [221] combine CNN features trained on ImageNet [160] with iDT features through a Kernel SVM. The TDD approach [207] extracts per-frame convolutional feature maps from two-stream CNN [170] and pools these over spatiotemporal cubes along extracted trajectories. Similar to [91], C3D [187] learns general-purpose features using a 3D-CNN, but the final action classifier is a linear SVM. Like end-to-end deep models, these methods rely on large datasets to learn generic useful features, which in practice perform on par with or worse than iDT.

Hybrid architectures: handcrafted features, deep classifier. There is little work on using unsupervised encodings of handcrafted local features in combination with a deep classifier. In early work, Baccouche *et al.* [10] learn temporal dynamics of traditional per-frame SIFT-BOW features using a RNN. The method, coupled with camera motion features, improves on BoW-SVM for a small set of soccer videos. In this thesis, we propose a new hybrid method and discuss its advantages and disadvantages in Chapter 3.

2.2 Local feature encoding approaches

Methods based on the encoding and subsequent pooling of local features were among the best performing methods for action recognition for several years [140, 141, 201, 202, 203, 204, 207], in part thanks to the steady proposal of techniques that could be used to improve it [27, 28, 101, 140, 207]. The Dense Trajectories (DT) approach of Wang *et al.* (*cf.* Figure 2.5) has been the most prominent example



(a) Sparse trajectories using the KLT tracker

(b) Sparse trajectories using SIFT point matching

(c) Dense trajectories from optical flow

Figure 2.5 – Comparison between different methods to extract pixel trajectories from video, showing the dense trajectories approach of Wang *et al.* at the bottom. Image adapted from [204].

of these methods, figuring multiple times among the state of the art, either as a standalone approach [101, 140, 141, 201, 202, 203, 204, 207], or in combination with other methods [49, 91, 187], often as a requirement for claiming state-of-the-art performance.

Given the sheer amount of related works specific to the dense trajectories and local feature encoding approaches, in the next sections we will create a common notation framework to refer to the most prominent works in this field and then use it to analyze these works in detail.

2.2.1 Dense trajectories and the local feature encoding pipeline

Most works based on dense trajectories follow a common pipeline that can be followed to map a source input video to its most likely class probabilities. The general pipeline for action recognition based on local features could be composed of the following stages: video preprocessing, local feature extraction, local feature encoding, representation extraction, and classification. In turn, each of these stages is composed of multiple steps which can be chosen and repeated at will before arriving at the next stage.

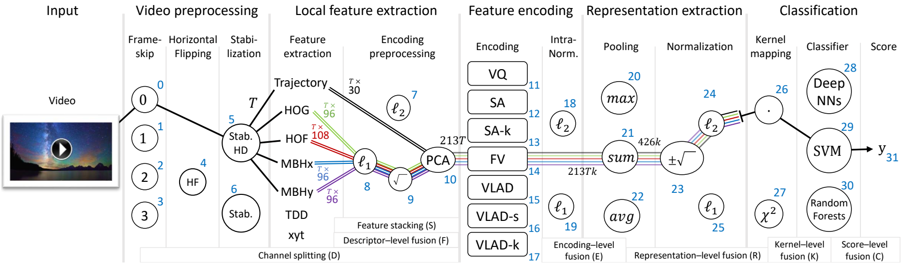


Figure 2.6 – Schematic representation of a local features pipeline for action recognition. Different stages of the pipeline are separated by the tallest vertical lines, with stage titles shown at the top. Different choices within each stage are represented by numbered blocks, and opportunity windows for channel division (D), feature stacking (S), and channel fusion (F, E, R, K, C) are shown in the bottom. The notation and numbering system are explained in Section 2.2.1.

We show a schematic representation of how a local feature encoding pipeline for dense trajectories can be organized in Figure 2.6. Possible steps within each stage are represented by the numbered blocks. In the next paragraphs we present an explanation for each of these blocks along with a discussion of their importance. We refer to each specific block in the pipeline using its number or name within parentheses.

Video preprocessing. The first stage of the pipeline receives the raw input video and prepares it for the next stages. The end result of this stage should still be a sequence of frames. Several video processing techniques can be applied at this stage, including frame-skipping [101], video mirroring [51, 75], and motion stabilization [51, 75, 101, 203, 204]. Depending on the skipping level, either all frames are kept (0), one frame is kept every two frames (1), one frame is kept three frames (2) or one frame is kept every four frames (3). After frame skipping, other potential operations includes horizontal mirroring (4), motion stabilization using a human detector to isolate the background from the human actor (5), or simpler motion stabilization using the entire scene (6).

Local feature extraction. The goal for the second stage of the pipeline is to take the preprocessed video from the previous stage and apply a local feature extractor to extract a variable number of local features from it. While it is possible to extract either dense or sparse features at this stage, dense sampling has been shown to

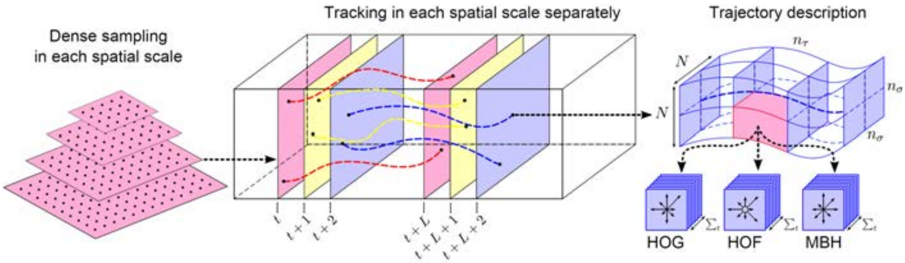


Figure 2.7 – The dense trajectories approach consists in registering pixel trajectories in optical flow fields, then extracting descriptors from the volume that forms around those trajectories. First, points are uniformly selected in dense grids from multiple spatial scales (left). Then, points are tracked in each spatial scale separately, reconstructing their trajectory (middle). Finally, local descriptors are extracted from regions of the video around each of these trajectories. Image adapted from [202].

be a superior alternative to sparse key-point tracking [103] in many benchmarks [140, 201, 202, 203, 204, 206]. For this reason, we maintain our focus in the dense features that can be extracted within the dense trajectories approach.

The dense trajectories approach consist in registering the trajectories of individual pixels inside optical flow fields from each video, and then extracting local descriptors from the volume that forms around these trajectories (*cf.* Figure 2.7). Examples of these descriptors include the sequence of pixel displacements within the pixel trajectory (Traj), Histograms of Gradients (HOG), Histograms of Optical Flow (HOF), Motion Boundary Histograms (MBH), the horizontal and vertical components of MBH (MBHx and MBHy), Trajectory-pooled Dense Descriptors (TDD) and the spatiotemporal location of the pixel that originated the trajectory (xyt). Works in the literature differ on which local descriptors should be included or excluded from the pipeline. For instance, [140, 203] have dropped the trajectory shape descriptors from their pipelines, unlike [101, 201, 202, 204] who maintain it.

Apart from being included, these descriptors can also be normalized using multiple techniques. One of the most well-known is RootSIFT [3], which is comprised by ℓ_1 normalization (8) followed by square-rooting (9). Although this technique had been initially proposed to postprocess SIFT histograms, it has been shown to be beneficial for other types of vector-based histogram descriptors. However, there is again no consensus on whether this technique should always be applied. For instance, in [140] RootSIFT is not applied at all, while in [101] it is applied even to the Traj descriptor, even though this descriptor is not a histogram.

Besides being normalized, descriptors can also be arbitrarily transformed. One example is Principal Component Analysis (PCA, 10), which can be used to reduce the number of dimensions in the descriptors [140, 150, 201, 202, 203, 204, 206] and decorrelate their dimensions. Ensuring that encodings have covariance matrices which are close to diagonal or unitary (*e.g.*, through whitening [150]) can help those vectors satisfy certain assumptions used during vocabulary creation in the next pipeline stage.

Local feature encoding. The third stage of the pipeline takes the local features extracted and post-processed in the previous stage and encodes then into intermediary feature-level representations. Local features can be encoded using a number of techniques. Examples include Vector Quantization (VQ, 11); Soft Assignment (SA, 12), Local Soft Assignment (SA-k, 13); Fisher Vector (FV, 14); Spatial Fisher Vector (SFV, 14); VLAD with hard-assignments (VLAD, 15); VLAD with soft assignments (VLAD-s, 16) and VLAD with local soft assignments (VLAD-k, 17). For an in-depth review of those techniques, see [140].

All aforementioned encoding techniques are based on the same common principle: The creation of a common *vocabulary* of local features, that represents the most common features typically present in the input videos; followed by the subsequent encoding of all local features based on how far the extracted local features are from elements of this vocabulary. It is easier to understand this mechanism by considering the simplest of these approaches: VQ.

The simplest approach for VQ consists in using the k-Means [70] algorithm[2] to cluster a subset of all local features (*e.g.*, all HOG descriptors) that can be extracted from all videos in a dataset. Taking only HOG descriptors as an example, this clustering will serve as the HOG vocabulary since each of its k centroids serves as a prototype for the most common HOG descriptors captured in the subsample (and which may therefore reflect the most common descriptors in the entire dataset). After vocabulary creation, it then becomes possible to encode each HOG descriptor $\mathbf{h} \in \mathbf{H}$ where $\mathbf{H}$ is the set of all HOG descriptors extracted from video $\mathbf{V}$ by replacing it with a one-hot vector $\phi(\mathbf{h}) = \delta\{i = \text{argmax}_{j \in \{1 \ldots k\}} \|\mathbf{h} - \mathbf{c_j}\|^2\}$ where $\mathbf{c_j}$ is the centroid vector for the j-th cluster found in k-means.

Even though VQ may be the simplest approach for feature encoding, best results are almost always achieved using FV-based encodings [125, 140, 141, 203, 207], obtained from Gaussian Mixture Model (GMM) clusterings, especially when incorporating spatiotemporal information [96, 101, 162]. There are three main ways of exploiting location information: Spatio-Temporal Augmentation (STA) as used in [101] (referred therein as Space-time Extended Descriptor, STED), Spatio-Temporal

[2] In practice, applying the plain k-Means algorithm to even a subsample of the trajectories extracted from all videos (256,000 being a common number) can be quite computational intensive. The reader may refer to [165] for a more efficient version of the k-means algorithm which may be more adequate to process the large amount of features generated by iDT.

Pyramids (STP) [105], and Spatial Fisher Vector (SFV) [96] (14). In the case of GMMs constrained to have diagonal variances, it is possible to see that the Spatial-GMM models needed for SFVs have exactly the same form as GMMs computed on STA descriptors. Since forcing diagonal variances implies an independence assumption between all dimensions, a global GMM using concatenated normalized coordinates can be seen as the Gaussian spatial model with MoG for appearance in [96]. However, one key difference between SFV and STA is vocabulary reuse. In the SFV approach, vocabularies do not have to be to be recomputed from scratch to include spatiotemporal information. The authors discovered that its possible to use a fixed spatial vocabulary shared across all visual words, such as centered at the center coordinates of the video [96], without loss of performance. We explore the effects of different approaches for incorporating spatiotemporal information in Section 3.2.

Representation extraction. The forth stage in the pipeline takes the variable number of local encodings produced in the previous stage and combine them into a single, fixed-length vector that represents the entire video. One way to combine these local encodings is by pooling. Techniques for pooling include max-pooling (20), sum-pooling (21), and average pooling (22). After the pooling, it is still possible to apply further normalizations such as power normalization (a.k.a signed square root, 23), ℓ_2-normalization (24) and ℓ_1-normalization (25).

Pooling can also be done across some common characteristics between the pooled features, *e.g.*, the spatiotemporal region (as in STP), or the image scale (as done in [207]). A series of normalizations can also be applied before pooling, such as intra-normalization using ℓ_2 (18) or ℓ_1 norms (19).

Normalizations performed during this stage can be used to address the *burstiness effect* [82]: the increased likelihood that similar features will occur more frequently within a same image or image patch. Many representations assume features within an image or video are independent and identically distributed (i.i.d). However, this is a rather unrealistic assumption [36, 82], since certain features are more likely to be present within an image given others also are (*e.g., window* features given that *door* features are also present). The burstiness effect further exacerbates the violation of this assumption, as similar features often repeat within images (*cf.* Figure 2.8). The presence of many repeated but similar features within a region may cause less prominent but still existent features to be eclipsed in the final representation.

The power normalization followed by ℓ_2 normalization has been shown to be an efficient way to dwindle this effect [82, 149]. On the other hand, intra-normalization [4] has been claimed to be able to completely suppress it. There are advantages and disadvantages in both normalizations: While the power normalization might not

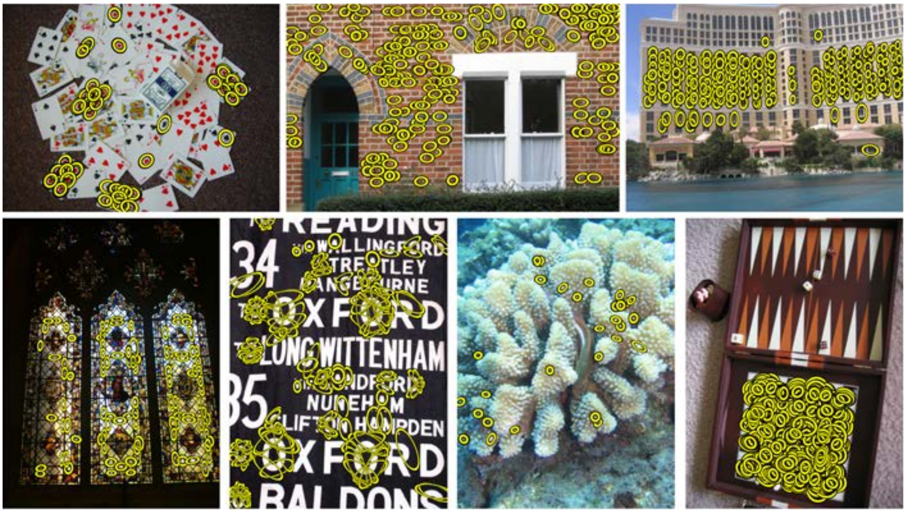


Figure 2.8 – An illustration for the *burstiness effect*, the increased likelihood that similar features occur more frequently within a same image. These visual features may dominate the final representation generated for the image, reducing the visibility of less common, but still present, visual features. Image reproduced from [82].

be able to suppress the burstiness effect completely, it offers a parameter (α) that can be used to fine-tune a model in search for better results (*cf.* [44] for an example where fine-tuning $\alpha = 0.2$ gives better results in image classification using VLAD). In contrast, the intra-normalization does not have any extra parameter, which may simplify model selection during hyper-parameter search.

Classification. After each video has been encoded as a single, fixed-size vector representing its contents, the task of the fifth and last stage in the pipeline is to take this vector and assign it to one or more action classes using a classifier. There are no fundamental constraints limiting which classifier could be used at this stage. Examples for possible classifiers include Neural Networks (28), SVMs (29) or Random Forests (31). However, due the possible high-dimensionality of the video representations originated in the previous stage, some classifiers may be more efficient than others.

Most works in the literature have successfully used the SVM (29) for this purpose [101, 140, 203, 204, 205, 207], with its linear kernel version being shown to be highly appropriate for large scale multi-class and multi-label learning [146]. However, the choice of a proper kernel function may still be highly dependent on the encoding used. For vector quantization encodings such as BoW a Chi-Square (27) type of kernel may be more appropriate (*e.g.,* [201, 203]), whereas for encodings that are natural expansions of existing kernels (*e.g.,* the FV) a linear kernel (26) may be more appropriate [140, 203, 204]. As a sidenote, when comparing works in the literature that do employ SVMs, it might be important to note that some works employ grid-search to find the best hyperparameter C [203], while others use a fixed value of $C = 100$ [101, 205], making their results not directly comparable.

Channel operations. Certain operations can occur during multiple stages of the pipeline. Those operations can divide or merge descriptor channels, and can be used to implement video-level pooling strategies commonly used in the literature. Those include:

- Channel splitting (**D**): Creating separate feature channels according to some common characteristic of the features, such as according to the x, y, and z coordinates of the trajectory path that originated them (*e.g.,* STP, [203]);
- Feature stacking (**S**): Stacking together all features extracted from variations of the same video (*e.g.,* [101]).
- Descriptor-level fusion (**F**): Concatenating together descriptor channels, such as by concatenating PCA-transformed HOG descriptors with the xyt coordinates of the trajectory path that originated them (*e.g.,* STA, as done in [101]);
- Encoding-level fusion (**E**): Concatenating descriptor channels before pooling. This is the case when using SFV: a separate encoding is generated for the feature channel and its spatiotemporal coordinates, both of which are then concatenated together to form a SFV encoding of the feature;
- Representation-level fusion (**R**): Concatenating descriptor channels to create a single representation for the entire video. Examples include the common feature channel concatenation happening before a linear SVM in [101, 140, 203, 204, 205, 207], and the final STP representation in [203];
- Kernel-level fusion (**K**): Merging together kernel matrices computed from the feature channels (*e.g.,* using the multi-channel approach of [222] as in [201]);
- Score-level fusion (**C**): Merging together scores from multiple classifiers to boost classification results, *e.g.,* by averaging classification scores, or by stacking [212] (*cf.* example in Appendix A). This type of fusion is not exclusive of local feature pipelines and can be used to merge results from different approaches (*e.g.,* combination of C3D and iDT probability scores in [187]).

2.2.2 State-of-the-art pipelines

We now use the numbered blocks in Figure 2.6 to denote the processing path followed by multiple works in the literature using a sequence of connected numbers. For example, in order to denote a pipeline that stabilizes a video, extracts HOG descriptors, encodes those descriptors using FV, aggregates those encodings using average pooling, and then classifies these pooled FVs using a linear SVM we write it as:

$$0-5-HOG-14-22-26-29. \tag{2.1}$$

To indicate parallelism between different data paths we separate their sub-paths by listing them within brackets. With this notation, the pipeline of [202] shown in Figure 2.6 can be written as:

$$0-5-\left\{\begin{matrix} Traj-9 \\ \left\{\begin{matrix} HOG \\ HOF \\ MBHx \\ MBHy \end{matrix}\right\}-8-9-10 \end{matrix}\right\}-14-21-23-24-\mathbf{R}-26-29-31. \tag{2.2}$$

Now that we have a new tool at our disposal to characterize and describe BoW-based action recognition pipelines, in this section we use the notation shown in Figure 2.6 to dissect and analyze the conclusions and conflicts between several of the works reported in section 2. The published performance for these methods is shown in Table 2.5.

Action recognition by dense trajectories, CVPR 2011 [201]. In the first publication on dense trajectories [201], Wang *et al.* applied VQ-based BoW to the Traj, HOG, HOF, and MBH descriptors obtained from DT. These descriptors are normalized using the ℓ_2-norm and then quantized using vocabularies of 4000 words, giving origin to 4 descriptor channels. These channels are later fused at the kernel-level by taking a weighted average of their χ^2 Gram (distance) matrices, and then feeding this final distance matrix into a generalized Gaussian kernel [222], as:

$$K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \exp\left\{-\sum_{k=1}^{d} \frac{1}{A^{(k)}} D\left(\boldsymbol{x}_{\boldsymbol{i}}^{(k)}, \boldsymbol{x}_{\boldsymbol{j}}^{(k)}\right)\right\} \tag{2.3}$$

where $D(\boldsymbol{x}_{\boldsymbol{i}}^{(k)}, \boldsymbol{x}_{\boldsymbol{j}}^{(k)})$ is the χ^2 distance between the descriptors at the k-th descriptor channel of video $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$, $A^{(k)}$ is the mean value of the χ^2 distances between

Table 2.5 – Performance of action recognition pipelines analyzed in Section 2.2.2.

		UCF-101	HMDB-51	Hollywood2	High-Five	Olympics	
Publication	Method	%mAcc (s.d.)	%mAcc (s.d.)	%mAP	%mAP+ (s.d.)	%mAP	Equation
[201]	DT+BoW	-	-	58.3	-	-	2.4
[202]	DT+BoW	-	46.6	58.2	-	74.1	2.5
	DT+BoW+STP	-	48.3	59.8	-	77.2	2.6
[204]	DT+BoW	-	47.2	58.5	-	75.4	2.7
	DT+FV	-	52.2	60.1	-	84.7	2.8
	iDT+BoW	-	52.1	62.2	-	83.3	2.9
	iDT+FV	-	55.9	63.0	-	90.2	2.10
	iDT+FV (HD)	-	57.2	64.3	-	91.1	2.11
[205]	iDT+FV	84.8	-	-	-	-	2.12
	iDT+FV+STP	85.9	-	-	-	-	2.13
[203]	DT+SFV+STP	83.5	55.9	63.6	62.5	85.8	2.14
	iDT+SFV+STP	85.7	59.3	66.1	68.1	89.6	2.15
	iDT+SFV+STP (HD)	86.0	60.1	66.8	69.4	90.4	2.16
[101]	iDT+MIFS	89.1	65.1	68.0	-	91.4	2.17
[207]	TDD	90.3	63.2	-	-	-	2.18
	TDD+iDT	91.5	65.9	-	-	-	2.19

Whenever applicable, results represent the average over all splits in each dataset.

the descriptors at the k-th descriptor channel for the training videos, and d is the number of different descriptor channels in the pipeline. This final kernel matrix is then used to create c binary SVMs within a *one-vs-rest* approach for multiple class classification, where c is the number of classes in each target dataset. The steps to reproduce the main pipeline from this work using the notation from Figure 2.6 are:

$$0-\left\{\begin{matrix} Traj \\ \left\{\begin{matrix} HOG \\ HOF \\ \left\{\begin{matrix} MBHx \\ MBHy \end{matrix}\right\} \end{matrix}\right\}-7 \end{matrix}\right\}-11-21-27-\mathbf{K}-29-31. \tag{2.4}$$

Dense trajectories and motion boundary descriptors for action recognition, IJCV 2013 [202]. In a subsequent publication [202], Wang *et al.* would expand their initial investigation, consolidating the efficacy of their methods in action recognition. The authors have extended their experiments considering more datasets and investigating the use of Spatio-Temporal Pyramids (STP) to increase their method's performance. The STP is performed by accumulating separate histograms per region of the video. All descriptor channels from all spatiotemporal regions are fused at the kernel-level using the multi-channel approach of [222] as in their previous

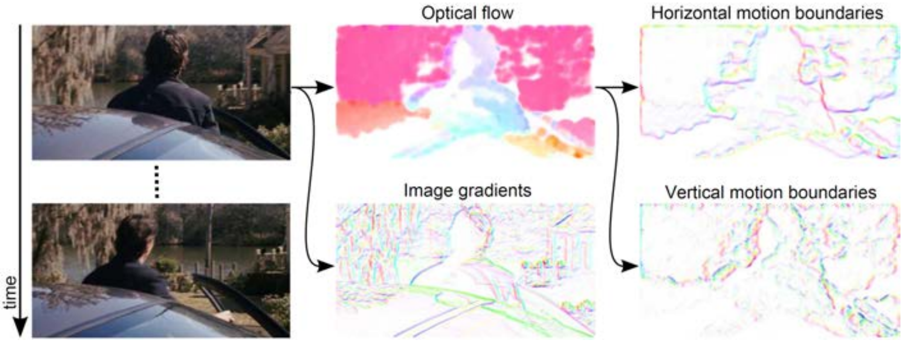


Figure 2.9 – Illustration of the information captured by the HOG, HOF, and MBH descriptors, as reproduced from [202]. As can be seen, the MBH descriptor is composed of two main components, horizontal and vertical, which are computed separately. While those descriptors have been merged together in the first publication about dense trajectories [201], it was later discovered that splitting those components in their own descriptor channels [202] could give a slight edge on performance (*cf.* differences between Equation 2.4 and 2.5 in Table 2.5).

work (*cf.* [201] above). In this work, the authors also explicitly mention that using separate descriptor channels for the horizontal and vertical components of the MBH descriptor (*cf.* [39]) may result in a slight performance increase (*cf.* Figure 2.9). The steps to reproduce the final pipeline from this work are:

$$0 - \left\{ \begin{matrix} Traj \\ \left\{ \begin{matrix} HOG \\ HOF \\ MBHx \\ MBHy \end{matrix} \right\} - 7 \end{matrix} \right\} - 11 - 21 - 27 - \mathbf{K} - 29 - 31, \tag{2.5}$$

and

$$0 - \left\{ \begin{matrix} Traj \\ \left\{ \begin{matrix} HOG \\ HOF \\ MBHx \\ MBHy \end{matrix} \right\} - 7 \end{matrix} \right\} - \mathbf{D} - 11 - 21 - 27 - \mathbf{K} - 29 - 31 \tag{2.6}$$

for the version with STP.

Action Recognition with Improved Trajectories, ICCV 2013 [204]. The successful Dense Trajectories approach would be soon be followed by its improved version in [204]. In this work, the authors discovered that by correcting for camera motion it was possible to significantly enhance the recognition results of their method in many datasets. They also show how it was possible to improve results even further by filtering out human actors during the motion stabilization, therefore forcing the stabilization to be based almost exclusively on the background motion. This work is also the first to explore the use of Fisher Vectors as an alternative for the Bag-of-Words encoding used in [201, 204] and the use of RootSIFT to preprocess histogram-based local descriptors before vocabulary creation and encoding. They test their approach using both the RBF-χ^2 kernel for VQ encodings (as in [201]) and the linear kernel for the FV encoding. For BoW they learn k-Means from 100,000 randomly sampled features with $k = 4000$. For the FV, they learn a GMM with 256 components from a random sample of 256,000 features. All experiments use $C = 100$ when learning SVMs. The concatenation of FVs is done after power and ℓ_2 normalizations. The steps to reproduce the main pipelines from this work are:

$$0 - \left\{ \begin{matrix} Traj \\ \left. \begin{Bmatrix} HOG \\ HOF \\ MBHx \\ MBHy \end{Bmatrix} - 8 - 9 \right. \end{matrix} \right\} - 11 - 21 - 27 - \mathbf{K} - 29 - 31 \tag{2.7}$$

for the baseline BoW version with the split MBH descriptor and now using RootSIFT,

$$0 - \left\{ \begin{matrix} Traj \\ \left. \begin{Bmatrix} HOG \\ HOF \\ MBHx \\ MBHy \end{Bmatrix} - 8 - 9 \right. \end{matrix} \right\} - 10 - 14 - 21 - 23 - 24 - \mathbf{R} - 26 - 29 - 31 \tag{2.8}$$

for the baseline with FV encoding,

$$0 - 6 - \left\{ \begin{matrix} Traj \\ \left. \begin{Bmatrix} HOG \\ HOF \\ MBHx \\ MBHy \end{Bmatrix} - 8 - 9 \right. \end{matrix} \right\} - 11 - 21 - 27 - \mathbf{K} - 29 - 31 \tag{2.9}$$

for the improved trajectories version with BoW encoding,

$$0-6-\left\{\begin{array}{c} Traj \\ \left\{\begin{array}{c} HOG \\ HOF \\ MBHx \\ MBHy \end{array}\right\}-8-9 \end{array}\right\}-10-14-21-23-24-\mathbf{R}-26-29-31 \tag{2.10}$$

for the improved trajectories version with FV encoding, and

$$0-5-\left\{\begin{array}{c} Traj \\ \left\{\begin{array}{c} HOG \\ HOF \\ MBHx \\ MBHy \end{array}\right\}-8-9 \end{array}\right\}-10-14-21-23-24-\mathbf{R}-26-29-31 \tag{2.11}$$

for the improved version with FV encoding and using a human detector to improve motion stabilization. The authors have also used a variation of this method to participate in the THUMOS Challenge 2013 [88]. In their submission [205], the authors considered a pipeline that did not include the trajectory descriptor, did not use human detection to improve camera stabilization, but included STP. The authors also do not mention the use of RootSIFT. Those pipelines could therefore be described as:

$$0-6-\left\{\begin{array}{c} HOG \\ HOF \\ MBHx \\ MBHy \end{array}\right\}-10-14-21-23-24-\mathbf{R}-26-29-31, \tag{2.12}$$

for the version without STP, and

$$0-6-\left\{\begin{array}{c} HOG \\ HOF \\ MBHx \\ MBHy \end{array}\right\}-10-\mathbf{D}-14-21-23-24-\mathbf{R}-26-29-31. \tag{2.13}$$

for the version with STP.

A robust and efficient video representation for action recognition, IJCV 2015 [203]. In the extended publication about iDT [203], Wang *et al.* apply STPs and use SFV to further boost the results of FV encodings of iDT local features. The authors also perform a grid-search for the SVM parameter C in all their experiments instead of using a fixed $C = 100$ as in [201, 204]. As in [205], the authors do not include the trajectory descriptor, but do include RootSIFT. The steps to reproduce the main pipeline from this work are:

$$0-\left\{\begin{array}{l}\left\{\begin{array}{c}HOG-8-9-10-14\\ xyt-14\end{array}\right\}-\mathbf{E}\\ \left\{\begin{array}{c}HOF-8-9-10-14\\ xyt-14\end{array}\right\}-\mathbf{E}\\ \left\{\begin{array}{c}MBHx-8-9-10-14\\ xyt-14\end{array}\right\}-\mathbf{E}\\ \left\{\begin{array}{c}MBHy-8-9-10-14\\ xyt-14\end{array}\right\}-\mathbf{E}\end{array}\right\}-21-23-24-\mathbf{R}-26-29-31 \tag{2.14}$$

for the baseline version,

$$0-6-\left\{\begin{array}{l}\left\{\begin{array}{c}HOG-8-9-10-14\\ xyt-14\end{array}\right\}-\mathbf{E}\\ \left\{\begin{array}{c}HOF-8-9-10-14\\ xyt-14\end{array}\right\}-\mathbf{E}\\ \left\{\begin{array}{c}MBHx-8-9-10-14\\ xyt-14\end{array}\right\}-\mathbf{E}\\ \left\{\begin{array}{c}MBHy-8-9-10-14\\ xyt-14\end{array}\right\}-\mathbf{E}\end{array}\right\}-21-23-24-\mathbf{R}-26-29-31 \tag{2.15}$$

for the camera-stabilized version, and

$$0-5-\left\{\begin{array}{l}\left\{\begin{array}{c}HOG-8-9-10-14\\ xyt-14\end{array}\right\}-\mathbf{E}\\ \left\{\begin{array}{c}HOF-8-9-10-14\\ xyt-14\end{array}\right\}-\mathbf{E}\\ \left\{\begin{array}{c}MBHx-8-9-10-14\\ xyt-14\end{array}\right\}-\mathbf{E}\\ \left\{\begin{array}{c}MBHy-8-9-10-14\\ xyt-14\end{array}\right\}-\mathbf{E}\end{array}\right\}-21-23-24-\mathbf{R}-26-29-31 \tag{2.16}$$

for the camera-stabilized version using a human detector to improve motion stabilization.

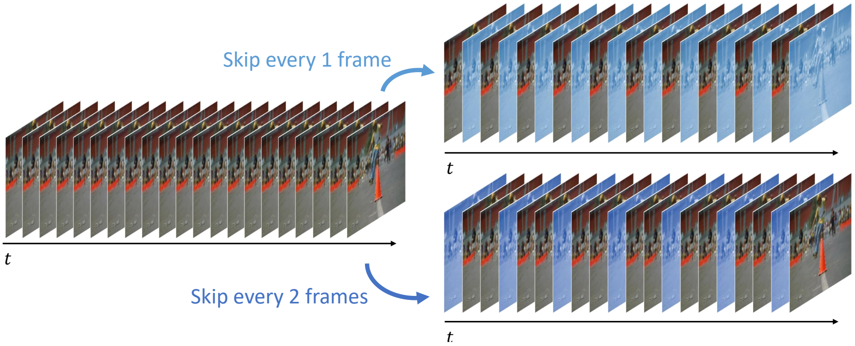


Figure 2.10 – Multiple versions of the same video considering different frame-skips.

Beyond Gaussian Pyramid: Multi-skip Feature Stacking for Action Recognition, CVPR 2015 [101]. Lan *et al.* [101] studied the effects of including frame-skipped versions of the input video in the trajectories pipeline. They have shown that augmenting data by considering different frame-skipped variations of the input video (*cf.* Figure 2.10) improves the learnability of feature matrices generated by feature extractors based on differential filters, such as MBH and STIP. In their pipeline, the authors apply RootSIFT normalization to all descriptors, including the pixel trajectories (Traj). They also use STA (referred therein as STED) to incorporate spatiotemporal information in the descriptors, and re-normalize FV after descriptor-channel fusion. The steps to reproduce the main pipeline from this work are:

$$\left\{\begin{matrix}0\\1\\2\\3\end{matrix}\right\} - 6 - \left\{\begin{matrix} \left\{\begin{matrix} Traj - \mathbf{S} - 8 - 9 - 10 \\ xyt - \mathbf{S} \end{matrix}\right\} - \mathbf{F} \\ \left\{\begin{matrix} HOG - \mathbf{S} - 8 - 9 - 10 \\ xyt - \mathbf{S} \end{matrix}\right\} - \mathbf{F} \\ \left\{\begin{matrix} HOF - \mathbf{S} - 8 - 9 - 10 \\ xyt - \mathbf{S} \end{matrix}\right\} - \mathbf{F} \\ \left\{\begin{matrix} MBHx - \mathbf{S} - 8 - 9 - 10 \\ xyt - \mathbf{S} \end{matrix}\right\} - \mathbf{F} \\ \left\{\begin{matrix} MBHy - \mathbf{S} - 8 - 9 - 10 \\ xyt - \mathbf{S} \end{matrix}\right\} - \mathbf{F} \end{matrix}\right\} - 14 - 21 - 23 - 24 - \mathbf{R} - 23 - 24 - 29 - 31. \tag{2.17}$$

As can be seen, this pipeline incorporates multiple differences from the previous pipelines shown in [201, 202, 203, 204], such as STA and double normalization besides MIFS. An analysis of each of those differences in isolation will be presented in Section 3.2 of this thesis.

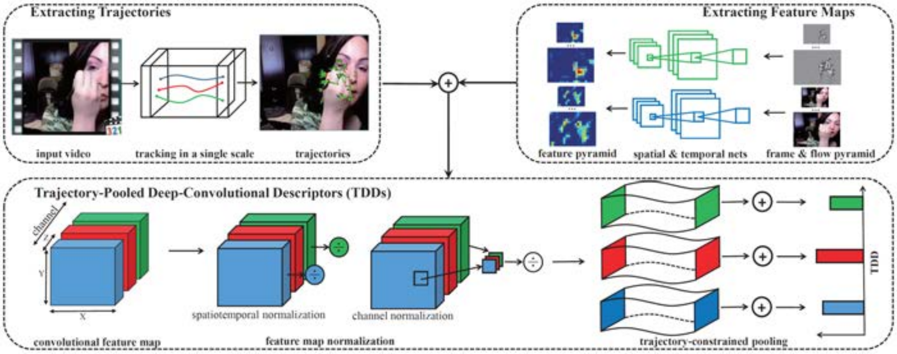

Extracting Trajectories
input video
tracking in a single scale
trajectories
Extracting Feature Maps
feature pyramid
spatial & temporal nets
frame & flow pyramid
Trajectory-Pooled Deep-Convolutional Descriptors (TDDs)
channel
Y
X
spatiotemporal normalization
channel normalization
convolutional feature map
feature map normalization
trajectory-constrained pooling
TDD

Figure 2.11 – The TDD pipeline. Image reproduced from [207].

Action Recognition with Trajectory-Pooled Deep-Convolutional Descriptors, CVPR 2015 [207]. In [207], the authors propose a method to extract local features from action recognition videos using deep convolutional networks and show how those features can be incorporated in a dense trajectories pipeline (*cf.* Figure 2.11). In order to do so, the authors apply the Two-Stream Network approach of [170] to obtain multiple feature maps for each frame in a video. The Two Stream Network approach (*cf.* Section 2.3.2 for a more in-depth description) involves learning two deep convolutional neural networks for video classification. One network processes individual RGB frames and is therefore referred as the spatial stream processing network (T^s), and the processes a stack of optical flow frames and is therefore referred as the temporal stream processing network (T^t). Since both networks are based on a the VGG-16 architecture [171] which is itself close to AlexNet [98], it becomes possible to extract meaningful features from individual convolutional layers such as *conv4* and *conv5* as done in image classification tasks [9, 62, 126, 217].

After extracting feature maps, the authors extract regions from those maps across the same optical flow trajectories that are used in iDT (with a modification to track at a single scale instead of at multiple scales), aggregating and normalizing those features in different ways. They also consider multiple normalization strategies and multiple convolutional layers from both spatial and temporal networks to create a set of Trajectory-pooled Deep-convolutional Descriptors (TDD) descriptors which are then FV-encoded and processed in the same way as other iDT descriptors. Furthermore, the authors find that it is still possible to retrieve better results by combining these descriptors with iDT, indicating that there is still some orthogonality between the deeply-learned descriptors and handcrafted features. On the other hand, the authors do not mention which normalizations are applied after the FV-encoding and before classification.

The final pipelines shown in this work includes 8 different variations of the TDD descriptors. These are the result of considering the two networks (T^s and (T^t)), feature maps from two convolutional layers from each network (*conv3* and *conv4* for spatial networks, *conv4* and *conv5* for temporal networks), and two normalization strategies when aggregating feature maps across the trajectory paths to form the TDD descriptor: Spatiotemporal Normalization ($\mathrm{TDD_{st}}$) and Channel Normalization ($\mathrm{TDD_c}$). The steps to reproduce the main pipelines as described in [207] would be:

$$0-6-\left\{\begin{matrix} TDD_{\mathrm{st}}(T^s_{conv4}) \\ TDD_{\mathrm{st}}(T^s_{conv5}) \\ TDD_{\mathrm{st}}(T^t_{conv4}) \\ TDD_{\mathrm{st}}(T^t_{conv3}) \\ TDD_{\mathrm{c}}(T^s_{conv4}) \\ TDD_{\mathrm{c}}(T^s_{conv5}) \\ TDD_{\mathrm{c}}(T^t_{conv4}) \\ TDD_{\mathrm{c}}(T^t_{conv3}) \end{matrix}\right\}-10-14-21-\mathbf{R}-26-29-30, \tag{2.18}$$

and

$$0-6-\left\{\begin{matrix} \left\{\begin{matrix} TDD_{\mathrm{st}}(T^s_{conv4}) \\ TDD_{\mathrm{st}}(T^s_{conv5}) \\ TDD_{\mathrm{st}}(T^t_{conv4}) \\ TDD_{\mathrm{st}}(T^t_{conv3}) \\ TDD_{\mathrm{c}}(T^s_{conv4}) \\ TDD_{\mathrm{c}}(T^s_{conv5}) \\ TDD_{\mathrm{c}}(T^t_{conv4}) \\ TDD_{\mathrm{c}}(T^t_{conv3}) \end{matrix}\right\} \\ \left\{\begin{matrix} Traj \\ \left\{\begin{matrix} HOG \\ HOF \\ MBHx \\ MBHy \end{matrix}\right\}-8-9 \end{matrix}\right\} \end{matrix}\right\}-10-14-21-\mathbf{R}-26-29-30 \tag{2.19}$$

for the combined version with iDT (using representation-level fusion and assuming exactly the same parameters as in [202]). While this work has used convolutional neural networks solely for the purposes of feature extraction, in the next sections we will see multiple approaches demonstrating how those networks can be used for end-to-end video classification.

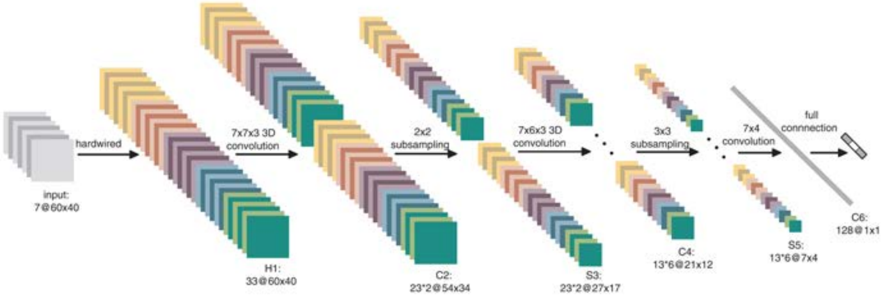


Figure 2.12 – The 3D CNN architecture for action recognition, reproduced from [85]. The first layer in this architecture is hardwired to produce feature maps that correspond to a grayscale transformation of the frames, their horizontal and vertical image gradients, and horizontal and vertical components of their optical flow fields.

2.3 Deep learning approaches

Deep learning models, such as deep neural networks (DNN), are a class of methods and models that are able to learn hierarchical features and prediction models directly from raw data. Since the beginning of this decade, these models have found remarkable success in addressing both small- and large-scale problems in image recognition, such as CIFAR-10, CIFAR-100 [97], PASCAL Visual Object Classes (VOC) [48], and the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [87, 160].

2.3.1 3D convolutional networks

Most successful deep neural networks for image classification are based on the application of 2D convolutional filters. Those filters can be used to process each frame individually, therefore operating in a $\mathbb{R}^2$, spatial domain. On the other hand, a single video can be described as a sequence of static frames, and therefore as a *volume* in a $\mathbb{R}^3$ spatiotemporal domain. As such, it would seem natural to expect that, by proposing a deep neural network architecture that could use 3D convolutional filters instead of 2D ones, one could obtain equally successful models for video classification. However, while multiple works explored this direction [85, 182, 187], it would take several years until 3D convolutional networks (3D CNNs) would perform better methods based on handcrafted features or variations of 2D convolutional networks that had been adapted for video [170] (*e.g.,* [22]).

One of the earliest works in this direction was [85], where the authors devised a 3D CNN architecture specifically for the purpose of action recognition. While this method achieved the best performance in the *TRECVID 2009 Surveillance Event Detection* [135] challenge and competitive results in the small scale KTH dataset [164], it would soon be surpassed by approaches based on handcrafted features in both KTH [203, 204] and in related later editions of the challenge [2]. Moreover, the 3D architectures used in this work were highly hand-tuned. The architecture included a hardwired first layer with fixed weights (*cf.* Figure 2.12) used to encode prior knowledge in the features, as well as very specific organizations of layers which had to be combined together in five different model variations to achieve their best results. Training those architectures also required employing a complex mechanism for model regularization in order to keep the learned features consistent across different points in time through a set of auxiliary feature extractors and auxiliary outputs.

In [187], the authors have proposed a similar 3D convolutional architecture that learns from fixed temporal windows of 16 consecutive frames. However, unlike [85], all 3D convolution filters in their architecture shared the same 3 × 3 × 3 size with a stride of 1 × 1 × 1. While their method achieved better performance on Sports-1M than the 2D architectures studied in [91], their method still performed significantly worse than multiple approaches based on handcrafted features when measured on UCF-101 [43, 101, 203] – even though their network had been pre-trained on *millions* of extra videos and on an undisclosed amount of images from an internal large-scale image dataset (referred to therein as I380k). Moreover, their best results could only be achieved when combined with the same handcrafted approaches they compare against, giving an indication that their method could produce features that are complementary, but still not better, than the state of the art.

2.3.2 Two-Stream Networks

The Two-stream Network (2SN) approach of Simonyan *et al.* [170] would become a milestone in the field of action recognition. While [170] could present results that were competitive to handcrafted features but still could not surpass their state of the art, this architecture would become the basis of increasingly more successful methods [41, 207, 208], until they would finally prevail as the de-facto state-of-the-art architecture for human action recognition in videos [22].

The main idea in the two-stream architecture was to employ two parallel networks (*cf.* Figure 2.13) that could be learned separately from two different data streams: one spatial, comprised of sequences of RGB frames; and one temporal,

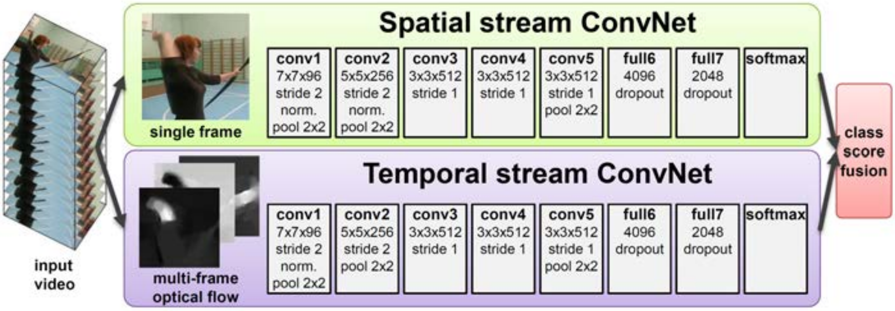


Figure 2.13 – Two stream Network architecture for action recognition, reproduced from [170]. This architecture is composed of two networks: one that process individual RGB frames and therefore operates in the spatial domain (Spatial stream ConvNet, top); and one that process a fixed-size stack of sequential optical flow frames (Temporal stream, bottom). These two networks are learned separately during the learning phase, and their classification scores are averaged during the evaluation phase.

comprised of sequences of optical flow frames. These two networks would share an almost identical architecture except for the first layers. The first layers of the spatial network would expect individual RGB frames as input in a similar way to a standard image classification network [28], whereas the first layers of the temporal network would expect stacked (*e.g.,* concatenated along the temporal dimension) optical flow displacement fields between consecutive frames.

Since the spatial network is based on a common architecture for image classification, this network can be pre-trained on a large scale image dataset (ImageNet ILSVRC-2012) before being fine-tuned on the target video dataset. Pre-training had been show to be an effective strategy for initializing the weights of deep convolutional neural networks [119, 134, 220], even when those networks are used for different tasks. It also provides prior knowledge about features and classes which may not be present in the training set of the target dataset but present in its testing set or when evaluating such models in the field. In [170], Simonyan *et al.* used pre-training to initialize the spatial networks, while training the temporal ones from scratch. Later works would then propose methods to apply this same initialization to both networks, as done for Wang *et al.* 's Temporal Segment Networks [208].

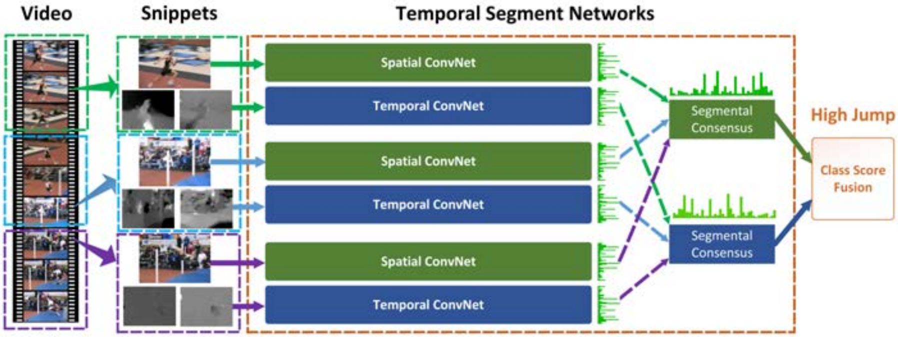


Figure 2.14 – Temporal Segment Networks architecture for action recognition, reproduced from [208]. Just as Simonyan *et al.* 's Two Stream Networks, this architecture employs two separate networks that operate in parallel for RGB and optical flow streams. However, those networks learn from multiple video segments at the same time. The figure exemplifies their working for the case of $k = 3$ video segments.

2.3.3 Temporal Segment Networks

The recent Temporal Segment Networks (TSN) architecture of [208] improves significantly on the original two-stream architecture of [170]. It processes both RGB frames and stacked optical flow frames using a deeper Inception architecture [183] with Batch Normalization [78] and DropOut [175]. Although it still requires massive labeled training sets, this architecture is more data efficient than the 2SN and 3D CNNs architectures discussed previously in this chapter, and therefore more suitable for action recognition in videos. In particular, [208] shows that both the appearance and motion streams of the TSNs can benefit from a strong initialization on ImageNet's image classification task, which is one of the main factors responsible for their high recognition accuracy.

Another improvement of the TSN with respect to Simonyan *et al.* 's 2SN is the explicit use of long-range temporal structure by jointly processing random short snippets from a uniform temporal subdivision of a video. The TSN computes separate predictions for k different temporal segments of a video, which are then condensed into a video-level decision using a segmental consensus function *e.g.,* the average of their scores (*cf.* Figure 2.14).

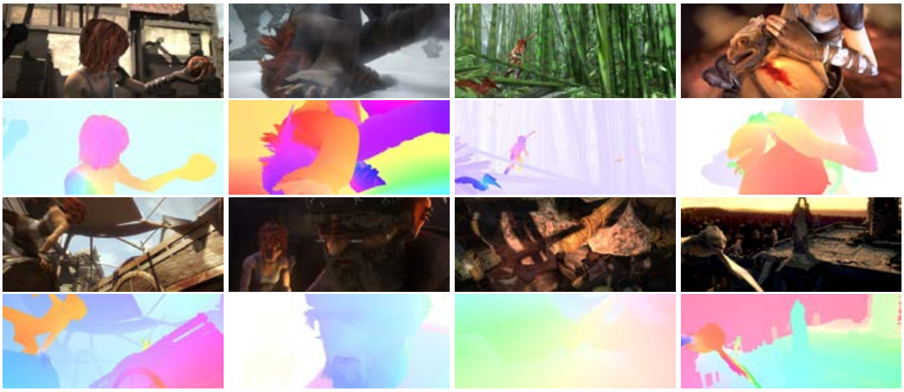


Figure 2.15 – Synthetic images from the SINTEL dataset and their respective pixel-perfect, ground-truth optical flow. Images adapted from [20].

2.4 The case for synthetic data and virtual worlds

During the investigation of this PhD thesis, virtual worlds gained momentum as a reliable technique for generating synthetic training data for many visual tasks [20, 57, 117, 156, 158]. This technique had been shown to be particularly suitable in the case of videos, where manually labeling every pixel in a sequence of frames is extremely difficult or even impossible (*e.g.*, optical flow). Due this difficulty in collecting and annotating video modalities, training data for video has been scarce. This scarcity of adequate labeled training data is widely accepted as a major bottleneck of deep learning algorithms for important video understanding tasks like action recognition.

Several works used synthetic scenarios to evaluate the performance of different feature descriptors [8, 89, 195, 196, 197] and to train and test optical and/or scene flow estimation methods [20, 117, 118, 133], stereo algorithms [66], or trackers [57, 185]. Among those, the SINTEL dataset [20] was one of the first popular datasets for evaluating optical flow (*cf.* Figure 2.15). This dataset would later be used to measure important advances in many tasks, from optical flow estimation [46, 77] to action recognition [90]. This work would later be followed by the FlyingThings3D, Monkaa, and Driving datasets proposed in [117], which would provide dense ground-truth for optical flow, disparity map, and disparity change estimation (*cf.* Figure 2.16) to further help advancing these fields.

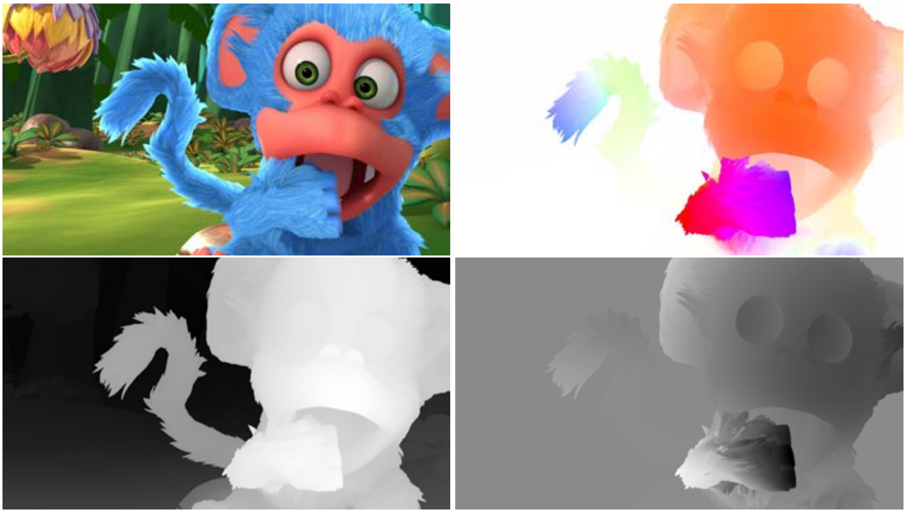


Figure 2.16 – A synthetic image frame from the Monkaa short video clip (top left) and its respective ground-truth in multiple modalities: optical flow (top right), disparity map (bottom left), and disparity change (bottom right). Images adapted from [117].

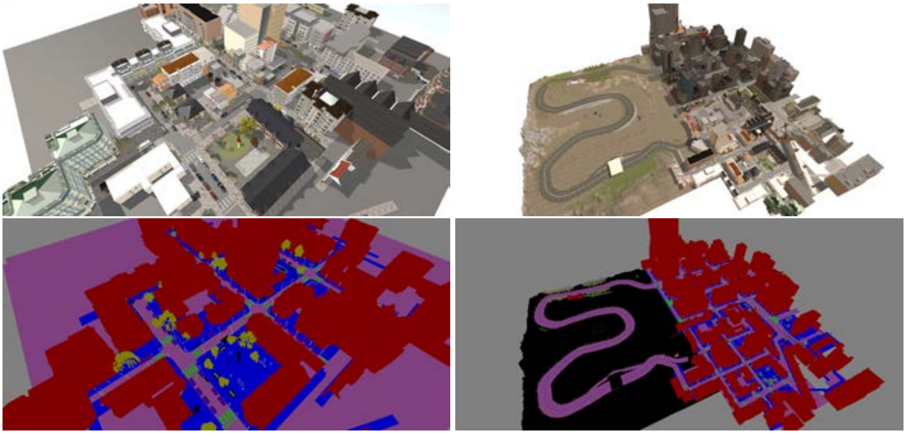


Figure 2.17 – The virtual world of SYNTHIA. Images adapted from [158].

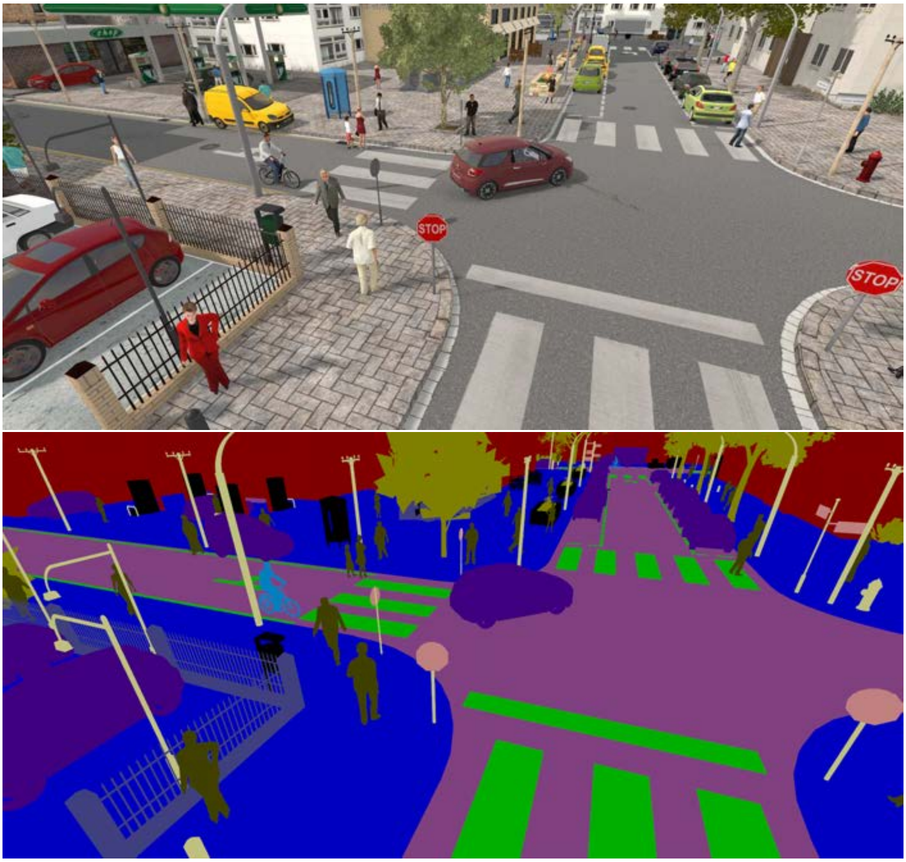


Figure 2.18 – Synthetic images from the SYNTHIA dataset, adapted from [158].

Synthetic data has also been used to train visual models for object detection and recognition, pose estimation, and indoor scene understanding [7, 30, 68, 69, 72, 113, 115, 136, 142, 143, 152, 163, 169, 178, 179, 192, 215, 216]. Haltakov *et al.* [67] used a virtual racing circuit to generate different types of pixel-wise ground truth (depth, optical flow and class labels). Ros *et al.* [156, 158] relied on game technology to create a virtual world (Figure 2.17) to train deep semantic segmentation networks using synthetically-generated ground-truth from this world (*cf.* Figure 2.18). Gaidon *et al.* [57] used a similar technique for multi-object tracking (*cf.* Figure 2.19), [168] for depth estimation from RGB, and [172] for place recognition.

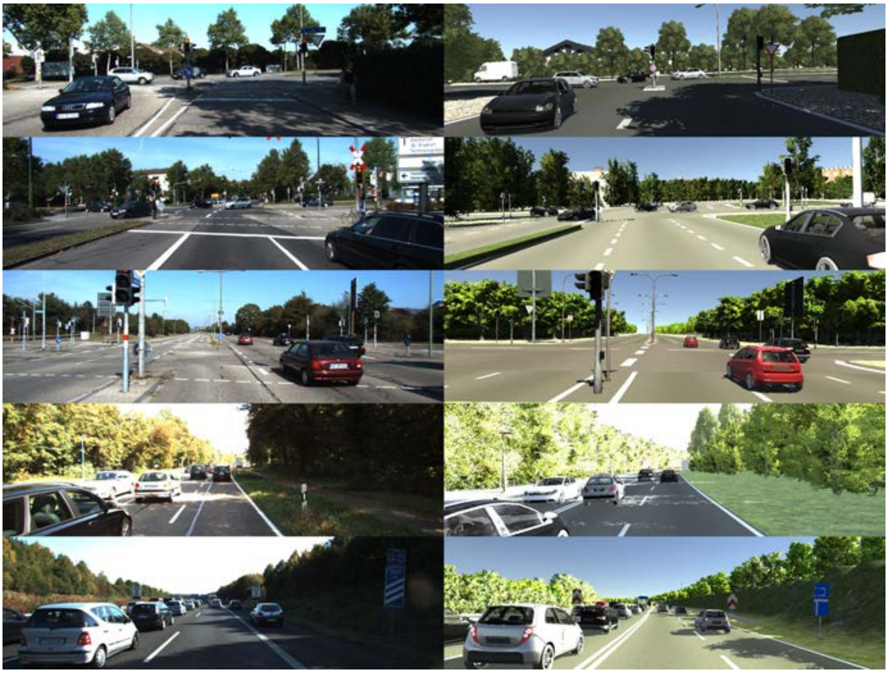


Figure 2.19 – Synthetic images from the Virtual KITTI dataset, adapted from [57].

Synthetic data and virtual worlds have also been used for learning artificial behaviors such as playing Atari games [122], imitating players in shooter games [6], end-to-end driving and navigation [29, 228], learning common sense [194, 229], and learning physical intuitions [108]. They have also been explored from an animator's perspective: Works in computer graphics have investigated producing animations from sketches [65], using physical-based models to add motion to sketch-based animations [64], and creating constrained camera-paths [58]. However, due to the formidable complexity of realistic animation, video generation, and scene understanding, these approaches focus on simple controlled game environments, motions, and action spaces.

In this regard, and to the best of our knowledge, the work we present in Chapter 4 was the first to investigate virtual worlds and game engines to generate synthetic training videos for action recognition. Although some of the aforementioned related works rely on virtual characters, these works do not focus on their actions, and neither are their actions procedurally generated, meaning their actions are often reduced to simply walking.

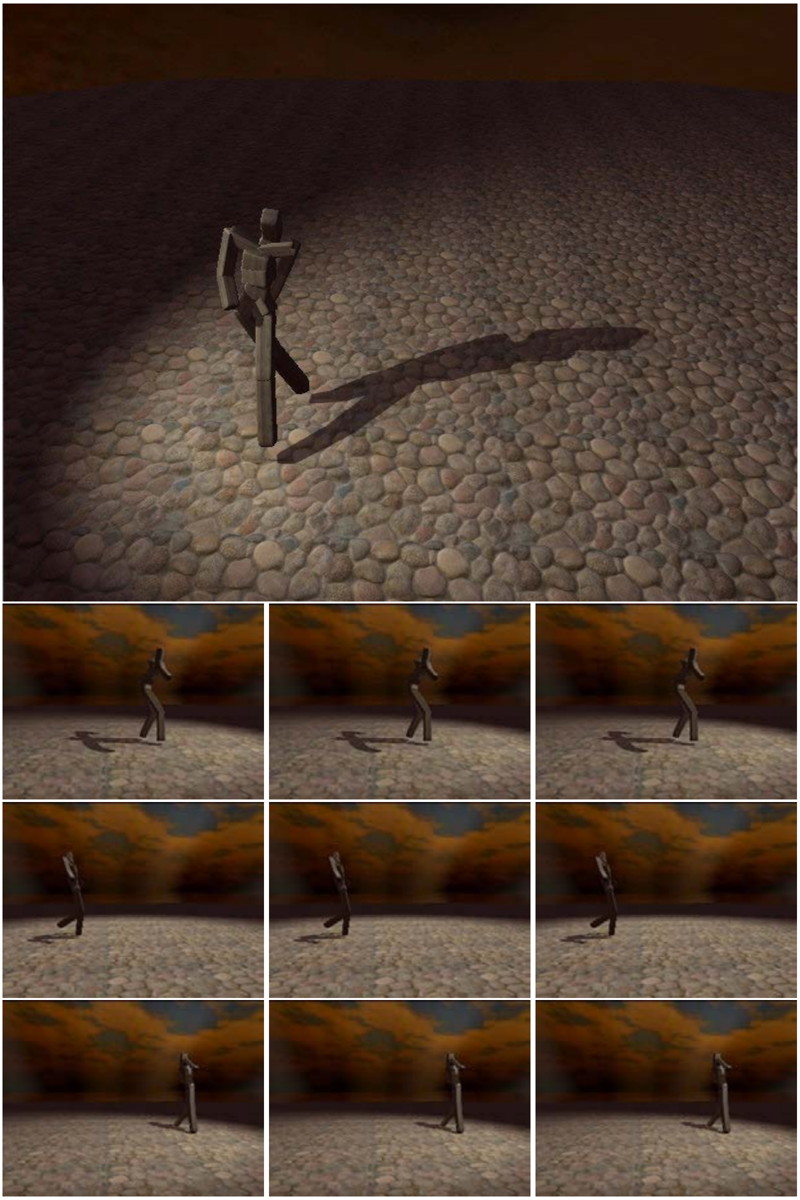


Figure 2.20 – Example frames from Matikainen *et al.* 's synthetic data, adapted from [116]. While the armature is abstract in appearance, its movement is derived from MOCAP data.

Perhaps one of the closest – but still highly distinct – related work in this direction is [116]. In this work Matikainen *et al.* use MOCAP data to induce realistic motions in an "abstract armature" placed in an empty synthetic environment, generating 2,000 short 3-second clips at 320 × 240 and 30FPS. From these non-photorealistic clips, handcrafted motion features are selected as relevant and later used to learn action recognition models for 11 actions in real-world videos (*cf.* Figure 2.20). In contrast, the approach we describe in Chapter 4 *does not just replay MOCAP, but procedurally generates new action categories* – including interactions between persons, objects and the environment – as well as random *physically plausible* variations of these new actions.

A recent alternative to virtual worlds that also does not require manual video labeling is the unsupervised Video Generative Adversarial Network (VGAN) of [198] (*cf.* Figure 2.21). Instead of leveraging prior structural knowledge about physics and human actions, Vondrick *et al.* view videos as tensors of pixel values and learn a two-stream GAN on 5,000 hours of unlabeled Flickr videos. Their method focuses on tiny and short videos, generating video cuboids composed of 32 frames with a spatial resolution of 64 × 64 and 25 FPS (*cf.* Figure 2.22 for example frames). Their generator network is composed of two streams, one comprised of spatial convolutions for generating static backgrounds, and another of spatiotemporal con-

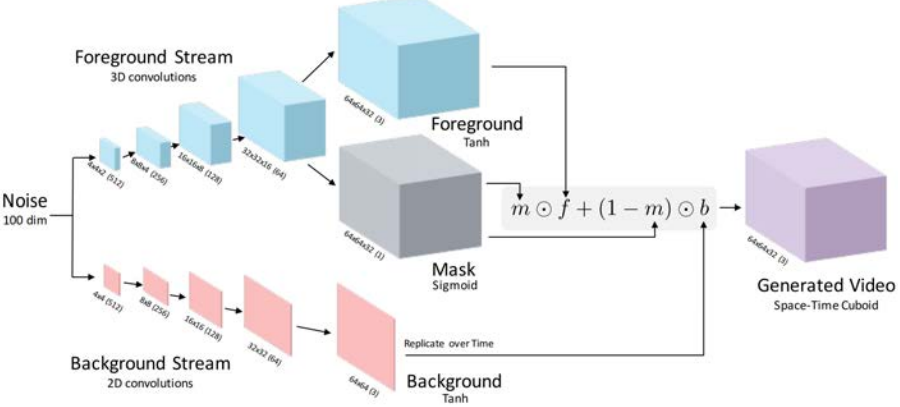


Figure 2.21 – A depiction of the VGAN architecture for unsupervised video generation, adapted from [198]. The architecture is divided in two streams: one focused on generating static background scenes, and another which focus on moving foreground elements. The input of both streams is a 100-dimensional vector comprised of Gaussian noise.

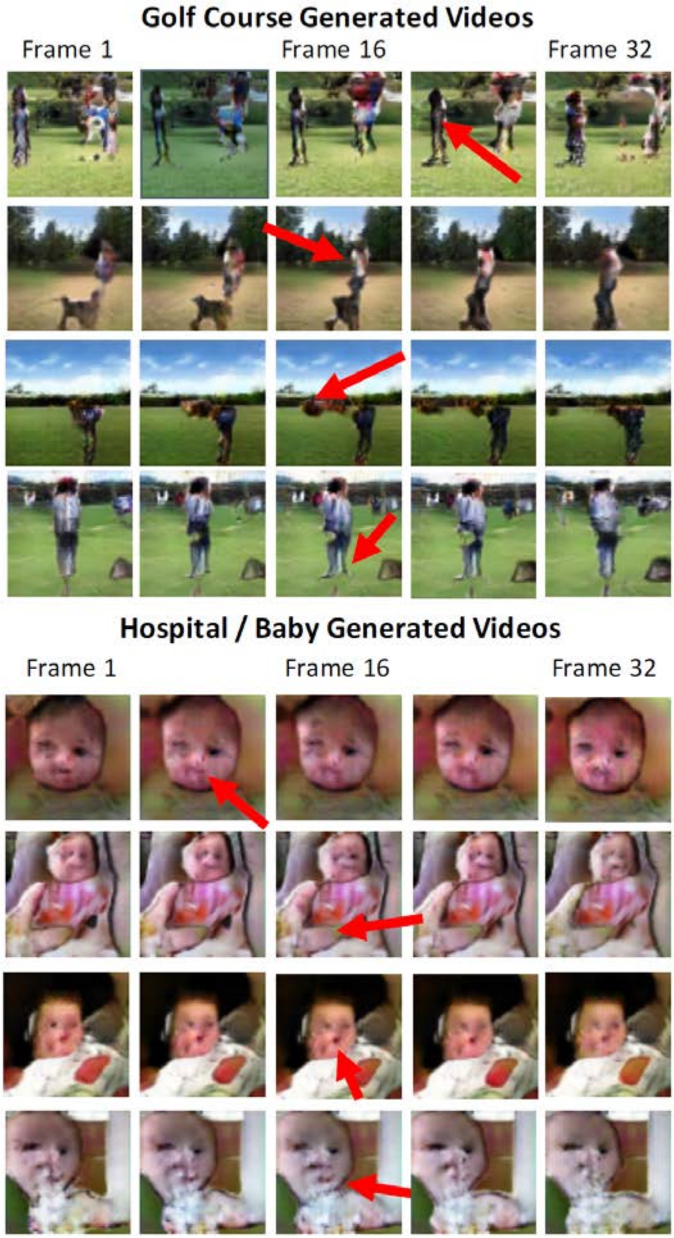

Golf Course Generated Videos
Frame 1
Frame 16
Frame 32
Hospital / Baby Generated Videos
Frame 1
Frame 16
Frame 32

Figure 2.22 – Examples of synthetic video frames generated by VGAN, adapted from [198]. The red arrows highlight examples of generated motions.

volutions for generating moving elements in the foreground. Their discriminator network is modeled as the reverse of the foreground stream, and therefore consists in a 5-layer 3D CNN whose last layer performs binary classification with the goal of discriminating between realistic and non-realistic videos. This architecture can be used for action recognition in videos by replacing the last binary classification layer from the discriminator network with a prediction layer and then fine-tuning this model on a set of labeled videos. In comparison, the approach we propose in Chapters 4 and 5 allows one to work with any state-of-the-art discriminative architecture, as video generation and action recognition are completely decoupled steps. We can, therefore, benefit from a strong ImageNet initialization for both appearance and motion streams as in Wang *et al.* 's TSN [208] – while at the same time, we can decide what specific actions, scenarios, and camera-motions to generate, enforcing diversity thanks to our interpretable parametrization.

2.5 Summary of the chapter

In this chapter we presented several of the most relevant works to the literature that are mostly correlated with the work presented in this thesis. In particular, we have explored the transition from handcrafted to deep learning models in the field of action recognition that occurred during the execution of this work. We have also identified sevevral key challenges related to both handcrafted and deep methods, as well as methods to possibly overcome them (*e.g.,* training using virtual data). In the next chapters, we will be re-using all this acquired knowledge to improve the state of the art in both classes of methods, proposing a new hybrid architecture for action recognition that unites the best of both worlds, as well as methods to generate better deep learning networks through the automatic, procedural generation of synthetic human action videos that can be used as a complement to real world data when such data is insufficient or otherwise hard to acquire.

3 Hybrid models for action recognition

> It might be said now that I have the best of both worlds.
>
> John F. Kennedy

In the previous chapter we presented an overview and analysis of various methods for video action recognition according to whether they were based on handcrafted features *vs.* deep features, and shallow classifiers *vs.* deep classifiers, exploring their advantages and disadvantages. In this chapter, we use this analysis to derive a strong handcrafted baseline for action recognition, and then show how to push the state of the art even further, proposing a new hybrid architecture combining the strengths of the dense trajectories pipeline and supervised deep multi-layer non-linear classifiers.

3.1 Introduction

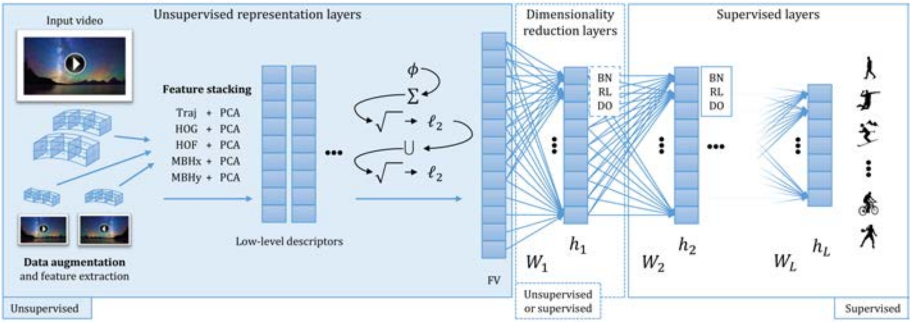
CURRENT STATE-OF-THE-ART ALGORITHMS for action recognition belong to two main categories: models relying on features *handcrafted* for action recognition (*e.g.,* [51, 56, 75, 101, 140, 201, 202, 203, 204]), or more recent end-to-end *deep architectures* (*e.g.,* [11, 22, 45, 49, 59, 85, 91, 127, 170, 176, 182, 208, 211, 213]). These approaches have complementary strengths and weaknesses. Models based on handcrafted features are data efficient, as they can easily incorporate structured prior knowledge (*e.g.,* the importance of motion boundaries along dense trajectories [201]), but their lack of flexibility may impede their robustness or modeling capacity. On the other hand, deep models make fewer assumptions and are learned end-to-end from data (*e.g.,* using 3D-ConvNets [187]), but they rely on handcrafted architectures and the acquisition of large manually labeled video datasets (*e.g.,* Sports-1M [91]), a costly and error-prone process that poses optimization, engineering, and infrastructure challenges.

Although deep learning for videos has recently made significant improvements (*e.g.,* [170, 182, 187]), models using handcrafted features prevailed as the the state of the art on many standard action recognition benchmarks at the beginning of this investigation (*e.g.,* [51, 75, 101]). These models are generally based on *improved Dense Trajectories* (iDT) [202, 204] with Fisher Vector (FV) encoding [147, 150] of local spatiotemporal descriptors (trajectory coordinates, HOG, HOF, MBH) computed from RGB and optical flow inputs. Recent deep models for action recognition therefore combine their predictions with complementary ones from iDT-FV for better performance [187, 207].

In this chapter, we study an alternative strategy to *combine the best of both worlds via a single hybrid classification architecture* consisting in chaining sequentially the iDT handcrafted features, the unsupervised FV representation, unsupervised or supervised dimensionality reduction, and a supervised deep network (*cf.* Figure 3.1). This family of models was shown in [148] to perform on par with the deep convolutional network of Krizhevsky *et al.* [98] for large scale image classification. We adapt this type of architecture differently for action recognition in videos with particular care for data efficiency.

In Section 2.1.2 we have organized existent works into *handcrafted features, shallow classifiers*; *deep-based features, shallow classifiers*; *deep-based features, deep classifiers*; and *handcrafted features, deep classifiers* (Table 2.4). The work we present in this chapter lies in this last category: it combines the strengths of iDT-FV encodings and supervised deep multi-layer non-linear classifiers. Our method is inspired by the work of Perronnin and Larlus [148], who stack several unsupervised FV-based and supervised layers. Their hybrid architecture shows significant improvements over the standard FV pipeline, closing the gap on [98], which suggests there is still much to learn about FV-based methods. This chapter investigates this type of hybrid architectures, however with several noticeable differences from [148]: (i) unlike in the field of image classsificiation, the FV is on par with the current state of the art for action recognition; (ii) iDT features contain many different appearance and motion descriptors, which also results in more diverse and higher-dimensional FV; and (iii) most action recognition training sets are small due to the cost of labeling and processing videos, so overfitting and data efficiency are major concerns. In this context, we adopt different techniques from modern handcrafted and deep models, and perform a wide architecture and parameter study showing conclusions regarding many design choices specific to action recognition.

The **first contribution** presented in this chapter consists in a careful design of the first *unsupervised* part of our hybrid architecture, which even with a simple SVM classifier is already on par with the state of the art. We experimentally observe that by giving *attention to details* (*e.g.,* spatiotemporal structure, normalization) and doing *data augmentation by feature stacking* (instead of duplicating training



Handcrafted features are extracted along optical flow trajectories from original and generated videos. Those features are then normalized using RootSIFT [3], PCA-transformed, and augmented with their (x, y, t) coordinates, forming our low-level descriptors. The descriptors for each feature channel are then encoded (ϕ) as Fisher Vectors, separately aggregated (Σ) into a video-level representation, square-rooted, and ℓ_2-normalized. These representations are then concatenated ($\cup$) and renormalized. A dimensionality reduction layer is learned supervisedly or unsupervisedly. Supervised layers are followed by Batch-Normalization (BN) [78], ReLU (RL) non-linearities [124], and Dropout (DO) [175] during training. The last layer uses sigmoids (multi-label datasets) or softmax (multi-class datasets) non-linearities to produce action-label estimates.

Figure 3.1 – Our hybrid unsupervised and supervised deep multi-layer architecture.

samples in more common data-augmentation approaches) are critical for achieving good performance, and we show that the optimal design decisions we take in our study do generalize across many datasets.

The **second contribution** we present in this chapter is our *data efficient hybrid architecture* combining unsupervised representation layers with a deep network of multiple fully connected layers. We show that *supervised mid-to-end learning of a dimensionality reduction layer together with non-linear classification layers* yields an excellent compromise between recognition accuracy, model complexity, and transferability of the model across datasets thanks to reduced risks of overfitting and modern optimization techniques.

This chapter is organized as follows. Section 3.2 presents the details of the first unsupervised part (based on iDT-FV) of our hybrid model, while Section 3.3 does so for the rest of the architecture and our learning algorithm. In Section 3.4 we report experimental conclusions from parametric studies and comparisons to the state of the art on five widely used action recognition datasets of different sizes. In particular, we show that our hybrid architecture improves significantly upon the state of the art, including recent combinations of iDT-FV predictions with deep models trained on millions of images and videos.

3.2 Fisher Vectors: From baseline to state of the art

We first recall the iDT approach of Wang & Schmid [204], then describe the improvements that can be stacked together to transform this strong baseline into a state-of-the-art method for action recognition. In particular, we propose a data augmentation by feature stacking method motivated by MIFS [101] and data augmentation for deep models.

3.2.1 Improved Dense Trajectories

Local spatiotemporal features. The iDT approach used in many state-of-the-art action recognition algorithms (*e.g.,* [51, 101, 140, 141, 202, 203, 204]) consists in first extracting dense trajectory video features [201] that efficiently capture appearance, motion, and spatiotemporal statistics. Those features include Trajectory shape (Traj) [201], HOG [38], HOF [39], and MBH [201] descriptors, which are extracted along trajectories obtained by median filtering dense optical flow. In this work, we extract dense trajectories from videos in the same way as in [204], applying RootSIFT normalization [3] (ℓ_1 normalization followed by square-rooting) to all descriptors. The dimensions of the Traj, HOG, HOF, MBHx, and MBHy descriptors are respectively 30, 96, 108, 96, and 96.

Unsupervised representation learning. Before classification, we combine the multiple trajectory descriptors in a single video-level representation by accumulating their Fisher Vector encodings (FV) [147, 150], which was shown to be particularly effective for action recognition [27, 203]. This high-dimensional representation is based on the gradient of a generative model, a Gaussian Mixture Model (GMM), learned in an *unsupervised* manner on a large set of trajectory descriptors in our case. Given a GMM with k Gaussians, each parameterized by its mixture weight w_i, mean vector $\boldsymbol{\mu_i}$, and standard deviation vector $\boldsymbol{\sigma_i}$ (assuming a diagonal covariance), the FV encoding of a trajectory descriptor $\boldsymbol{x} \in \mathbb{R}^d$ is $\Phi(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \ldots, \phi_k(\boldsymbol{x})] \in \mathbb{R}^{2kd}$, where:

$$\phi_i(\boldsymbol{x}) = \left[\frac{\gamma(i)}{\sqrt{w_i}} \left(\frac{\boldsymbol{x} - \boldsymbol{\mu_i}}{\boldsymbol{\sigma_i}} \right), \frac{\gamma(i)}{\sqrt{2w_i}} \left(\frac{(\boldsymbol{x} - \boldsymbol{\mu_i})^2}{\boldsymbol{\sigma_i^2}} - 1 \right) \right] \tag{3.1}$$

and $\gamma(i)$ denotes the soft assignment of descriptor $\boldsymbol{x}$ to the i-th component Gaussian distribution. We use $k = 256$ Gaussians as a good compromise between accuracy and efficiency [202, 203, 204]. We randomly sample 256,000 trajectories from the pool of training videos, irrespectively of their labels, to learn one GMM per descriptor channel using 10 iterations of Expectation-Maximization (EM). Before learning the GMMs, we apply PCA to the descriptors, reducing their dimensionality by a factor of two. The reduced dimensions for the Traj, HOG, HOF, MBHx, and MBHy descriptors therefore are, respectively, 15, 48, 54, 48, and 48. After learning the GMMs, we extract FV encodings for all descriptors in each descriptor channel and combine these encodings into a per-channel, video-level representation using sum-pooling, *i.e.* by adding FVs together before normalization. In addition, we apply further post-processing and normalization steps, as discussed in the next subsection.

Supervised classification. When using a linear classification model, we use a linear SVM. As it is standard practice and in order to ensure comparability with previous works [101, 204, 205, 207], we fix $C = 100$ unless stated otherwise and use *one-vs-rest* for multi-class and multi-label classification. This forms a strong baseline for action recognition, as shown by previous works [203, 207] and confirmed in our experiments. We will now show how to make this baseline competitive with recent state-of-the-art methods.

3.2.2 Bag of tricks for Bag-of-Words

One of the reasons for the resilience of BoW-based methods was the steady increase in the number of improvements to the BoW pipeline proposed since its conception [27, 28, 101, 140, 150, 207].

Incorporating global spatiotemporal structure. Incorporating the spatiotemporal position of local features can improve the FV representation. We do not use Spatio-Temporal Pyramids (STP) [104], as they significantly increase both the dimensionality of the representation and its variance [162]. Instead, we simply concatenate the PCA-transformed descriptors with their respective $(x, y, t) \in \mathbb{R}^3$ coordinates, as in [101, 162]. We referred to this method in Section 2.2.1 as Spatio-Temporal Augmentation (STA). This approach is linked to the Spatial Fisher Vector (SFV) [96], a compact model related to soft-assign pyramids, in which the descriptor generative model is extended to explicitly accommodate the (x, y, t) coordinates of the local descriptors. When the SFV is created using Gaussian spatial models (*cf.* Equation 18 in [96]), the model becomes equivalent to a GMM created from augmented descriptors (assuming diagonal covariance matrices). Using STA, the dimensions for the descriptors before GMM estimation become 18, 51, 57, 51, 51. With 256 Gaussians, the dimension of the FVs generated for each descriptor channel are, accordingly, 9,216, 26,112, 29,184, 26,112, and 26,112.

Normalization. Another important detail to obtain maximum performance with bag-of-words models is their normalization. We apply signed-square-rooting followed by ℓ_2 normalization, then concatenate all descriptor-specific FVs and reapply this same normalization, following [101]. The double normalization re-applies square rooting, and is thus similar to using a smaller power normalization [150], which improves action recognition performance [125].

Multi-Skip Feature Stacking (MIFS). MIFS [101] improves the robustness of FV to videos of different lengths by increasing the pool of features with frame-skipped versions of the same video. Standard iDT features are extracted from those frame-skipped versions and stacked together before descriptor encoding, decreasing the expectation and variance of the condition number [17, 101, 153] of the extracted feature matrices. We will now see that the mechanics of this technique can be expanded to other transformations.

3.2.3 Data Augmentation by Feature Stacking (DAFS)

Data augmentation is an important part of deep learning [28, 95, 207], but it is rarely used with handcrafted features and shallow classifiers, particularly for action recognition where duplicating training examples can vastly increase the computational cost. Common data augmentation techniques for images include the use of random horizontal flipping [28, 207], random cropping [28], and even automatically determined transformations [139]. For video classification, [51, 75] duplicate the training set by mirroring.

Instead, we propose to generalize MIFS to arbitrary transformations, an approach we call *Data Augmentation by Feature Stacking* (DAFS). First, we extract features from multiple transformations of an input video (frame-skipping, mirroring, etc.) that do not change its semantic category. Second, we obtain a large feature matrix by stacking the obtained spatiotemporal features prior to encoding. Third, we encode the feature matrix, pool the resulted encodings, and apply the aforementioned normalization steps along this pipeline to obtain a *single augmented video-level representation*. This approach yields a representation that simplifies the learning problem, as it can improve the condition number of the feature matrix further than just MIFS thanks to leveraging data augmentation techniques traditionally used for deep learning. In contrast to data augmentation for deep approaches, however, we build a single more robust and useful representation instead of duplicating training examples. Note also that DAFS is particularly suited to FV-based representation of videos as pooling FV from a much larger set of features decreases one of the sources of variance for FV [16]. After concatenation, the final representation for each video has exactly 116,736 dimensions.

3.3 Hybrid architectures for action recognition

In this section we describe our hybrid architectures and how they can be learned.

3.3.1 System architecture

Our hybrid action recognition model combining FV with neural networks (*cf.* Figure 3.1) starts with the previously described steps of our iDT-DAFS-FV pipeline, which can be seen as a set of *unsupervised layers*. The next part of our architecture consists of a set of l fully connected *supervised layers*, each comprising a dot-product followed by a non-linearity.

Let $\boldsymbol{h_0}$ denote the FV output from the last unsupervised layer in our hybrid architecture, $\boldsymbol{h_{j-1}}$ the input of layer $j \in \{1, ..., l\}$, $\boldsymbol{h_j} = g(\boldsymbol{W_j} \boldsymbol{h_{j-1}})$ its output, with $\boldsymbol{W_j}$ the corresponding parameter matrix to be learned. We omit the biases from our equations for better clarity. For intermediate hidden layers we use the Rectified Linear Unit (ReLU) non-linearity [124] for g. For the final output layer we use different non-linearity functions depending on the task. For multi-class classification over c classes, we use the softmax function $g(\boldsymbol{z}_i) = \exp(\boldsymbol{z}_i) / \sum_{j=1}^{c} exp(\boldsymbol{z}_j)$. For multi-label tasks we consider the sigmoid function $g(\boldsymbol{z}_i) = 1/(1 + exp(-\boldsymbol{z}_i))$.

Connecting the last unsupervised layer to the first supervised layer can result in a much higher number of weights in this section than in all other layers of the architecture. Since this might be an issue for small datasets due to the higher risk of overfitting, we study the impact of different ways to learn the weights of this *dimensionality reduction layer*: either with unsupervised learning (*e.g.,* using PCA as in [148]), or by learning a low-dimensional projection end-to-end with the next layers of the architecture.

3.3.2 Learning

The learning of our model is done in two stages. In the first stage, we learn the unsupervised layers (*cf.* Figure 3.1, left) to obtain strong representations of handcrafted features. In the second stage, we learn the supervised classification layers (*cf.* Figure 3.1, right) using modern optimization algorithms. Our middle layers can be learned either with or without supervision, and we provide experiments to evaluate the best strategy in multiple datasets.

Unsupervised layers. Our unsupervised layers are learned as described in Section 3.2.1. Namely, we learn one GMM of $k = 256$ Gaussians per descriptor channel using EM on a set of 256,000 trajectories randomly sampled from the pool of training videos.

Supervised layers. We use the standard cross-entropy between the network output $\hat{\boldsymbol{y}}$ and the ground-truth label vectors $\boldsymbol{y}$ as loss function. For multi-class classification problems, we minimize the categorical cross-entropy cost function over all n samples:

$$C_{cat}(\boldsymbol{y},\hat{\boldsymbol{y}}) = -\sum_{i=1}^{n}\sum_{j=1}^{c} \boldsymbol{y}_{ij}\, log(\hat{\boldsymbol{y}}_{ij}), \tag{3.2}$$

whereas for multi-label problems we minimize the binary cross-entropy:

$$C_{bin}(\boldsymbol{y},\hat{\boldsymbol{y}}) = -\sum_{i=1}^{n}\sum_{j=1}^{c} \boldsymbol{y}_{ij}\, log(\hat{\boldsymbol{y}}_{ij}) \; - \; (1-\boldsymbol{y}_{ij})\; log(1-\hat{\boldsymbol{y}}_{ij}). \tag{3.3}$$

Optimization. For parameter optimization we use the recently introduced Adam algorithm [94]. Since Adam automatically computes individual adaptive learning rates for the different parameters of our model, this alleviates the need for fine-tuning of the learning rate with a costly grid-search or similar methods.

Adam uses estimates of the first and second-order moments of the gradients in the update rule:

$$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \alpha \cdot \frac{m_t}{(1-\beta_1^t)\sqrt{\frac{v_t}{1-\beta_2^t}} + \epsilon} \tag{3.4}$$

where

$$\begin{array}{lcl} \boldsymbol{g}_t & \leftarrow & \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}_{t-1}) \\ \boldsymbol{m}_t & \leftarrow & \beta_1 \cdot \boldsymbol{m}_{t-1} + (1-\beta_1)\cdot \boldsymbol{g}_t \\ \boldsymbol{v}_t & \leftarrow & \beta_2 \cdot \boldsymbol{v}_{t-1} + (1-\beta_2)\cdot {\boldsymbol{g}_t}^2 \end{array} \tag{3.5}$$

and where $f(\boldsymbol{\theta})$ is the function with parameters $\boldsymbol{\theta}$ to be optimized, t is the index of the current iteration, $m_0 = 0$, $v_0 = 0$, and β_1^t and β_2^t denotes β_1 and β_2 to the power of t, respectively. We use the default values for its parameters $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ proposed in [94] and implemented in Keras [33].

Batch normalization and regularization. During learning, we use batch normalization (BN) [78] and dropout (DO) [175]. Each BN layer is placed immediately before the ReLU non-linearity and parametrized by two vectors γ and β learned alongside each fully-connected layer. Given a training set $\boldsymbol{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n}\}$ of n

training samples, the transformation learned by BN for each input vector $\boldsymbol{x} \in \boldsymbol{X}$ is given by:

$$BN(\boldsymbol{x};\gamma,\beta) = \gamma \frac{\boldsymbol{x} - \boldsymbol{\mu_B}}{\sqrt{\boldsymbol{\sigma}^2_{\boldsymbol{B}} + \epsilon}} + \beta \tag{3.6}$$

where

$$\boldsymbol{\mu_B} \leftarrow \tfrac{1}{n} \sum_{i=1}^{n} \boldsymbol{x_i}, \quad \text{and} \quad \boldsymbol{\sigma}^2_{\boldsymbol{B}} \leftarrow \tfrac{1}{n} \sum_{i=1}^{n} (\boldsymbol{x_i} - \boldsymbol{\mu}_B)^2. \tag{3.7}$$

Together with DO, the operation performed by hidden layer j can now be expressed as $\boldsymbol{h_j} = \boldsymbol{r} \odot g(BN(\boldsymbol{W_j h_{j-1}};\gamma_j,\beta_j))$, where $\boldsymbol{r}$ is a vector of Bernoulli-distributed variables with probability p and $\odot$ denotes the element-wise product. The last output layer is not affected by this modification. Finally, we always consider the same DO rate p for all layers; the same number of neurons in each supervised layer; and we always use Glorot [61] uniform initialization when initializing the network's weights.

Dimensionality reduction layer. When unsupervised, we fix the weights of the dimensionality reduction layer from the projection matrices learned by PCA dimensionality reduction followed by whitening and ℓ_2 normalization [148]. When it is supervised, it is treated as the first fully-connected layer, to which we apply BN and DO as with the rest of the supervised layers. To explain our initialization strategy for the unsupervised case, let us denote the set of n mean-centered d-dimensional FVs for each sample in our dataset as a matrix $\boldsymbol{X} \in \mathbb{R}^{d \times n}$. Recall that the goal of PCA projection is to find a $r \times d$ transformation matrix $\boldsymbol{P}$, where $r \leqslant d$ on the form $\boldsymbol{Z} = \boldsymbol{PX}$ such that the rows of $\boldsymbol{Z}$ are uncorrelated, and therefore its $d \times d$ scatter matrix $\boldsymbol{S} = \boldsymbol{ZZ}^t$ is diagonal. In its primal form, this can be accomplished by the diagonalization of the $d \times d$ covariance matrix $\boldsymbol{XX}^t$. However, when $n \ll d$ it can become computationally inefficient to compute $\boldsymbol{XX}^t$ explicitly. For this reason, we diagonalize the $n \times n$ Gram matrix $\boldsymbol{X}^t\boldsymbol{X}$ instead. By Eigendecomposition of $\boldsymbol{X}^t\boldsymbol{X} = \boldsymbol{V\Lambda V}^t$ we can take $\boldsymbol{P} = \boldsymbol{V}^t\boldsymbol{X}^t\boldsymbol{\Lambda}^{-1/2}$, which also diagonalizes the scatter matrix $\boldsymbol{S}$ but is more efficient to compute [14, 83]. To accommodate whitening, we set the weights of our first reduction layer to $\boldsymbol{W_1} = \boldsymbol{V}^t\boldsymbol{X}^t\boldsymbol{\Lambda}^{-1}\sqrt{n}$ and keep them fixed during training.

Bagging. Since our first unsupervised layers can be fixed, we can train ensemble models and average their predictions very efficiently for bagging purposes [112, 148, 226] by caching the output of the unsupervised layers and reusing it in the subsequent models.

3.4 Experiments

We first list the datasets used in our experiments, then provide a detailed analysis of the iDT-FV pipeline and our proposed improvements. Based on our observations, we then perform an ablative analysis of our proposed hybrid architecture. Finally, we study the transferability of our hybrid models, and compare to the state of the art.

3.4.1 Datasets

We use five publicly available and commonly used datasets for action recognition (described in detail in Section 2.1.1). We briefly recall their main characteristics and their evaluation protocols.

- The **Hollywood2** [114] dataset contains 1,707, distributed over 12 overlapping action classes. Results are reported using the mAP in a single train/test split.
- The **HMDB-51** [99] dataset contains 6,766 videos distributed over 51 distinct action categories. Results are reported using mAcc over three fixed splits [99].
- The **UCF-101** [173] dataset contains 13,320 video clips distributed over 101 distinct classes. Performance is measured as the mAcc on three fixed splits.
- The **Olympics** [129] dataset contains 783 videos of athletes performing 16 different sport actions. We report mAP over the provided train/test split.
- The **High-Five** [138] dataset contains 300 videos distributed over four different human interactions and a negative (no-interaction) class. We report mAP for the positive classes (mAP+) over two provided train/test splits.
- The **ActivityNet** [73] dataset (release 1.3) contains 19,994 untrimmed videos distributed over 200 activity classes. We report mAP for the validation set of this dataset. As this is a dataset contains very long, untrimmed action videos (*cf.* Section 2.1.1 and Figure 2.1) which result in an extremely large number of pixel trajectories per video, we subsample those trajectories, considering only 20% of the extracted features per video when using iDT.

3.4.2 Study of trajectory baselines for action recognition

Table 3.1 reports our results comparing the iDT baseline (Section 3.2.1), its improvements discussed in Section 3.2.2, and our proposed data augmentation strategy (Section 3.2.3).

Reproducibility. We first note that there are multiple differences between the iDT pipelines used across the literature (*cf.* Section 2.2.1 for a detailed analysis). While [204] applies RootSIFT only on HOG, HOF, and MBH, in [101] this normalization is also applied to the Traj descriptor. While [204] includes Traj in their pipeline, [203] omits it. Additionally, person bounding boxes are used to ignore human motions when doing camera motion compensation in [203], but are not publicly available for all datasets. Therefore, we reimplemented the main baselines and compare our results to the officially published ones. As shown in Table 3.1, we successfully reproduce the original iDT results from [204] and [205], as well as the MIFS results of [101].

Improvements of iDT. Table 3.1 shows that double-normalization (DN) alone improves performance over iDT on most datasets without the help of STA. We show that STA gives comparable results to SFV+STP, as hypothesized in section 3.2.2. Given that STA and DN are both beneficial for performance, we combine them with our own method.

Data Augmentation by Feature Stacking (DAFS). Although more sophisticated transformations can be used, we found that combining a limited number of simple transformations already allows to significantly improve the iDT-based methods in conjunction with the aforementioned improvements, as shown in the "iDT + STA + DAFS + DN" line of Table 3.1. In practice, we generate on-the-fly 7 different versions for each video, considering the possible combinations of frame-skipping up to level 3 and horizontal flipping. As an implementation detail, we use FFmpeg to generate those versions on-the-fly before extracting their feature matrices using iDT. After extraction, we vertically stack those matrices together, *e.g.,* forming a new matrix which has the same number of columns (descriptor dimensions) as the originals, but the total of their numbers of rows (number of trajectories). Our results are shown in the last row of Table 3.1.

Fine tuned and non-linear SVMs. Attempting to improve our best results, we also performed experiments both fine-tuning C and also using a Gaussian kernel while fine-tuning γ. However, we found that those two sets of experiments did not lead to significant improvements. As DAFS already brings results competitive with the current state of the art, we set those results with fixed C as our current shallow baseline (FV-SVM). We will now incorporate those techniques in the first unsupervised layers of our hybrid models.

Table 3.1 – Analysis of iDT baselines and several improvements.

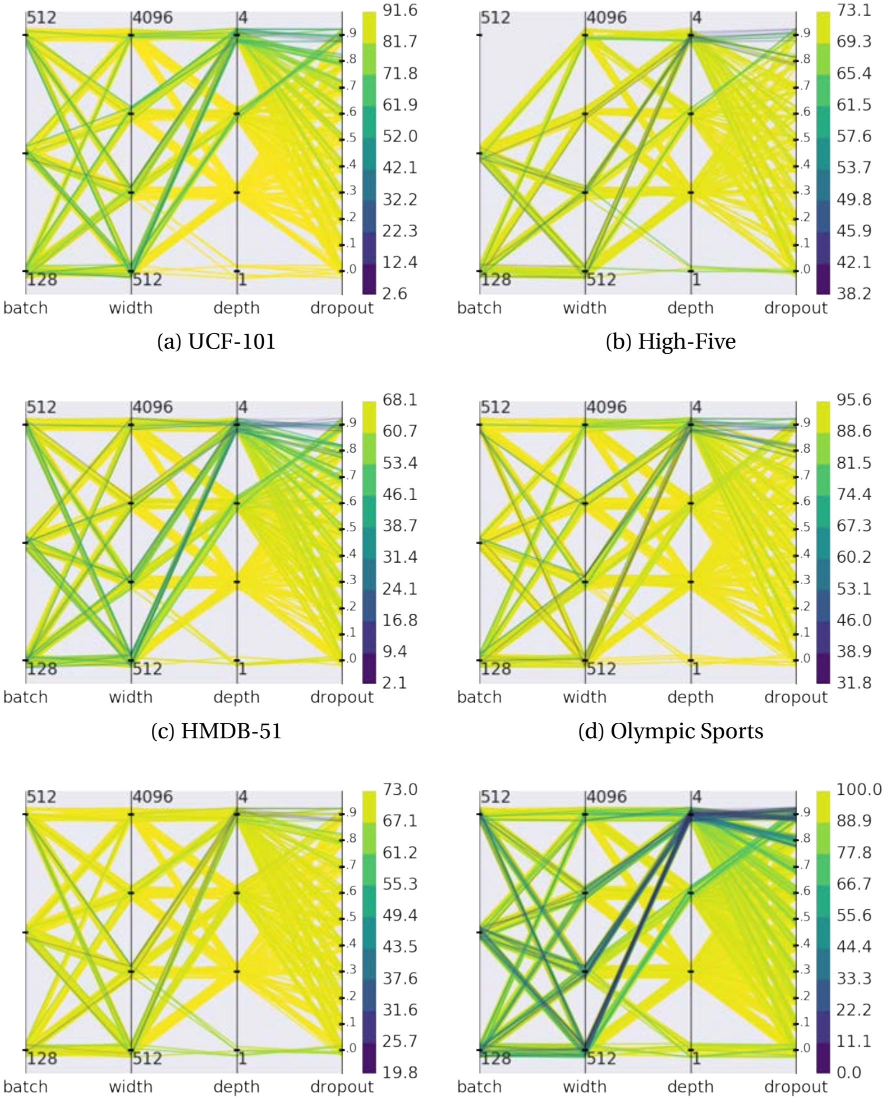
	UCF-101 %mAcc (s.d.)	HMDB-51 %mAcc (s.d.)	Hollywood2 %mAP	High-Five %mAP+ (s.d.)	Olympics %mAP	ActivityNet %mAP
iDT [204]	84.8 [205]*†	57.2	64.3	-	91.1	-
Our reproduction	85.0 (1.32)*†	57.0 (0.78)	64.2	67.7 (1.90)	88.6	60.90
iDT+SFV+STP [203]	85.7*†	60.1*	66.8*	68.1*†	90.4*	-
Our reproduction	85.4 (1.27)*†	59.3 (0.80)*	67.1*	67.8 (3.78)*†	88.3*	57.56
iDT+STA+DN [101]	87.3	62.1	67.0	-	89.8	-
Our reproduction	87.3 (0.96)†	61.7 (0.90)	66.8	70.4 (1.63)	90.7	62.36
iDT+STA+MIFS+DN [101]	89.1	65.1	68.0	-	91.4	-
Our reproduction	89.2 (1.03)†	65.4 (0.46)	67.1	70.3 (1.84)	91.1	64.12
iDT+DN	86.3 (0.95)†	59.1 (0.45)	65.7	67.5 (2.27)	89.5	62.27
iDT+STA	86.0 (1.14)†	60.3 (1.32)	66.8	70.4 (1.96)	88.2	60.94
iDT+HFDA [51]	86.4 (1.39)†	59.9 (0.96)	65.7	69.6 (1.58)	93.4	62.99
iDT+HFFS	86.7 (1.19)†	60.3 (0.46)	66.5	69.4 (1.90)	93.8	63.26
iDT+DAFS	88.6 (1.06)†	65.2 (0.99)	67.1	70.3 (1.93)	92.3	64.96
iDT+HFFS+DN	88.3 (1.23)†	62.9 (0.47)	67.5	69.2 (2.49)	93.2	64.83
iDT+STA+DAFS+DN	**90.6 (0.91)**†	**67.8 (0.22)**	**69.1**	**71.0 (2.46)**	**92.8**	**66.49**

iDT: Improved Dense Trajectories; SFV: Spatial Fisher Vector; STP: Spatio-Temporal Pyramids; STA: Spatio-Temporal Augmentation; MIFS: Multi-skIp Feature Stacking; DN: Double-Normalization; DAFS: Data Augmentation Feature Stacking; *without Trajectory descriptor; †without Human Detector.

3.4.3 Analysis of hybrid models

In this section, we start from hybrid architectures with unsupervised dimensionality reduction learned by PCA. For UCF-101 and ActivityNet (the largest datasets) we initialize $\boldsymbol{W}_1$ with $r = 4096$ dimensions, whereas for all other datasets we use the number of dimensions responsible for 99% of the variance (yielding less dimensions than training samples).

Brief explanation of Parallel Coordinate (PC) plots. PC plots are a visualization technique for displaying high-dimensional data in 2D. They can highlight meaningful multivariate patterns, especially when used interactively [52]. In a PC plot, each data dimension is associated with a vertical line crossing the x-axis. Plotting a single multi-dimensional data sample involves two steps: (i) placing each dimension on its related vertical line, then (ii) connecting these points with a line of the same color, thus creating a path that crosses all the y-axes. Each dimension is normalized to the unit interval before plotting. In our case, each data point is a quadruplet (*batch*, *width*, *depth*, *dropout*) coupled with a recognition performance number. Each hyper-parameter value has a position along the y-axis corresponding to it. We slightly shift that position randomly for better readability. The recognition performance of the path corresponding to a set of hyper-parameters is encoded in the color of the line (brighter is better) and its transparency (more solid is better).



(a) UCF-101

(b) High-Five

(c) HMDB-51

(d) Olympic Sports

(e) Hollywood2

(f) Combined (normalized)

Figure 3.2 – Parallel Coordinates plots showing the impact of multiple parameters on our hybrid architectures with unsupervised dimensionality reduction for each dataset. Each line represents one combination of parameters and colour indicates performance. Depth 2 correlates with high-performing architectures, whereas a small width and a large depth is suboptimal. The combined plot (f) has been generated by normalizing the performance in each dataset to the [0-100] interval and stacking the architectures tuples together (*cf.* explanatory text in Section 3.4.3).

Table 3.2 – Top-5 best performing hybrid architectures with consistent improvements across multiple datasets.

Depth	Width	Batch	UCF-101 %mAcc	HMDB-51 %mAcc	Hollywood2 %mAP	High-Five %mAP+	Olympics %mAP	ActivityNet % mAP	Relative Improv.
2	**4096**	**128**	91.6	68.1	72.6	**73.1**	**95.3**	68.3	**2.46%**
2	4096	256	91.6	67.8	72.5	72.9	95.3	**68.6**	2.27%
2	2048	128	91.5	68.0	72.7	72.7	94.8	68.3	2.21%
2	2048	256	91.4	67.9	72.7	72.5	95.0	68.5	2.18%
2	512	128	91.0	67.4	**73.0**	72.4	95.3	68.3	2.05%
1	-	-	**91.9**	**68.5**	70.4	71.9	93.5	65.6	1.28%
Best FV-SVM (*cf.* Tab. 3.1)			90.6	67.8	69.1	71.0	92.8	66.5	0.00%

Ablative analysis. We study the interactions between four parameters that can influence the performance of our hybrid models: the output dimension of the intermediate fully connected layers (*width*), the number of layers (*depth*), the dropout rate, and the mini-batch size of Adam (*batch*). This gives us a total of *480* architectures *for each dataset*, corresponding to three batch sizes $(128, 256, 512)$, four widths $(512, 1024, 2048, 4096)$, four depths $(1, 2, 3, 4)$ and ten dropout rates $(0.0, 0.1, \ldots, 0.9)$. We systematically evaluate all possible combinations and rank the architectures by the average relative improvement *w.r.t.* the best FV-SVM model. Training all 480 combinations for one split of UCF-101 can be accomplished in less than two days with a single Tesla K80 GPU. We report the top results in Table 3.2 and present per-dataset parallel coordinates plots showing all the explored hybrid architectures with *unsupervised* dimensionality reduction in Figure 3.2. Our observations are as follows.

Unsupervised dimensionality reduction. Performing dimensionality reduction using the weight matrix from PCA is beneficial for all datasets, and using this layer alone, achieves 1.28% average improvement (Table 3.2, depth 1) upon our best SVM baseline.

Width. We consider networks with fully connected layers of size 512, 1024, 2048, and 4096. We find that a large width (4096) gives the best results in 5 of 6 datasets.

Depth. We consider hybrid architectures with depth between 1 and 4. Most well-performing models have depth 2 as shown in Figure 3.2, but one layer is enough for big datasets of short video clips.

Dropout rate. We consider dropout rates from 0 to 0.9. We find dropout to be dependent of both architecture and dataset. A high dropout rate significantly impairs classification results when combined with a small width and a large depth.

Mini-batch size. We consider mini-batch sizes of 128, 256, and 512. We find lower batch sizes to bring best results, with 128 being the more consistent across all datasets. We observed that large batch sizes were detrimental to networks with a small width.

Visual analysis. All PC plots share a strong visible pattern associating an elevated level of brightness with networks of depth 2, indicating a strong relationship between this choice of depth and the best results. Another visible pattern is the relationship between lower widths and higher depths, which is always marked by the presence of suboptimal architectures (dark lines from width 512 and 1024 to depth 4). Other patterns associated with suboptimal architectures appear between large batch sizes and a small width (dark lines from batch 512 to width 512). On the other hand, connections between large widths and depth 2 are always clear in the plots for all datasets. This pattern supports the best architecture found by our systematic ranking of the architectures discussed above: batch size of 128, width of 4096, and depth 2. Regarding the dropout rate, the PC plots in Figure 3.2 suggest that too high rates might be detrimental to the model, and the rate should be tuned depending on the rest of the architecture and on the dataset.

Best configuration with unsupervised dimensionality reduction. We find the following parameters to work the best: small batch sizes, a large width, moderate depth, and dataset-dependent dropout rates. The most consistent improvements across datasets are with a network with batch-size 128, width 4096, and depth 2.

Supervised dimensionality reduction. Our previous findings indicate that the dimensionality reduction layer can have a large influence on the overall classification results. Therefore, we investigate whether a *supervised* dimensionality reduction layer trained *mid-to-end* with the rest of the architecture could improve results further. Due to memory limitations imposed by the higher number of weights to be learned between our 116K-dimensional input FV representation and the intermediate fully-connected layers, we decrease the maximum network width to 1024. In spite of this limitation, our results in Table 3.3 show that much smaller hybrid architectures with supervised dimensionality reduction improve (on the larger UCF-101, HMDB-51 and ActivityNet datasets) or maintain (on the other smaller datasets) recognition performance.

Table 3.3 – Supervised dimensionality reduction hybrid architecture evaluation.

			UCF-101	HMDB-51	Hollywood2	High-Five	Olympics	ActivityNet
Depth	Width	Batch	%mAcc (s.d.)	%mAcc (s.d.)	%mAP	%mAP+ (s.d.)	%mAP	%mAP
1	1024	128	92.3 (0.77)	**69.4 (0.16)**	72.5	71.8 (1.37)	95.2	69.5
1	512	128	**92.3 (0.70)**	69.2 (0.09)	72.2	72.2 (1.14)	95.2	**70.2**
2	1024	128	91.9 (0.78)	68.8 (0.46)	71.8	72.0 (1.03)	94.8	68.1
2	512	128	92.1 (0.68)	69.1 (0.36)	70.8	71.9 (2.22)	94.2	67.3
Best unsup. (*cf.* Tab. 3.2)			91.9	68.5	**73.0**	**73.1**	**95.3**	68.6

Comparison to hybrid models for image recognition. Our experimental conclusions and optimal model differ from [148], both on unsupervised and supervised learning details (*e.g.*, dropout rate, batch size, learning algorithm), and in the usefulness of a supervised dimensionality reduction layer trained mid-to-end (not explored in [148]).

3.4.4 Transferability of hybrid models

In this section, we study whether the first layers of our architecture can be transferred across datasets. As reference points, we use the first split of UCF-101 and the training set of ActivityNet to create base hybrid models. We then transfer elements from those models to models created for other datasets. We chose UCF-101 and ActivityNet for the following reasons: Both UCF-101 and ActivityNet are the largest datasets we consider with a similar number of action videos, have the largest diversity in number of actions, and contain multiple categories of actions, including human-object interaction, human-human interaction, body-motion interaction, and practicing sports. However, a crucial difference between those is that UCF-101 contains short trimmed action videos, whereas ActivityNet videos are untrimmed and extremely long in comparison with the other datasets. We use this opportunity to study how this difference impacts the transferability of our learned features.

Unsupervised representation layers. We start by replacing the dataset-specific GMMs with the GMMs from the base models. Our results in the second row of Table 3.4 show that the GMMs transferred from UCF-101 give similar performance to the ones using dataset-specific GMMs. This, therefore, greatly simplifies the task of learning a new model for a new dataset. We keep the transferred GMMs fixed in the next experiments. When transferring GMMs from ActivityNet (second row of Table 3.5) we see that those features do not transfer as well as when transferring from a short-clip dataset, except for the smallest dataset considered (High-Five).

Table 3.4 – Transferability experiments from a dataset of short action clips (UCF-101) involving unsupervised dimensionality reduction

Representation	Reduction	Supervised	HMDB-51	Hollywood2	High-Five	Olympics
Layers	Layer	Layers	%mAcc (s.d.)	%mAP	%mAP+ (s.d.)	%mAP
own	own	own	68.0 (0.65)	**72.6**	73.1 (1.01)	95.3
UCF	own	own	**68.0 (0.40)**	72.4	73.7 (1.76)	94.2
UCF	UCF	own	66.5 (0.88)	70.0	**76.3 (0.96)**	94.0
UCF	UCF	UCF	66.8 (0.36)	69.7	71.8 (0.12)	**96.0**

Table 3.5 – Transferability experiments from a dataset of long untrimmed videos (ActivityNet) involving unsupervised dimensionality reduction

Representation	Reduction	Supervised	UCF101	HMDB-51	Hollywood2	High-Five	Olympics
Layers	Layer	Layers	%mAcc (s.d.)	%mAcc (s.d.)	%mAP	%mAP+ (s.d.)	%mAP
own	own	own	**91.6 (0.68)**	**68.0 (0.65)**	**72.6**	73.1 (1.01)	**95.3**
ActivityNet	own	own	90.5 (1.03)	67.7 (0.26)	70.8	**74.3 (0.18)**	94.3
ActivityNet	ActivityNet	own	90.4 (1.15)	66.2 (0.20)	71.7	70.6 (1.90)	94.6
ActivityNet	ActivityNet	ActivityNet	87.3 (0.91)	66.3 (0.24)	63.5	71.4 (5.67)	94.1

Unsupervised dimensionality reduction layer. Instead of configuring the unsupervised dimensionality reduction layer with weights from the PCA learned on its own dataset, we configure it with the weights learned in UCF-101 and ActivityNet. Our results are in the third row of Tables 3.4 and 3.5. This time we observe a different behavior when transferring for UCF-101: for Hollywood2 and HMDB-51, the best models were found without transfer, whereas for Olympics it did not have any measurable impact. However, transferring PCA weights brings significant improvement in High-Five. One of the reasons for this improvement is the evidently smaller training set size of High-Five (150 samples) in contrast to other datasets. The fact that the improvement becomes less visible as the number of samples in each dataset increases (before eventually degrading performance) indicates there is a threshold below which transferring starts to be beneficial (around a few hundred training videos). On the other hand, when transferring from ActivityNet we found a similar behavior as before: Transferring the dimensionality reduction weights learned from a dataset of long, untrimmed videos did not bring any performance improvements for short-clip datasets.

Supervised layers after unsupervised reduction. We also study the transferability of further layers in our architecture, after the unsupervised dimensionality reduction transfer. We take our base models learned in the first split of UCF-101

Table 3.6 – Transferability experiments from a dataset of short action clips (UCF-101) involving supervised dimensionality reduction.

Representation Layers	Supervised Layers	HMDB-51 %mAcc (s.d.)	Hollywood2 %mAP	High-Five %mAP+ (s.d.)	Olympics %mAP
own	own	69.2 (0.09)	72.2	72.2 (1.14)	95.2
UCF	own	69.4 (0.16)	**72.5**	71.8 (1.37)	95.2
UCF	UCF	**69.6 (0.36)**	72.2	**73.2 (1.89)**	**96.3**

Table 3.7 – Transferability experiments from a dataset of long untrimmed videos (ActivityNet) involving supervised dimensionality reduction.

Representation Layers	Supervised Layers	UCF-101 %mAcc (s.d.)	HMDB-51 %mAcc (s.d.)	Hollywood2 %mAP	High-Five %mAP+ (s.d.)	Olympics %mAP
own	own	**92.3 (0.70)**	**69.2 (0.09)**	**72.2**	**72.2** (1.14)	**95.2**
ActivityNet	own	90.4 (1.02)	67.3 (0.64)	71.4	70.9 (2.03)	94.2
ActivityNet	ActivityNet	90.9 (1.00)	66.8 (0.79)	70.4	69.9 (1.26)	94.8

and ActivityNet, remove its last classification layer, re-insert a classification layer with the same number of classes as the target dataset, and fine-tune this new model in the target dataset, using an order of magnitude lower learning rate. The results can be seen in the last row of Tables 3.4 and 3.5. When transferring from UCF-101, we observe the same behavior for HMDB-51 and Hollywood2. However, we notice a decrease in performance for High-Five and a performance increase for Olympics. We attribute this to the presence of many sports-related classes in UCF-101. We again do not observe any gain in performance when transferring from ActivityNet.

Mid-to-end reduction and supervised layers. Finally, we study whether the architecture with supervised dimensionality reduction layer transfers across datasets, as we did for the unsupervised layers. We again replace the last classification layer from the corresponding model learned on the first split of UCF-101, and fine-tune the whole architecture on the target dataset. Our results in the second and third rows of Table 3.6 show that transferring this architecture from UCF-101 brings improvements for Olympics and HMDB-51, but performs worse than transferring unsupervised layers only on High-Five. Transferring from ActivityNet is again detrimental to performance (Table 3.7). This gives further indication that the length and specificity of the action videos contained in the source dataset is of major importance when attempting feature transfer.

Table 3.8 – Comparison against the state of the art* in action recognition.

	Method	UCF-101 %mAcc (s.d.)	HMDB-51 %mAcc (s.d.)	Hollywood2 %mAP	High-Five %mAP+ (s.d.)	Olympics %mAP	ActivityNet % mAP
HANDCRAFTED	iDT+FV [204]	84.8 [205]	57.2	64.3	-	91.1	
	SDT-ATEP [56]	-	41.3	54.4	62.4	85.5	
	iDT+FM [140]	87.9	61.1	-	-	-	
	RCS [75]	-	-	73.6	71.1	-	
	iDT+SFV+STP [203]	86.0	60.1	66.8	69.4	90.4	
	iDT+MIFS [101]	89.1	65.1	68.0	-	91.4	
	VideoDarwin [51]	-	61.6	69.6	-	-	
	VideoDarwin+HF+iDT [51]	-	63.7	**73.7**	-	-	
DEEP-BASED	2S-CNN [170]$^{\text{IN}}$	88.0	59.4	-	-	-	
	2S-CNN+Pool [127]$^{\text{IN}}$	88.2	-	-	-	-	
	2S-CNN+LSTM [127]$^{\text{IN}}$	88.6	-	-	-	-	
	Objects+Motion(R*) [81]$^{\text{IN}}$	88.5	61.4	66.4	-	-	
	Comp-LSTM [176]$^{\text{ID}}$	84.3	44.0	-	-	-	
	C3D+SVM [187]$^{\text{S1M,ID}}$	85.2	-	-	-	-	
	FSTCN [182]$^{\text{IN}}$	88.1	59.1	-	-	-	
HYBRID	iDT+StackFV [141]	-	66.8	-	-	-	
	TDD [207]$^{\text{IN}}$	90.3	63.2	-	-	-	
	TDD+iDT [207]$^{\text{IN}}$	91.5	65.9	-	-	-	
	CNN-hid6 [221]$^{\text{S1M}}$	79.3	-	-	-	-	
	CNN-hid6+iDT [221]$^{\text{S1M}}$	89.6	-	-	-	-	
	C3D+iDT+SVM [187]$^{\text{S1M,ID}}$	90.4	-	-	-	-	
	Best from state of the art	91.5 [207]	66.8 [141]	**73.7** [51]	71.1 [75]	91.4 [101]	-
	Our best FV+SVM	90.6 (0.91)	67.8 (0.22)	69.1	71.0 (2.46)	92.8	66.5
	Our best hybrid	**92.5 (0.73)**	**70.4 (0.97)**	72.6	**76.7 (0.39)**	**96.7**	72.5

*State of the art at the time the work in this chapter was published in ECCV'16. Methods are organized by category (*cf.* Table 2.4) and sorted in chronological order in each block. Our hybrid models improve upon the state of the art, and our handcrafted-shallow FV-SVM improves upon competing end-to-end architectures relying on external data sources (IN: uses ImageNet. S1M: uses Sports-1M. ID: uses private internal data – [187] pre-trains models on an internal dataset referred to as I380K, whereas [176] uses additional 300h of unrelated Youtube videos).

3.4.5 Comparison to the state of the art

In this section, we compare our best models found previously to the state of the art.

Best models. For UCF-101, the most effective model leverages its large training set using supervised dimensionality reduction (*cf.* Table 3.3). For HMDB-51 and Olympics, the best models result from transferring the supervised dimensionality reduction models from the related UCF-101 dataset (*cf.* Table 3.6). Due to its specificity, the best architecture for Hollywood2 is based on unsupervised dimensionality reduction learned on its own data (*cf.* Table 3.2), although there are similarly-performing end-to-end transferred models (*cf.* Table 3.6). For High-Five, the best model is obtained by transferring the unsupervised dimensionality reduction models from UCF-101 (*cf.* Table 3.4).

Bagging. As it is standard practice [148], we take the best models and perform bagging with 8 models initialized with distinct random initializations. This improves results by around one point on average. We show our final results in the last row of Table 3.8.

Discussion. In contrast to [148], our models outperform the state of the art at the time of publication of this work, including methods trained on massive labeled datasets like ImageNet or Sports-1M, confirming both the excellent performance and the data efficiency of our approach. As shown in Figure 3.3, our method leads to substantial improvements for all datasets considered. Table 3.9 illustrates some failure cases of our methods. We provide confusion matrices and precision-recall curves for the all short-clip datasets evaluated in this work in the next section, for fine-grained analysis.

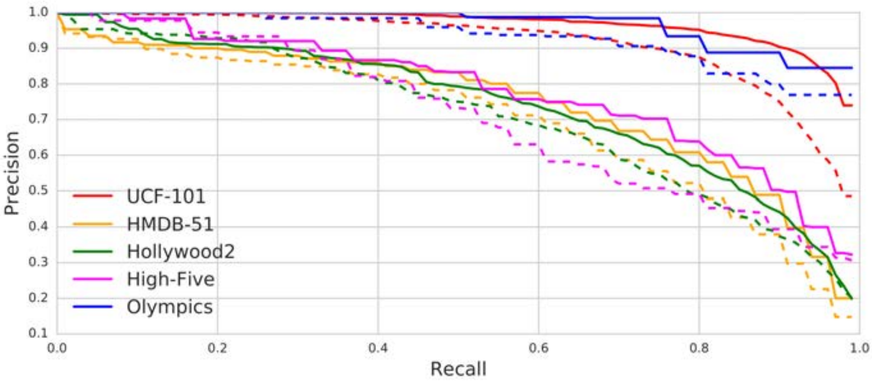


Figure 3.3 – Average Precision-Recall curves. Dashed line presents curves for our best FV-SVM baseline, and full lines presents curves for our best hybrid model.

Table 3.9 – Top-5 most confused classes for our best FV-SVM and Hybrid models

Top-5 Confusions		#1	#2	#3	#4	#5
UCF-101	Hybrid	 P: ShavingBeard GT: BrushingTeeth	 P: FrisbeeCatch GT: LongJump	 P: ApplyLipstick GT: ShavingBeard	 P: Nunchucks GT: PizzaTossing	 P: Rafting GT: Kayaking
	FV-SVM	 P: BreastStroke GT: FrontCrawl	 P: ShavingBeard GT: BrushingTeeth	 P: Rafting GT: Kayaking	 P: FrisbeeCatch GT: LongJump	 P: ApplyLipstick GT: ShavingBeard
HMDB-51	Hybrid	 P: sword GT: punch	 P: flic_flac GT: cartwheel	 P: draw_sword GT: sword_exercise	 P: sword_exercise GT: draw_sword	 P: drink GT: eat
	FV-SVM	 P: sword GT: punch	 P: sword_exercise GT: draw_sword	 P: chew GT: smile	 P: throw GT: swing_baseball	 P: fencing GT: sword
High-Five	Hybrid	 P: negative GT: highFive	 P: negative GT: handShake	 P: hug GT: handShake	 P: hug GT: kiss	 P: handShake GT: highFive
	FV-SVM	 P: negative GT: highFive	 P: negative GT: handShake	 P: hug GT: kiss	 P: hug GT: handShake	 P: negative GT: kiss
Olympics	Hybrid	 P: long_jump GT: triple_jump	 P: clean_and_jerk GT: snatch	 P: high_jump GT: vault	 P: discus_throw GT: hammer_throw	 P: vault GT: high_jump
	FV-SVM	 P: vault GT: high_jump	 P: long_jump GT: triple_jump	 P: discus_throw GT: hammer_throw	 P: pole_vault GT: high_jump	 P: bowling GT: shot_put

3.4.6 Detailed failure and success cases per dataset

In this section we provide a detailed analysis for failure and success cases for four of our considered datasets, contrasting our best hybrid models against our strong FV-SVM baseline.

The datasets used in our experiments can be divided according to two criteria: the presence of overlapping classes (multi-label or multi-class) and performance metric (mean Average Precision, mAP; or mean accuracy, mAcc). We therefore present precision-recall curves for datasets whose performance is computed using mAP (Hollywood2, High-Five, Olympics) and confusion matrices for datasets which are multi-class (UCF-101, HMDB-51, Olympics, High-Five). We cluster the rows and columns of the confusion matrices by level of confusion, which we obtain by ordering the rows via a 1D Locally Linear Embedding (LLE) [159]. For the largest datasets, we show only the top-25 confusion classes as determined by this embedding. For the precision-recall curves, we show at every recall decile the quantized cumulated precision segmented per class.

UCF-101. We first determine the top 25 classes that are responsible for most of the confusion of our baseline model (FV-SVM) using the first split of this dataset. Then, we show the evolution of those same classes in the confusion matrix of our best hybrid model (Hybrid) in Figure 3.4. Note that the differences are small overall, as both the FV-SVM and Hybrid models have excellent recognition performance (90.6% and 92.5% respectively, *cf.* Table 3.9).

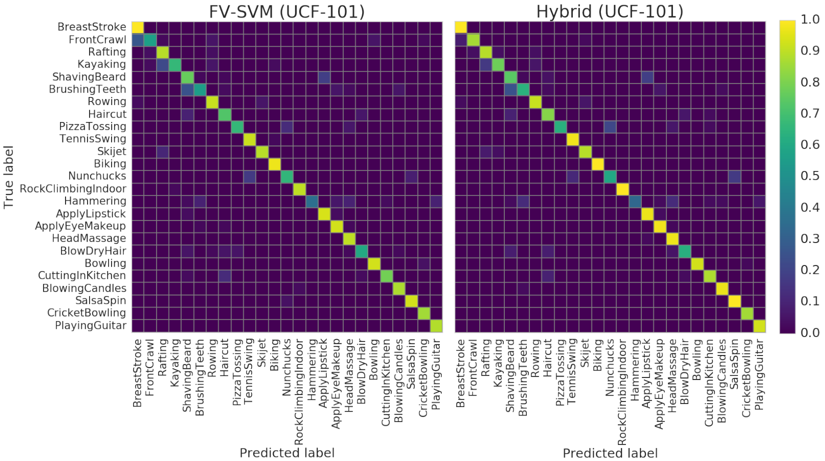


Figure 3.4 – Top-25 most confused classes for UCF-101.

The most confused classes are *BreastStroke, FrontCrawl, Rafting, Kayaking, ShavingBeard* and *Brushing Teeth*. Inspecting both confusion matrices, we can see how the Hybrid model solves most of the confusion between the *BreastStroke* and *FrontCrawl* swimming actions. It also solves a visible confusion between *FrontCrawl* and *Rowing*. Furthermore, it also improves many other classes, such as *ApplyLipstick, ApplyEyeMakeUp*, and *HeadMassage*, but seems to reinforce certain mistakes such as the confusion between *Nunchucks* and *SalsaSpin*. Our results therefore suggest that our higher-capacity hybrid models can improve on some fine-grained recognition tasks, but not all (*i.e. Rafting* and *Kayaking*), which confirms the previously known good performance of FV and linear classifiers for fine-grained tasks [63].

HMDB-51. Using split 1, we generate the confusion matrices using the same method as before and show the evolution of the top 25 classes in Figure 3.5. Most of the confusion in this dataset originates from the classes *punch, draw_sword, fencing, shoot_bow, sword_exercise* and *sword*. While our Hybrid model significantly improves the mean accuracy for this dataset (+2.6%, *cf.* Table 8 in the main text), the improvements are well distributed across all classes and are less visible when inspecting individual class pairs. Some improvements are between the classes *sword_exercise* and *shoot_bow, kick_ball* and *catch, wave* and *sword_exercise*.

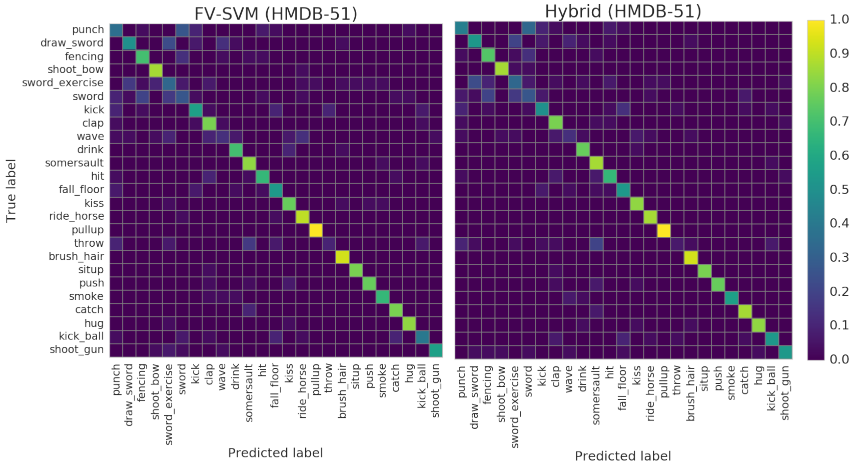


Figure 3.5 – Top-25 most confused classes for HMDB-51.

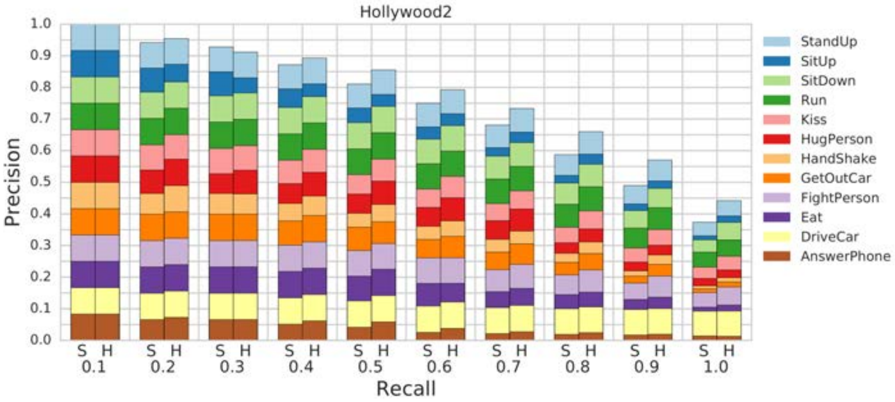


Figure 3.6 – Quantized precision and recall segmented per class for the Hollywood2 dataset, comparing class-specific precision between our baseline model (S) and the best hybrid model (H) at different recall rates.

Hollywood2. As this is a multi-label dataset, we present a cumulative quantized precision-recall bar chart segmented per-class in Figure 3.6. This visualization can be understood as a quantized version of the precision-recall curve, but allows us to identify the separate influence of each class in the performance result. We can see from the figure how the stacked bars corresponding to the Hybrid model (H) associated with higher recall rates present higher precision than the baseline (S). The difference in performance comes mostly from the *GetOutCar* and *HandShake* classes, whose precision improves at higher recall rates for the Hybrid model.

Olympics. Olympics is multi-class and evaluated with mAP, so we present both confusion matrices (Figure 3.7) and per-class quantized precision-recall bar charts (Figure 3.8). Most of the confusion originates from the *long_jump*, *triple_jump*, *vault*, and *high_jump* classes. Our hybrid classifier is able to solve the strong confusion between *high_jump*, *vault*, and *pole_vault*, as well as *vault* and *platform_10m*. However, the Hybrid model has trouble distinguishing between *long_jump* and *triple_jump*, the smallest class in the dataset (17 videos for training, only 4 for testing). It has therefore low impact on mean accuracy: the overall performance in this dataset is indeed significantly improved (+3.9% *w.r.t.* FV-SVM and +5.3% *w.r.t.* the state of the art, *cf.* Table 3.8 in the main text).

Figure 3.8 shows that both methods have excellent precision at low recall, with

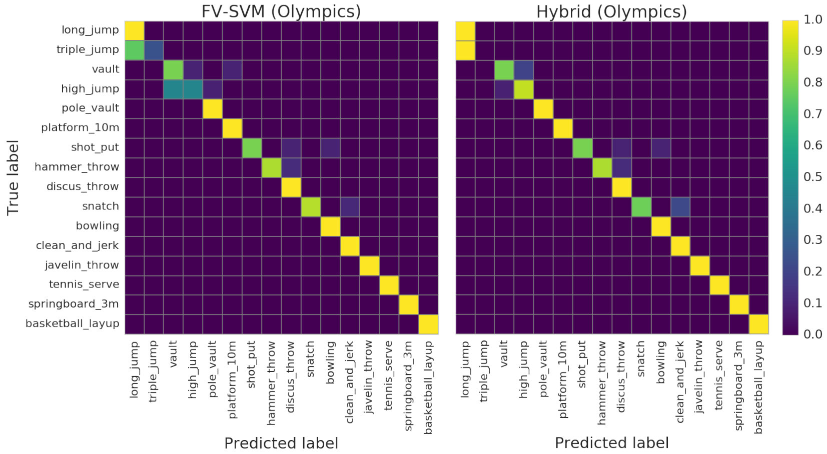


Figure 3.7 – Confusion matrices for Olympic Sports.

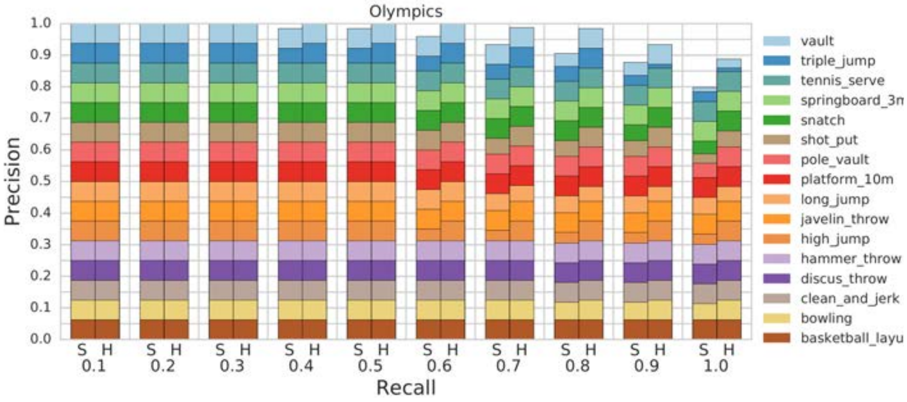


Figure 3.8 – Quantized precision-recall segmented per class for Olympics.

the performance between the two methods starting to differ after a recall of 40%. For higher recall rates, we can see how most of the differences stem indeed from the *triple_jump* class. The decrease in area for segments of this class are compensated by steady improvements in all other classes, as can be seen by comparing the class-specific segments between the baseline (S) and Hybrid (H) results for the 0.9 and 1.0 recall rates. This suggest that the majority of the improvements remaining are for this single class.

High-Five High-Five is the smallest dataset (and among the most fine-grained) we experiment on. As it is multi-class and evaluated using mAP, we present confusion matrices (Figure 3.9) and per-class quantized precision-recall curves (Figure 3.10) on its first cross-validation split. Most of the confusion is between *kiss vs. hug*, and *handShake vs. hug*. The Hybrid model improves the disambiguation between *kiss* and *hug*, and *kiss* and *highFive*, but introduces some confusion between *highFive* and *handShake*, and *kiss* and *handShake*. Figure 3.10 shows that our Hybrid model is more precise at high recall rates, especially for *kiss* and *handShake*. Moderate improvements can be observed for all classes.

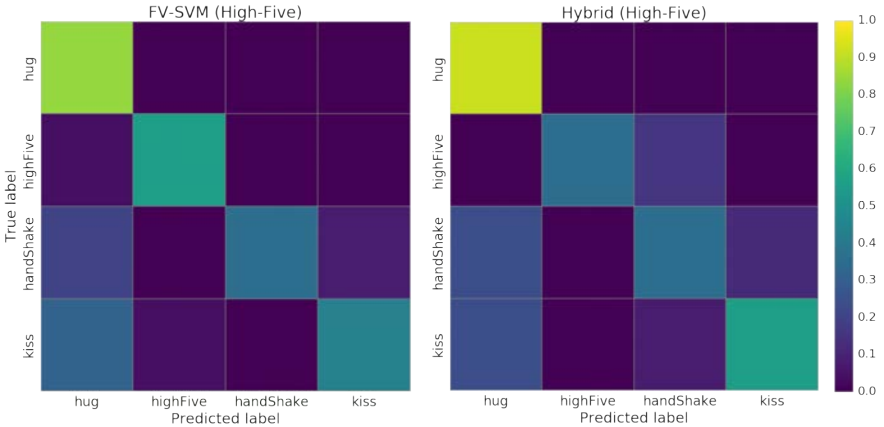


Figure 3.9 – Confusion matrices for the first cross-validation split of High-Five.

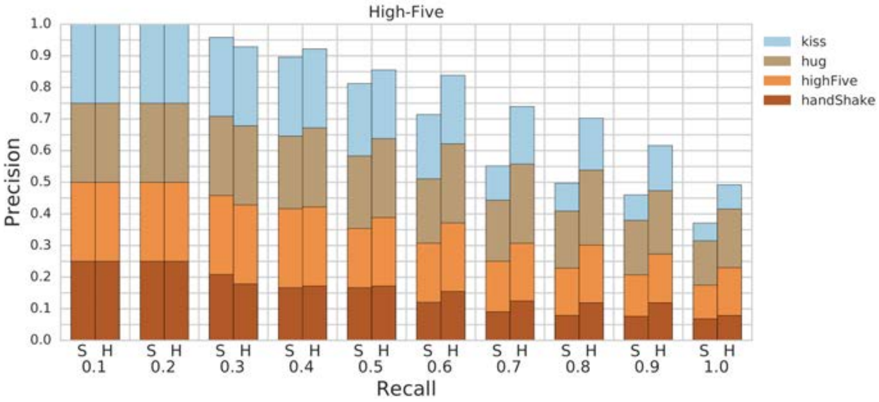


Figure 3.10 – Quantized precision-recall segmented per class for High-Five split 1.

3.5 Summary of the chapter

In this chapter, we investigated hybrid architectures for action recognition, effectively combining handcrafted spatiotemporal features, unsupervised representation learning based on the FV encoding, and deep neural networks. In addition to paying attention to important details like normalization and spatiotemporal structure, we integrate data augmentation at the feature level, end-to-end supervised dimensionality reduction, and modern optimization and regularization techniques. We performed an extensive experimental analysis on a variety of datasets, showing that our hybrid architecture yields data efficient, transferable models of small size that yet outperform much more complex deep architectures trained end-to-end on millions of images and videos.

Our detailed per-dataset analysis suggests that the improvements brought by our hybrid models are generally distributed across the classes of most datasets, without a single class being responsible for most of the performance increase. We also show that our hybrid models can differentiate between some fine-grained action groups, confirming that the unsupervised video-level FV representation contains fine-grained information about the original video, an information that may be more successfully exploited by our more complex (deeper) hybrid models.

4 Procedural Human Action Videos Dataset

> Obviously, with a CGI character, you're building a character in much the same way as a real creature is built. You build the bones, the skeletons, the muscles. You put layers of fat on. You put a layer of skin on which has to have a translucency, depending on what the character is.
>
> Peter Jackson

In the previous chapter, we have seen how to design data-efficient models by leveraging existing prior knowledge on action recognition in the form of carefully designed handcrafted features. In this chapter, we will study an alternative method for feeding prior knowledge to machine learning models by embedding this knowledge in the training data itself. And for this, we will show how to create an interpretable parametric generative model of human action videos to generate a diverse, realistic, and physically plausible dataset of human action videos using the strong *a priori* knowledge about the real world stored in the physic engines of modern video-game development systems.

4.1 Introduction

ACCURATE REPRESENTATIONS of both appearance and motion require either carefully handcrafting features with prior knowledge (*e.g.,* the dense trajectories of [202]) or end-to-end deep learning of high capacity models with a large amount of labeled data (*e.g.,* the two-stream network of [170]). In the past chapters, we have seen that these two families of methods have complementary strengths and weaknesses, and how they can be combined to achieve state-of-the-art action recognition performance [43, 207].

Nevertheless, deep neural networks have a much higher potential to significantly improve their accuracy based on training data. Hence they are becoming the de-facto standard for recognition problems where it is possible to collect large labeled training sets, often by crowd-sourcing manual annotations (*e.g.,* ImageNet [87], MS-COCO [109], CityScapes [37]). However, manual labeling is costly, time-consuming, error-prone, raises privacy concerns, and requires massive human intervention for every new task. This is often impractical, especially for videos, or even unfeasible for ground truth modalities like optical flow or depth.

Using synthetic data generated from virtual worlds alleviates these issues. Thanks to modern modeling, rendering, and simulation software, virtual worlds allow for the efficient generation of vast amounts of controlled and algorithmically labeled data, including for modalities that cannot be labeled by a human. This approach has recently shown great promise for deep learning across a breadth of computer vision problems, including optical flow [117], depth estimation [109], object detection [113, 142, 180, 192, 216], pose and viewpoint estimation [136, 169, 178], tracking [57], and semantic segmentation [69, 156, 158].

In this chapter, we investigate *procedural generation of synthetic human action videos* from virtual worlds in order to generate training data for action recognition models. This is an open problem with formidable technical challenges, as it requires a full generative model of videos with realistic appearance and motion statistics conditioned on specific action categories.

The **first contribution** of this chapter is a *parametric generative model of human action videos* relying on physics, scene composition rules, and procedural animation techniques like "ragdoll physics" that provide a much stronger prior than just considering videos as tensors or sequences of frames. We show how to procedurally generate physically plausible variations of different types of action categories obtained by MOCAP datasets, animation blending, physics-based navigation, or entirely from scratch using programmatically defined behaviors. We use naturalistic actor-centric randomized camera paths to film the generated actions with care for physical interactions of the camera. Furthermore, our manually designed generative model has *interpretable parameters* that allow to either randomly sample or precisely control discrete and continuous scene (weather, lighting, environment, time of day, etc), actor, and action variations to generate large amounts of diverse, physically plausible, and realistic human action videos.

The **second contribution** is an experimental validation of our parametric model, using a modern and accessible game engine (Unity®Pro) to synthesize a labeled dataset of $39,982$ videos, corresponding to approximately 6M frames, with more than $1,000$ example videos for each of 35 action categories: 21 grounded in MOCAP data, and 14 entirely synthetic ones defined procedurally. The dataset we propose in this chapter, called *PHAV* for "Procedural Human Action Videos" (*cf.* Figure 4.1

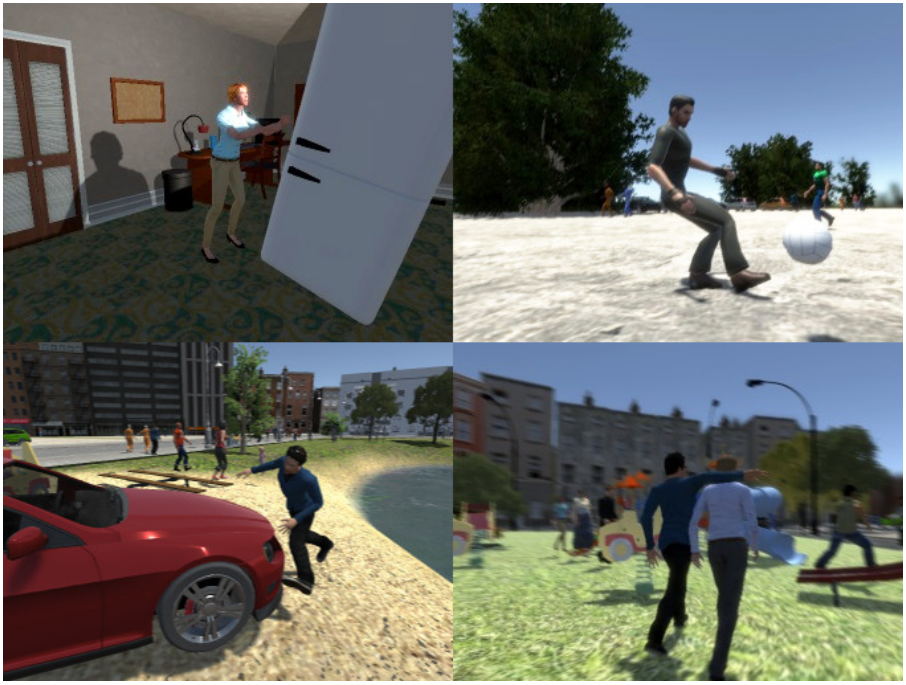


Figure 4.1 – Procedurally generated human action videos. Depicted actions, from the top left: *push, kick ball, car hit, walking hug.* Top are based on variations of existent MOCAP sequences for these actions. Bottom have been programatically defined, with the final movement sequences being created on-the-fly through ragdoll physics and simulating the effect of physical interactions.

for example frames), is publicly available for download[1]. Our procedural generative model took approximately 2 months of work by 2 engineers to be programmed and our PHAV dataset took 3 days to be generated using 4 gaming GPUs.

The rest of the chapter is organized as follows. In Section 4.2, we present the scene and action elements contained in our virtual world and the technologies we use to manipulate them. In section 4.3, we present our interpretable parametric generative model that establishes the relationship between all variables in our virtual world. In Section 4.4, we use our model to procedurally generate our PHAV dataset, listing its main characteristic, and detailing the provided data modalities.

[1] Dataset and tools available in http://adas.cvc.uab.es/phav/

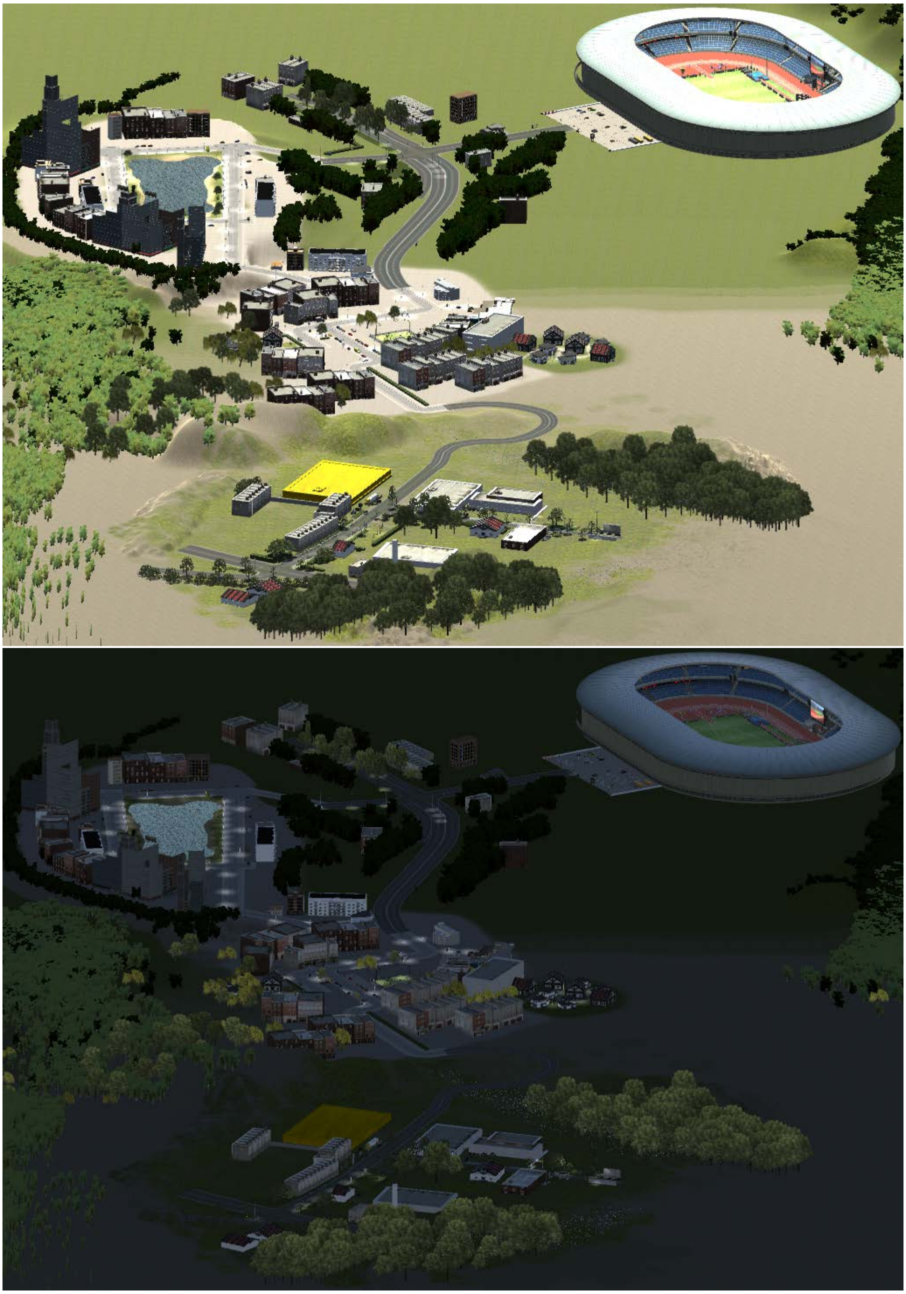


Figure 4.2 – Orthographic view of different world regions during day and night. Time of the day affects lighting and shadows of the world, with urban lights activating at dusk and deactivating at dawn.

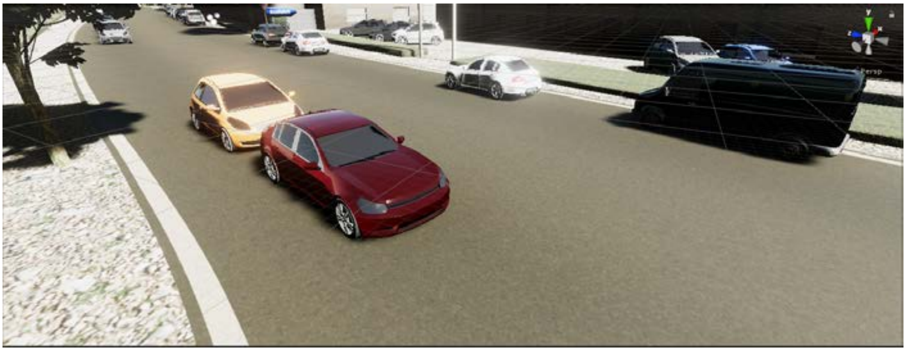


Figure 4.3 – World location shared between PHAV and Virtual KITTI [57], as seen from within the Unity® Pro editor. Image courtesy of Yohann Cabon.

4.2 Virtual scene and action elements

In this section we describe the procedural generation techniques we leverage to randomly sample diverse yet physically plausible appearance and motion variations, both for MOCAP-grounded actions and programmatically defined categories.

4.2.1 Action scene composition

In order to generate a human action video, we place a *protagonist* performing an *action* in an *environment*, under particular *weather conditions* at a specific *period* of the day. There can be one or more *background actors* in the scene, as well as one or more *supporting characters*. We film the virtual scene using a parametric *camera behavior*.

The protagonist is the main human model performing the action. For actions involving two or more people, one is chosen to be the protagonist. Background actors can freely walk in the current virtual environment, while supporting characters are actors with a secondary role necessary to complete an action, *e.g.,* hold hands.

The action is a human motion belonging to a predefined semantic category originated from one or more motion data sources (described in section 4.2.3), including predetermined motions from a MOCAP dataset, or programmatic actions defined using procedural animation techniques [47, 190], in particular ragdoll physics. In addition, we use these techniques to sample physically-plausible motion variations (described in section 4.2.4) to increase diversity.

The environment refers to a region in the virtual world (*cf.* Figure 4.2), which

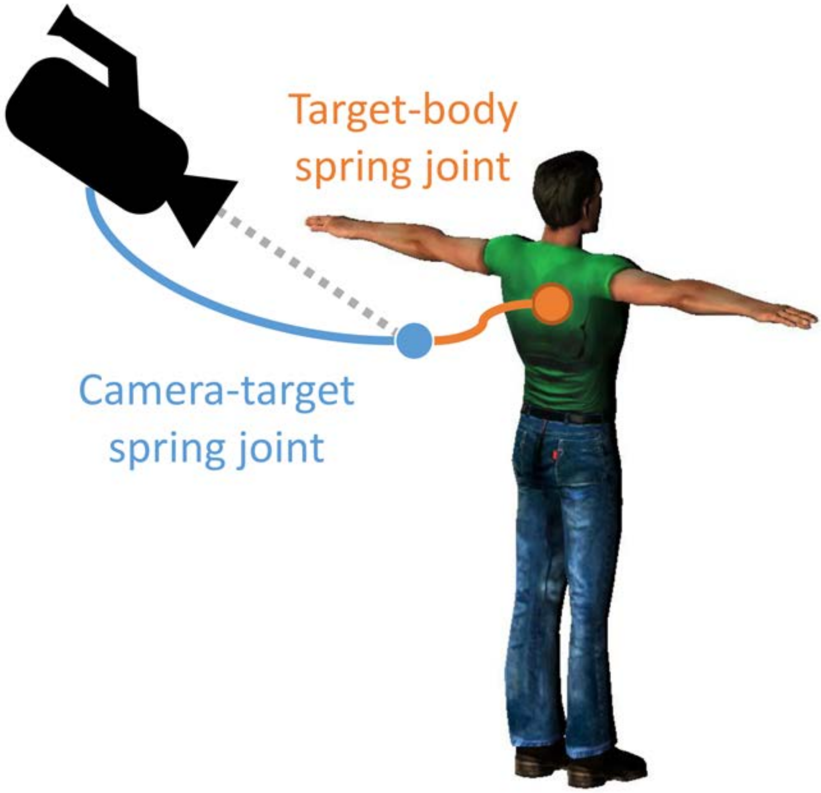


Figure 4.4 – Schematic representation of our Kite camera.

consists of large urban areas, natural environments (*e.g.,* forests, lakes, and parks), indoor scenes, and sports grounds (*e.g.,* a stadium). Each of these environments may contain moving or static background pedestrians or objects – *e.g.,* cars, chairs – with which humans can physically interact, voluntarily or not. The outdoor weather in the virtual world can be rainy, overcast, clear, or foggy. The period of the day can be dawn, day, dusk, or night.

Similar to [57, 158], we use a library of pre-made 3D models obtained from the Unity Asset Store, which includes artist-designed human, object, and texture models, as well as semi-automatically created realistic environments (*e.g.,* selected scenes from the VKITTI dataset [57], *cf.* Figure 4.3).

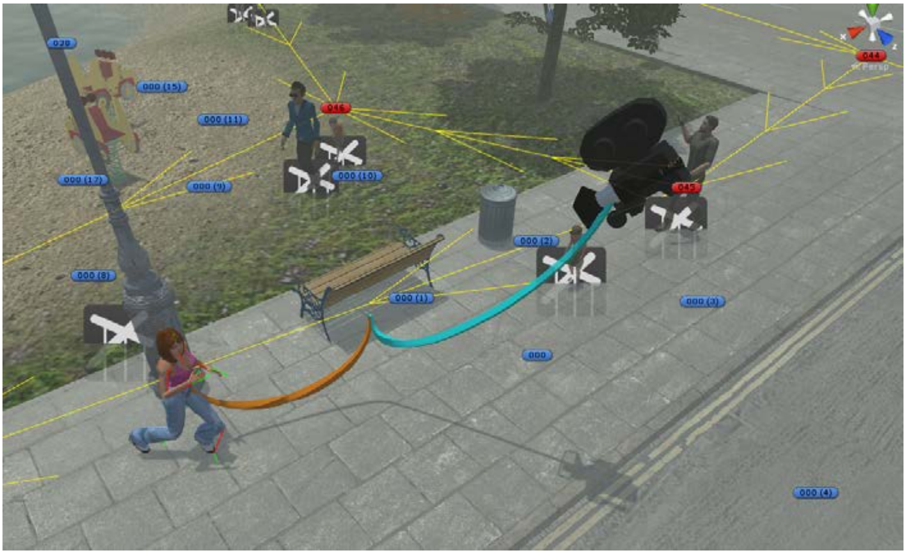


Figure 4.5 – In-editor representation of the Kite camera. The camera is a physical object capable of interacting with other objects in the world, which avoids trespassing walls or filming from unfeasible locations. The camera focuses on a point (contact point between orange and blue cords) which is simultaneously attached to the protagonist and to the camera.

4.2.2 Camera

We use a physics-based camera which we call the Kite camera (*cf.* Figure 4.4) to track the protagonist in a scene. This physics-aware camera is governed by a rigid body attached by a spring to a target position that is, in turn, attached to the protagonist by another spring. By randomly sampling different parameters for the drag and weight of the rigid bodies, as well as elasticity and length of the springs, we can achieve cameras with a wide range of shot types, 3D transformations, and tracking behaviors, such as following the actor, following the actor with a delay, or stationary. Another parameter controls the direction and strength of an initial impulse that starts moving the camera in a random direction. With different rigid body parameters, this impulse can cause our camera to simulate a handheld camera, move in a circular trajectory, or freely bounce around in the scene while filming the attached protagonist. A representation of the camera attachment in the virtual world is shown in Figure 4.5.

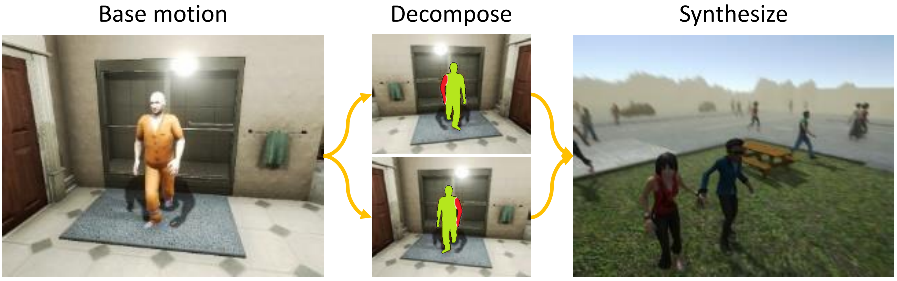


Figure 4.6 – One of our approaches for creating new animation sequences from a motion data source. We decompose existing action sequences (left) into atomic motions (middle) and then recombine them into new animation sequences using procedural animation techniques, like blending and ragdoll physics. This technique can be used to both generate new motion variations for an existing action category, and to synthesize new motion sequences for entirely synthetic categories which do not exist in the data source using simple programmable rules *e.g.,* by tying the ragdoll hands together (right). The physics engine enforces that the performed ragdoll manipulations result in physically plausible animations.

4.2.3 Actions

Our approach relies on two main existing data sources for basic human animations. First, we use the CMU MOCAP database [21], which contains 2605 sequences of 144 subjects divided in 6 broad categories, 23 subcategories and further described with a short text. We leverage relevant motions from this dataset to be used as a motion source for our procedural generation based on a simple filtering of their textual motion descriptions. Second, we use a large amount of hand-designed realistic motions made by animation artists and available on the Unity Asset Store.

The key insight of our approach is that *these sources need not necessarily contain motions from predetermined action categories of interest, neither synthetic nor target real-world actions (unknown a priori).* Instead, we propose to use these sources to form a *library of atomic motions* to procedurally generate realistic action categories. We consider atomic motions as individual movements of a limb in a larger animation sequence. For example, atomic motions in a "walk" animation include movements such as rising a left leg, rising a right leg, and pendular arm movements. Creating a library of atomic motions enables us to later recombine those atomic actions into new higher-level animation sequences, e.g., "hop" or "stagger".

Table 4.1 – Actions categories included in our PHAV dataset.

Type	Count	Actions
sub-HMDB	21	brush hair, catch, clap, climb stairs, golf, jump, kick ball, push, pick, pour, pull up, run, shoot ball, shoot bow, shoot gun, sit, stand, swing baseball, throw, walk, wave
One-person synthetic	10	car hit, crawl, dive floor, flee, hop, leg split, limp, moonwalk, stagger, surrender
Two-people synthetic	4	walking hug, walk hold hands, walk the line, bump into each other

Our PHAV dataset contains 35 different action classes (*cf.* Table 4.1), including 21 simple categories present in HMDB-51 and composed directly of some of the aforementioned atomic motions. In addition to these actions, we programmatically define 10 action classes involving a single actor and 4 action classes involving two person interactions. We create these new synthetic actions by taking atomic motions as a base and using procedural animation techniques like blending and ragdoll physics (*cf.* Section 4.2.4) to compose them in a physically plausible manner according to simple rules defining each action, such as tying hands together (*e.g.,* "walk hold hands", *cf.* Figure 4.6), disabling one or more muscles (*e.g.,* "crawl", "limp"), or colliding the protagonist against obstacles (*e.g.,* "car hit", "bump into each other").

4.2.4 Physically plausible motion variations

We now describe procedural animation techniques [47, 190] to randomly generate large amounts of physically plausible and diverse action videos, far beyond what can be achieved by simply replaying source atomic motions.

Ragdoll physics. A key component of our work is the use of ragdoll physics. Ragdoll physics are limited real-time physical simulations that can be used to animate a model (such as a human model) while respecting basic physics properties such as

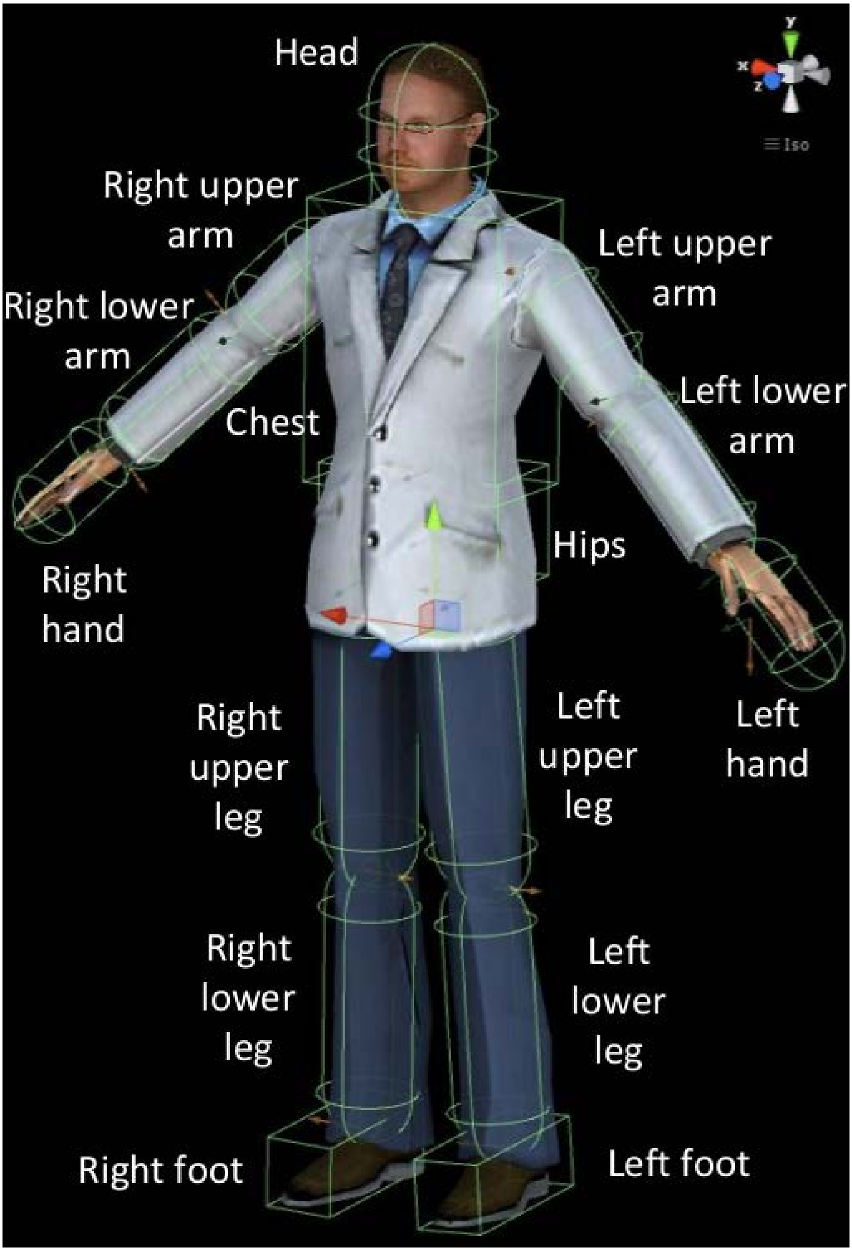


Figure 4.7 – Ragdoll configuration with 15 muscles.

connected joint limits, angular limits, weight and strength. We consider ragdolls with 15 movable body parts (referenced herein as muscles), as illustrated in Figure 4.7. For each action, we separate those 15 muscles into two disjoint groups: those that are strictly necessary for performing the action, and those that are complementary (altering their movement should not interfere with the semantics of the currently considered action). The presence of the ragdoll allows us to introduce variations of different nature in the generated samples. The other modes of variability generation described in this section will assume that the physical plausibility of the models is being kept by the use of ragdoll physics. We use RootMotion's PuppetMaster[2] for implementing and controlling human ragdolls in Unity® Pro.

Random perturbations. Inspired by [144], we create variations of a given motion by adding random perturbations to muscles that should not alter the semantic category of the action being performed. Those perturbations are implemented by adding a rigid body to a random subset of the complementary muscles. Those bodies are set to orbit around the muscle's position in the original animation skeleton, drifting the movement of the puppet's muscle to its own position in a periodic oscillating movement. More detailed references on how to implement variations of this type can be found in [47, 144, 145, 190] and references therein.

Muscle weakening. We vary the strength of the avatar performing the action. By reducing its strength, the actor performs an action with seemingly more difficulty.

Action blending. Similarly to modern video games, we use a blended ragdoll technique to constrain the output of a pre-made animation to physically plausible motions. In action blending, we randomly sample a different motion sequence (coming either from the same or from a different action class, which we refer to as the *base motion*) and replace the movements of current complementary muscles with those from this new sequence. We limit the number of blended sequences in PHAV to be at most two.

Objects. The last physics-based source of variation is the use of objects. First, we manually annotated a subset of the MOCAP actions marking the instants in time where the actor started or ended the manipulation of an object. Second, we use inverse kinematics to generate plausible programmatic interactions.

[2] http://root-motion.com

Table 4.2 – Overview of key random variables of our parametric generative model of human action videos (*cf.* Section 4.3.2 for a more detailed list).

Parameter	Variable	Count	Possible values
Human Model	H	20	models designed by artists
Environment	E	7	simple, urban, green, middle, lake, stadium, house interior
Weather	W	4	clear, overcast, rain, fog
Period of day	D	4	night, dawn, day, dusk
Variation	V	5	none, muscle perturbation, muscle weakening, action blending, objects

4.3 Interpretable parametric generative model

In this section we introduce our interpretable parametric generative model of videos depicting particular human actions, and show how we use it to generate our PHAV dataset. We start by providing a simplified version of our model (*cf.* Figure 4.8), listing the main variables in our approach and giving an overview of how our model is organized. After this brief overview, we show our complete model (*cf.* Figure 4.9) and describe its multiple components in detail.

4.3.1 Overview

We define a human action video as a random variable

$$X = \langle H, A, L, B, V, C, E, D, W \rangle \,, \tag{4.1}$$

where H is a human model, A an action category, L a video length, B a set of basic motions (from MOCAP, manual design, or programmed), V a set of motion variations, C a camera, E an environment, D a period of the day, W a weather condition, and possible values for those parameters are shown in Table 4.2. Given this definition, a simplified version for our generative model (*cf.* Figure 4.8) for an action video X can then be given by:

$$
\begin{aligned}
P(X) = & P(H)\ P(A)\ P(L \mid B)\ P(B \mid A) \\
& P(\Theta_v \mid V)\ P(V \mid A)\ P(\Theta_e \mid E)\ P(E \mid A) \\
& P(\Theta_c \mid C)\ P(C \mid A, E) \\
& P(\Theta_d \mid D)\ P(D)\ P(\Theta_w \mid W)\ P(W)
\end{aligned}
\tag{4.2}
$$

where Θ_w is a random variable on weather-specific parameters (*e.g.*, intensity of rain, clouds, fog), Θ_c is a random variable on camera-specific parameters (*e.g.*, weights and stiffness for Kite camera springs), Θ_e is a random variable on environment-specific parameters (*e.g.*, current waypoint, waypoint locations, background pedestrian starting points and destinations), Θ_d is a random variable on period-specific parameters (*e.g.*, amount of sunlight, sun orientation), and Θ_v is a random variable on variation-specific parameters (*e.g.*, strength of each muscle, strength of perturbations, blending muscles). The probability functions associated with categorical variables (*e.g.*, A) can be either uniform, or configured manually to use pre-determined weights. Similarly, probability distributions associated with continuous values (*e.g.*, Θ_c) are either set using a uniform distribution with finite support, or using triangular distributions with pre-determined support and most likely value.

We now proceed to give additional details about our graphical model, as well as the values used to configure the parameter distributions in the next sections.

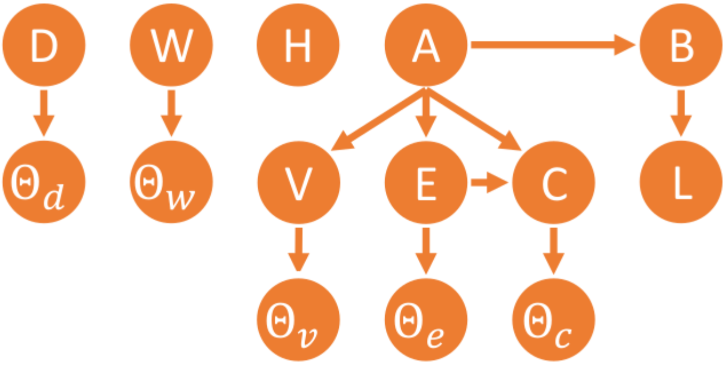


Figure 4.8 – A simplified view of the graphical model for our generator (*cf.* Section 4.3.2 for the meaning of each variable). A complete and more detailed version is shown in Figure 4.9.

4.3.2 Variables

After the past overview, we now proceed to define the complete version of our generative model. We start by giving a more precise definition for its main random variables. Here we focus only on critical variables that are fundamental in understanding the orchestration of the different parts of our generation, whereas all part-specific variables are shown in Section 4.3.3. The categorical variables that drive most of the procedural generation are:

$$
\begin{aligned}
H:\ h &\in \{model_1, model_2, \ldots, model_{20}\}\\
A:\ a &\in \{``clap", \ldots, ``bump\ into\ each\ other"\}\\
B:\ b &\in \{motion_1, motion_2, \ldots, motion_{953}\}\\
V:\ v &\in \{``none", ``random\ perturbation",\\
&\quad ``weakening", ``objects", ``blending"\}\\
C:\ c &\in \{``kite", ``indoors", ``closeup", ``static"\}\\
E:\ e &\in \{``urban", ``stadium", ``middle",\\
&\quad ``green", ``house", ``lake"\}\\
D:\ d &\in \{``dawn", ``day", ``dusk", ``night"\}\\
W:\ w &\in \{``clear", ``overcast", ``rain, ``fog"\}
\end{aligned}
\tag{4.3}
$$

where H is the human model to be used by the protagonist, A is the action category for which the video should be generated, B is the motion sequence (*e.g.,* from MOCAP, created by artists, or programmed) to be used as a base upon which motion variations can be applied (*e.g.,* blending it with secondary motions), V is the motion variation to be applied to the base motion, C is the camera behavior, E is the environment of the virtual world where the action will take place, D is the day phase, and W is the weather condition.

These categorical variables are in turn controlled by a group of parameters that can be adjusted in order to drive the sample generation. These parameters include the θ_A parameters of a categorical distribution on action categories A, the θ_W for weather conditions W, θ_D for day phases D, θ_H for model models H, θ_V for variation types V, and θ_C for camera behaviors C.

Additional parameters include the conditional probability tables of the dependent variables: a matrix of parameters θ_{AE} where each row contains the parameters for categorical distributions on environments E for each action category A, the matrix of parameters θ_{AC} on camera behaviors C for each action A, the matrix of parameters θ_{EC} on camera behaviors C for each environment E, and the matrix of parameters θ_{AB} on motions B for each action A.

Finally, other relevant parameters include T_{min}, T_{max}, and T_{mod}, the minimum, maximum and most likely durations for the generated video. We denote the set of all parameters in our model by $\boldsymbol{\theta}$.

4.3.3 Model

The complete interpretable parametric probabilistic model used by our generation process, given our generation parameters $\boldsymbol{\theta}$, can be written as:

$$\begin{aligned} &P(H,A,L,B,V,C,E,D,W \mid \boldsymbol{\theta}) = \\ &\qquad P_1(D,W \mid \boldsymbol{\theta}) \quad P_2(H \mid \boldsymbol{\theta}) \quad P_3(A,L,B,V,C,E,W \mid \boldsymbol{\theta}) \end{aligned} \tag{4.4}$$

where P_1, P_2 and P_3 are defined by the probabilistic graphical models represented on Figure 4.9a, 4.9b and 4.9c, respectively. We use extended plate notation [14] to indicate repeating variables, marking parameters (non-variables) using filled rectangles.

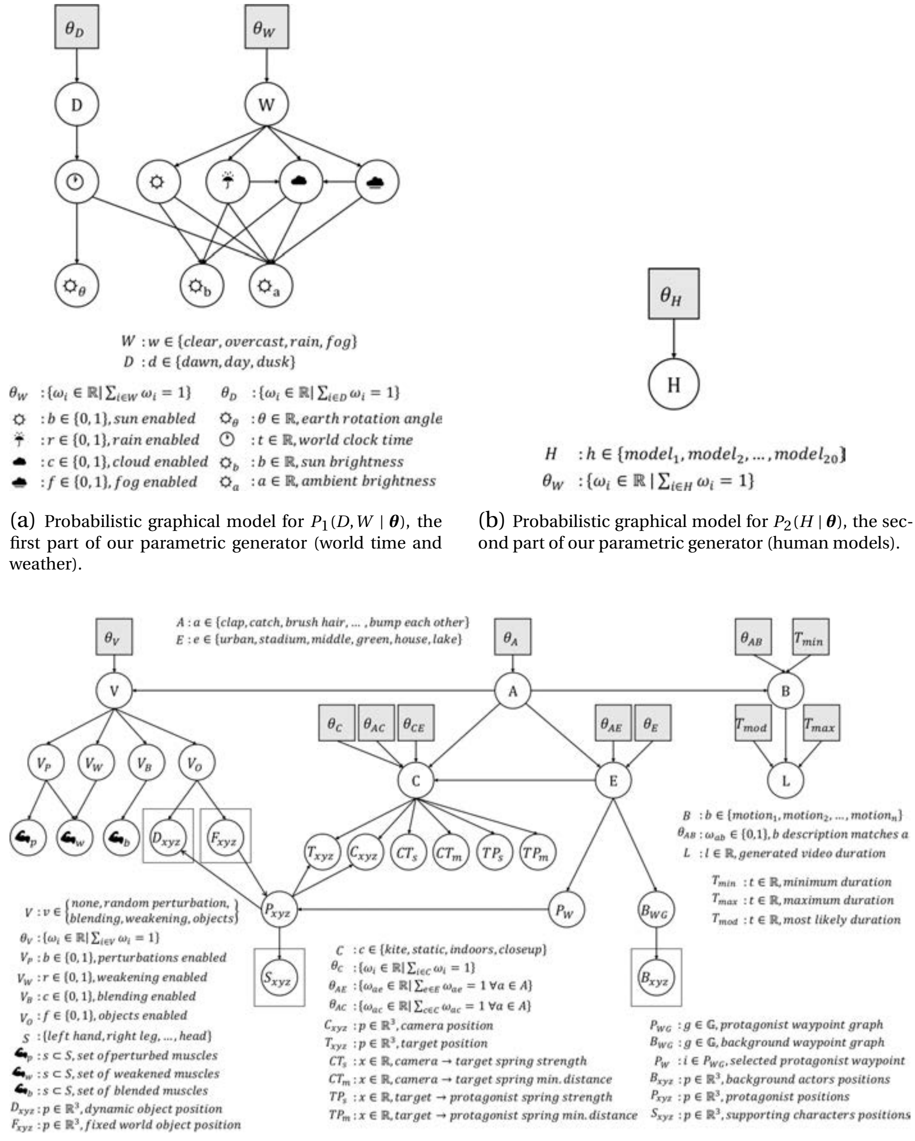
4.3.4 Distributions

The generation process makes use of four main families of distributions: categorical, uniform, Bernoulli and triangular. We adopt the following three-parameter formulation for the triangular distribution:

$$Tr(x \mid a,b,c) = \begin{cases} 0 & \text{for } x < a, \\ \frac{2(x-a)}{(b-a)(c-a)} & \text{for } a \le x < c, \\ \frac{2}{b-a} & \text{for } x = c, \\ \frac{2(b-x)}{(b-a)(b-c)} & \text{for } c < x \le b, \\ 0 & \text{for } b < x. \end{cases} \tag{4.5}$$

All distributions are implemented using the open-source Accord.NET Framework[3] [40]. While we have used mostly uniform distributions to create the dataset used in our experiments, we have the possibility to bias the generation towards values that are closer to real-world dataset statistics.

[3] http://accord-framework.net


θ_D
D
θ_W
W
$W : w \in \{clear, overcast, rain, fog\}$
$D : d \in \{dawn, day, dusk\}$
$\theta_W : \{\omega_i \in \mathbb{R} \mid \sum_{i \in W} \omega_i = 1\}$
$\theta_D : \{\omega_i \in \mathbb{R} \mid \sum_{i \in D} \omega_i = 1\}$
$: b \in \{0,1\}$, sun enabled
$\theta : \theta \in \mathbb{R}$, earth rotation angle
$: r \in \{0,1\}$, rain enabled
$: t \in \mathbb{R}$, world clock time
$: c \in \{0,1\}$, cloud enabled
$b : b \in \mathbb{R}$, sun brightness
$: f \in \{0,1\}$, fog enabled
$a : a \in \mathbb{R}$, ambient brightness
θ_H
H
$H : h \in \{model_1, model_2, \dots, model_{20}\}$
$\theta_W : \{\omega_i \in \mathbb{R} \mid \sum_{i \in H} \omega_i = 1\}$
$A : a \in \{clap, catch, brush\ hair, \dots, bump\ each\ other\}$
$E : e \in \{urban, stadium, middle, green, house, lake\}$
θ_V
V
θ_A
A
θ_{AB}
T_{min}
B
θ_C
θ_{AC}
θ_{CE}
θ_{AE}
θ_E
T_{mod}
T_{max}
V_P
V_W
V_B
V_O
C
E
L
p
w
b
D_{xyz}
F_{xyz}
T_{xyz}
C_{xyz}
CT_s
CT_m
TP_s
TP_m
P_{xyz}
P_W
B_{WG}
S_{xyz}
B_{xyz}
$B : b \in \{motion_1, motion_2, \dots, motion_n\}$
$\theta_{AB} : \omega_{ab} \in \{0,1\}$, b description matches a
$L : l \in \mathbb{R}$, generated video duration
$T_{min} : t \in \mathbb{R}$, minimum duration
$T_{max} : t \in \mathbb{R}$, maximum duration
$T_{mod} : t \in \mathbb{R}$, most likely duration
$V : v \in \{none, random\ perturbation, blending, weakening, objects\}$
$\theta_V : \{\omega_i \in \mathbb{R} \mid \sum_{i \in V} \omega_i = 1\}$
$V_P : b \in \{0,1\}$, perturbations enabled
$V_W : r \in \{0,1\}$, weakening enabled
$V_B : c \in \{0,1\}$, blending enabled
$V_O : f \in \{0,1\}$, objects enabled
$S : \{left\ hand, right\ leg, \dots, head\}$
$_p : s \subset S$, set ofperturbed muscles
$_w : s \subset S$, set of weakened muscles
$_b : s \subset S$, set of blended muscles
$D_{xyz} : p \in \mathbb{R}^3$, dynamic object position
$F_{xyz} : p \in \mathbb{R}^3$, fixed world object position
$C : c \in \{kite, static, indoors, closeup\}$
$\theta_C : \{\omega_i \in \mathbb{R} \mid \sum_{i \in C} \omega_i = 1\}$
$\theta_{AE} : \{\omega_{ae} \in \mathbb{R} \mid \sum_{e \in E} \omega_{ae} = 1\ \forall a \in A\}$
$\theta_{AC} : \{\omega_{ac} \in \mathbb{R} \mid \sum_{c \in C} \omega_{ac} = 1\ \forall a \in A\}$
$C_{xyz} : p \in \mathbb{R}^3$, camera position
$T_{xyz} : p \in \mathbb{R}^3$, target position
$CT_s : x \in \mathbb{R}$, camera → target spring strength
$CT_m : x \in \mathbb{R}$, camera → target spring min. distance
$TP_s : x \in \mathbb{R}$, target → protagonist spring strength
$TP_m : x \in \mathbb{R}$, target → protagonist spring min. distance
$P_{WG} : g \in \mathbb{G}$, protagonist waypoint graph
$B_{WG} : g \in \mathbb{G}$, background waypoint graph
$P_W : i \in P_{WG}$, selected protagonist waypoint
$B_{xyz} : p \in \mathbb{R}^3$, background actors positions
$P_{xyz} : p \in \mathbb{R}^3$, protagonist positions
$S_{xyz} : p \in \mathbb{R}^3$, supporting characters positions

(a) Probabilistic graphical model for $P_1(D, W \mid \boldsymbol{\theta})$, the first part of our parametric generator (world time and weather).

(b) Probabilistic graphical model for $P_2(H \mid \boldsymbol{\theta})$, the second part of our parametric generator (human models).

(c) Probabilistic graphical model for $P_3(A, L, B, V, C, E, W \mid \boldsymbol{\theta})$, the third part of our parametric generator (scene and action preparation).

Figure 4.9 – Our complete probabilistic graphical model, divided in three parts.

Day phase. As real-world action recognition datasets are more likely to contain video recordings captured during daylight, we fixed the parameter θ_D such that

$$\begin{aligned} P(D = dawn \quad &| \theta_D) = \quad 1/3 \\ P(D = day \quad &| \theta_D) = \quad 1/3 \\ P(D = dusk \quad &| \theta_D) = \quad 1/3 \\ P(D = night \quad &| \theta_D) = \quad 0. \end{aligned} \tag{4.6}$$

We note that although our system can also generate night samples, we do not include them in PHAV at this moment.

Weather. In order to support a wide range of applications of our dataset, we fixed the parameter θ_W such that

$$\begin{aligned} P(W = clear \quad &| \theta_W) = \quad 1/4 \\ P(W = overcast \quad &| \theta_W) = \quad 1/4 \\ P(W = rain \quad &| \theta_W) = \quad 1/4 \\ P(W = fog \quad &| \theta_W) = \quad 1/4, \end{aligned} \tag{4.7}$$

ensuring all weather conditions are present.

Camera. In addition to the Kite camera, we also included specialized cameras that can be enabled only for certain environments (Indoors), and certain actions (Close-Up). We fixed the parameter θ_C such that

$$\begin{aligned} P(C = kite \quad &| \theta_C) = \quad 1/3 \\ P(C = closeup \quad &| \theta_C) = \quad 1/3 \\ P(C = indoors \quad &| \theta_C) = \quad 1/3. \end{aligned} \tag{4.8}$$

However, we have also fixed θ_{CE} and θ_{AC} such that the Indoors camera is only available for the house environment, and that the Close-Up camera can also be used for the *BrushHair* action in addition to Kite.

Environment, human model and variations. We fixed the parameters θ_E, θ_H, and θ_V using equal weights, such that the variables E, H, and V can have uniform distributions.

Base motions. We select a main motion sequence which will be used as a base upon which a variation V is applied *cf.* Section 4.3.2). Base motions are weighted according to the minimum video length parameter T_{min}, where motions whose duration is less than T_{min} are assigned weight zero, and others are set to uniform, such that

$$P(B = b | T_{min}) \propto \begin{cases} 1 & \text{if } length(b) \geq T_{min} \\ 0 & \text{otherwise} \end{cases}. \tag{4.9}$$

This weighting is used to ensure that the motion that will be used as a base is long enough to fill the minimum desired duration for a video. We then perform the selection of a motion B given a category A by introducing a list of regular expressions associated with each of the action categories. We then compute matches between the textual description of the motion in its source (*e.g.,* short text descriptions in [21]) and these expressions, such that

$$(\theta_{AB})_{ab} = \begin{cases} 1 & \text{if } match(\text{regex}_a, \text{desc}_b) \\ 0 & \text{otherwise} \end{cases} \quad \forall a \in A, \forall b \in B. \tag{4.10}$$

We then use θ_{AB} such that

$$P(B = b \mid A = a, \theta_{AB}) \propto (\theta_{AB})_{a,b}. \tag{4.11}$$

Weather elements. The selected weather W affects world parameters such as the sun brightness, ambient luminosity, and multiple boolean variables that control different aspects of the world (*cf.* Figure 4.9a). The activation of one of these boolean variables (*e.g.,* fog visibility) can influence the activation of others (*e.g.,* clouds) according to Bernoulli distributions ($p = 0.5$).

World clock time. The world time is controlled depending on D. In order to avoid generating a large number of samples in the borders between two periods of the day, where the distinction between both phases is blurry, we use different triangular distributions associated with each phase, giving a larger probability to hours of interest (sunset, dawn, noon) and smaller probabilities to hours at the transitions. We therefore define the distribution of the world clock times $P(T)$ as:

$$P(T=t \mid D) \propto \sum_{d \in D} P(T=t \mid D=d) \tag{4.12}$$

where

$$\begin{aligned}
P(T=t \mid D=dawn\ \ ) &= Tr(t \mid\ 7h, \quad 10h,\ \ 9h\ \ )\\
P(T=t \mid D=day\ \ ) &= Tr(t \mid 10h, \quad 16h,\ 13h\ \ )\\
P(T=t \mid D=dusk\ \ ) &= Tr(t \mid 17h, \quad 20h,\ 18h\ \ )\\
P(T=t \mid D=night\ \ ) &= Tr(t \mid 20h, \quad\ \ 7h,\ \ 0h\ \ ).
\end{aligned} \tag{4.13}$$

Generated video duration. The selection of the clip duration L given the selected motion b is performed considering the motion length L_b, the maximum video length T_{min} and the desired mode T_{mod}:

$$\begin{aligned}
P(L=l \mid B=b) = Tr(a = T_{min},&\\
b = min(L_b, T_{max}),&\\
c = min(T_{mod}, L_b)).&
\end{aligned} \tag{4.14}$$

Actors placement and environment. Each environment E has at most two associated waypoint graphs. One graph refers to possible positions for the protagonist, while an additional second graph gives possible positions B_{WG} for spawning background actors. Indoor scenes (*cf.* Figure 4.10) do not include background actor graphs. After an environment has been selected, a waypoint P_W is randomly selected from the graph using an uniform distribution. The protagonist position P_{xyz} is then set according to the position of P_W. The S_{xyz} position of each supporting character, if any, is set depending on P_{xyz}. The position and destinations for the background actors are set depending on B_{WG}.

Camera placement and parameters. After a camera has been selected, its position C_{xyz} and the position T_{xyz} of the target are set depending on the position P_{xyz} of the protagonist. The camera parameters are randomly sampled using uniform distributions on sensible ranges according to the observed behavior in Unity. The most relevant secondary variables for the camera are shown in Figure 4.9c. They

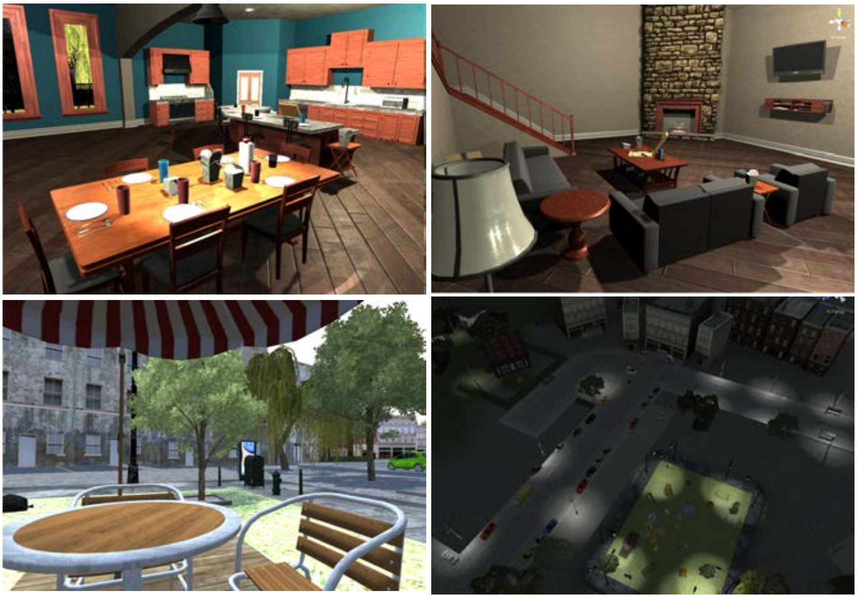


Figure 4.10 – Example of indoor (top) and outdoor (bottom) locations.

include Unity-specific parameters for the camera-target (CT_s, CT_m) and target-protagonist springs (TP_s, CT_m) that can be used to control their strength and a minimum distance tolerance zone in which the spring has no effect (remains at rest). In our generator, the minimum distance is set to either 0, 1 or 2 meters with uniform probabilities. This setting is responsible for a "delay" effect that allows the protagonist to not be always in the center of camera focus (and thus avoiding creating such bias in the data).

Action variations. After a variation mode has been selected, the generator needs to select a subset of the ragdoll muscles (*cf.* Figure 4.7) to be perturbed (random perturbations) or to be replaced with movement from a different motion (action blending). These muscles are selected using a uniform distribution on muscles that have been marked as non-critical depending on the previously selected action category A. When using weakening, a subset of muscles will be chosen to be weakened with varying parameters independent of the action category. When using objects, the choice of objects to be used and how they have to be used is also dependent on the action category.

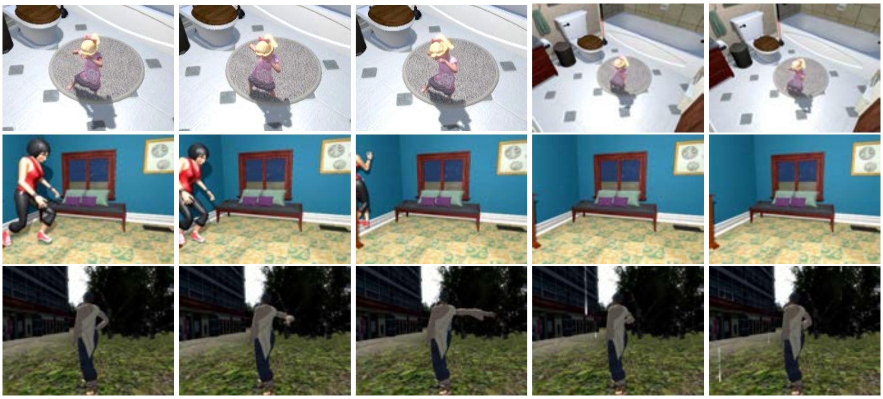


Figure 4.11 – Example generation failure cases. First row: too strong perturbations (tiny model, brushing hair looks like dancing). Second row: limitation in the physics engine together with ragdoll system and MOCAP action can lead to physics violations (passing through a wall). Third row: problems in the automatic configuration of the ragdoll model can result in overconstrained joints and unintended variations.

Failure cases. Although our approach uses physics-based procedural animation techniques, unsupervised generation of large amounts of random variations with a focus on diversity inevitably causes edge cases where physical models fail. This results in glitches reminiscent of typical video game bugs (*cf.* Figure 4.11). Using a random 1% sample of our dataset, we manually estimated that this corresponds to less than 10% of the videos generated. Although this could be improved, our experiments in Chapter 5 show that this noise does not prevent us from improving the training of deep action recognition networks using this data.

Extension to complex activities. Using ragdoll physics and a large enough library of atomic actions, it is possible to create complex actions by hierarchical composition. For instance, our "Car Hit" action is procedurally defined by composing atomic actions of a person (walking and/or doing other activities) with those of a car (entering in a collision with the person), followed by the person falling in a physically plausible fashion. However, while atomic actions have been validated as an effective decomposition for the recognition of potentially complex actions [55], we have not studied how this approach would scale with the complexity of the actions, notably due to the combinatorial nature of complex events. We leave this as future work.

4.4 Generating a synthetic action dataset

We validate our approach for synthetic video generation by generating a new dataset for action recognition, such that the data from this dataset could be used to complement the training set of existing target real-world datasets in order to obtain action classification models which perform better in their respective real-world tasks. In this section we give details about how we have used the aforedescribed model to generate our PHAV dataset.

In order to create PHAV, we generate videos with lengths between 1 and 10 seconds, at 30 FPS, and resolution of 340 × 256 pixels, as this is the same resolution expected by state-of-the-art action recognition models such as [208]. We use anti-aliasing, motion blur, and standard photo-realistic cinematic effects (*cf.* Figure 4.12). We have generated 55 hours of videos, with approximately $6M$ frames and at least $1,000$ videos per action category.

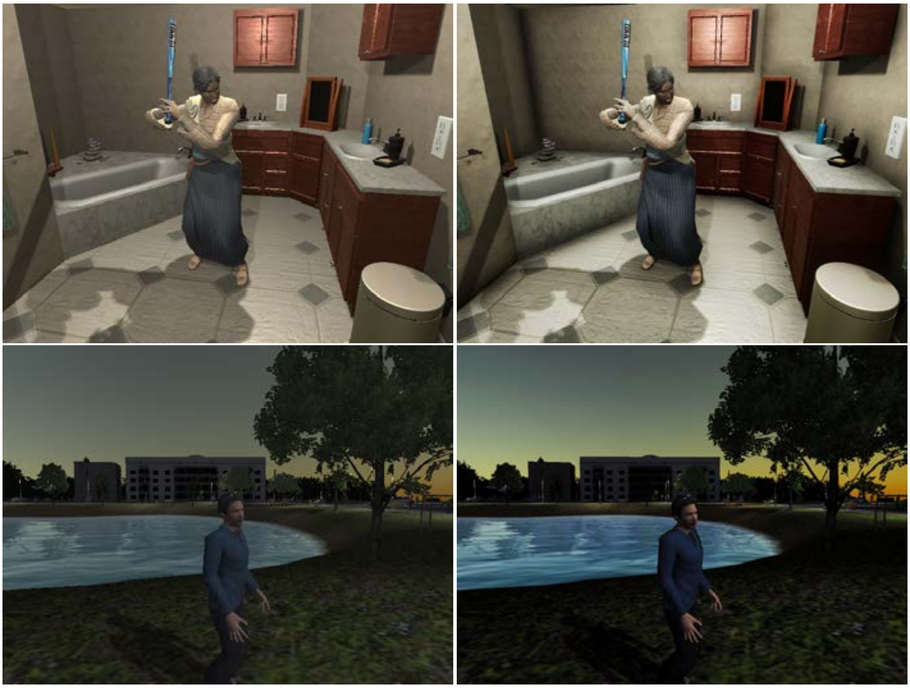


Figure 4.12 – Comparison between raw (left) *vs.* post-processed (right) RGB frames.

Table 4.3 – Statistics of the generated dataset instance.

Statistic	Value
Total dataset clips	39,982
Total dataset frames	5,996,286
Total dataset duration	2d07h31m
Average video duration	4.99s
Average number of frames	149.97
Frames per second	30
Video width	340
Video height	256
Average clips per category	1,142.3
Image modalities (streams)	6

Our parametric model can generate fully-annotated action videos (including depth, flow, semantic segmentation, and human pose ground-truths) at 3.6 FPS using one consumer-grade gaming GPU (NVIDIA GTX 1070). In contrast, the average annotation time for data-annotation methods such as [19, 37, 156] are significantly below 0.5 FPS. While those works deal with semantic segmentation (where the cost of annotation is higher than for action classification), we can generate all modalities for roughly the same cost as RGB.

4.4.1 Statistics

A summary of the key statistics for the generated dataset can be seen in Table 4.3. Figure 4.13 shows the number of videos generated for each action category in PHAV. As it can be seen, the number is higher than 1,000 samples for all categories. Figure 4.14 shows the number of videos generated by value of each main random generation variable. The histograms reflect the probability values presented in Section 4.3.4. While our parametric model is flexible enough to generate a wide range of world variations, we have focused on generating videos that would be more similar to those in the target datasets.

4.4.2 Data modalities

Our generator outputs multiple data modalities for a single video, which we include in our public release of PHAV (*cf.* Figures 4.15 and 4.16). Those data modalities are rendered roughly at the same time using Multiple Render Targets (MRT), resulting in a superlinear speedup as the number of simultaneous output data modalities grow. The modalities in our public release include:

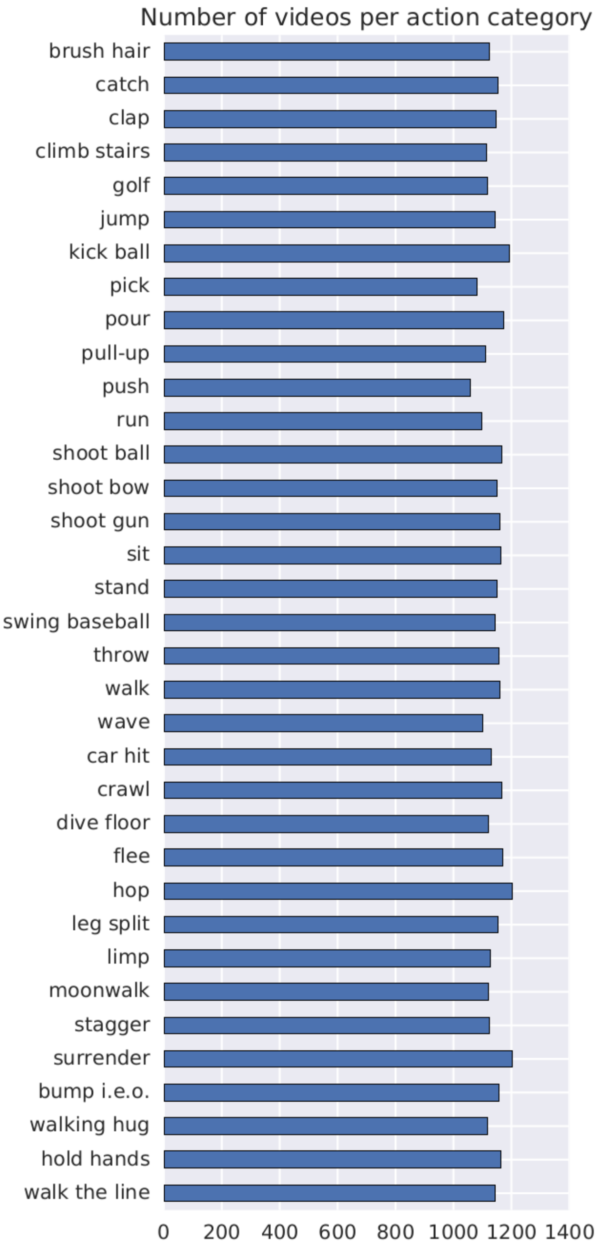


Figure 4.13 – Plot of the number of videos generated for each category in PHAV.

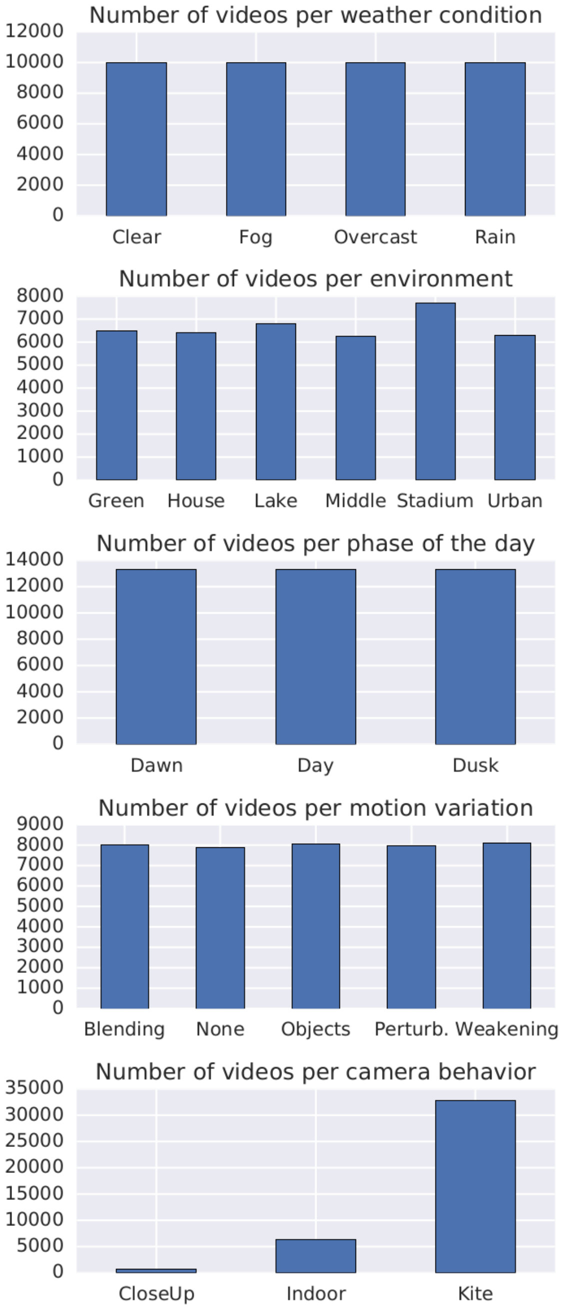


Figure 4.14 – Plot of the number of videos per parameter value in PHAV.

Rendered RGB Frames. These are the RGB frames that constitute the action video. They are rendered at 340 × 256 resolution and 30 FPS such that they can be directly feed to Two-Stream style networks. Those frames have been post-processed with 2x Supersampling Anti-Aliasing (SSAA) [23, 123], motion blur [177], bloom [177], ambient occlusion [102, 120, 157], screen space reflection [174], color grading [166], and vignette [223].

Semantic Segmentation. These are the per-pixel semantic segmentation ground-truths containing the object class label annotations for every pixel in the RGB frame. They are encoded as sequences of 24-bpp PNG files with the same resolution as the RGB frames. We provide 63 pixel classes (*cf.* Table 4.4), which include the same 14 classes used in Virtual KITTI [57], classes specific for indoor scenarios, classes for dynamic objects used in every action, and 27 classes depicting body joints and limbs (*cf.* Figure 4.17).

Instance Segmentation. These are the per-pixel instance segmentation ground-truths containing the person identifier encoded as different colors in a sequence of frames. They are encoded in exactly the same way as the semantic segmentation ground-truth explained above.

Depth Map. These are depth map ground-truths for each frame. They are represented as a sequence of 16-bit grayscale PNG images with a fixed far plane of 655.35 meters. This encoding ensures that a pixel intensity of 1 can correspond to a 1cm distance from the camera plane.

Optical Flow. These are the ground-truth (forward) optical flow fields computed from the current frame to the next frame. We provide separate sequences of frames for the horizontal and vertical directions of optical flow represented as sequences of 16-bpp JPEG images with the same resolution as the RGB frames.

Raw RGB Frames. These are the raw RGB frames before any of the post-processing effects mentioned above are applied. This modality is mostly included for completeness, and has not been used in experiments shown in this thesis.

Pose, location and additional information. Although not an image modality, our generator also produces extended metadata for every frame. This metadata includes camera parameters, 3D and 2D bounding boxes, joint locations in screen coordinates (pose), and muscle information (including muscular strength, body limits and other physical-based annotations) for every person in a frame.

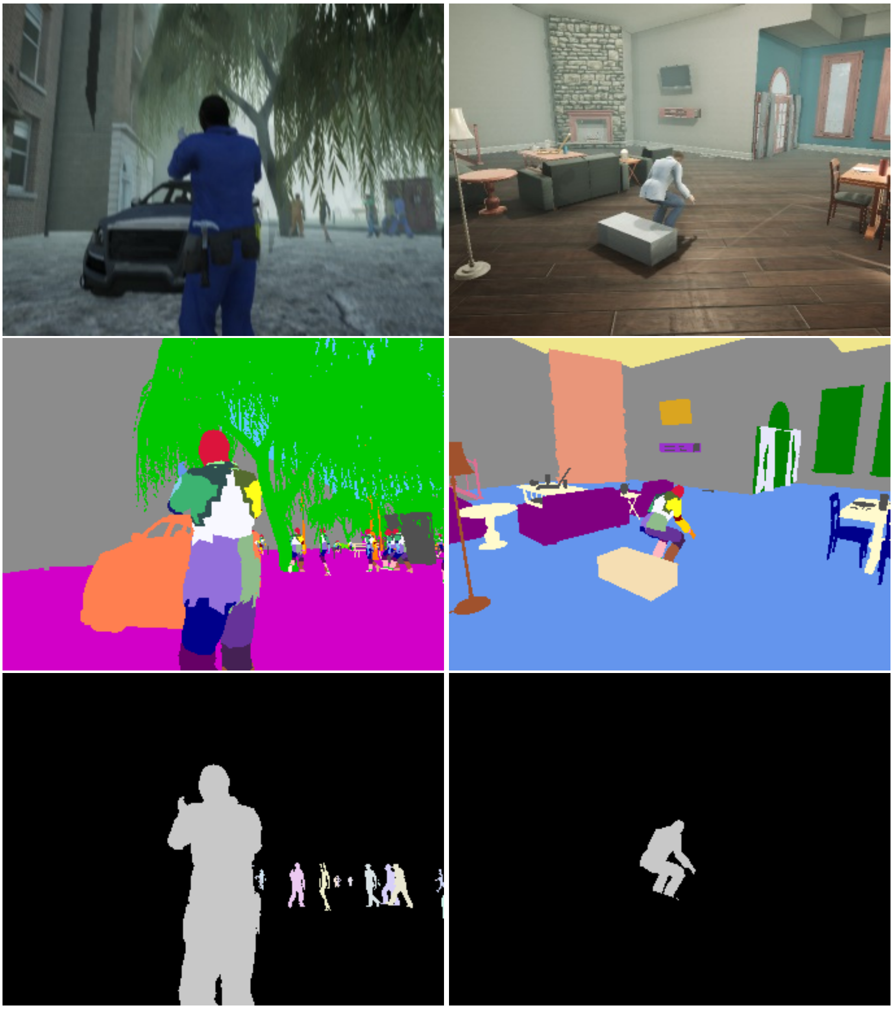


Figure 4.15 – Example frames and data modalities for a synthetic action (car hit, left) and MOCAP-based action (sit, right). From top to bottom: Rendered RGB Frames, Semantic Segmentation, Instance Segmentation.

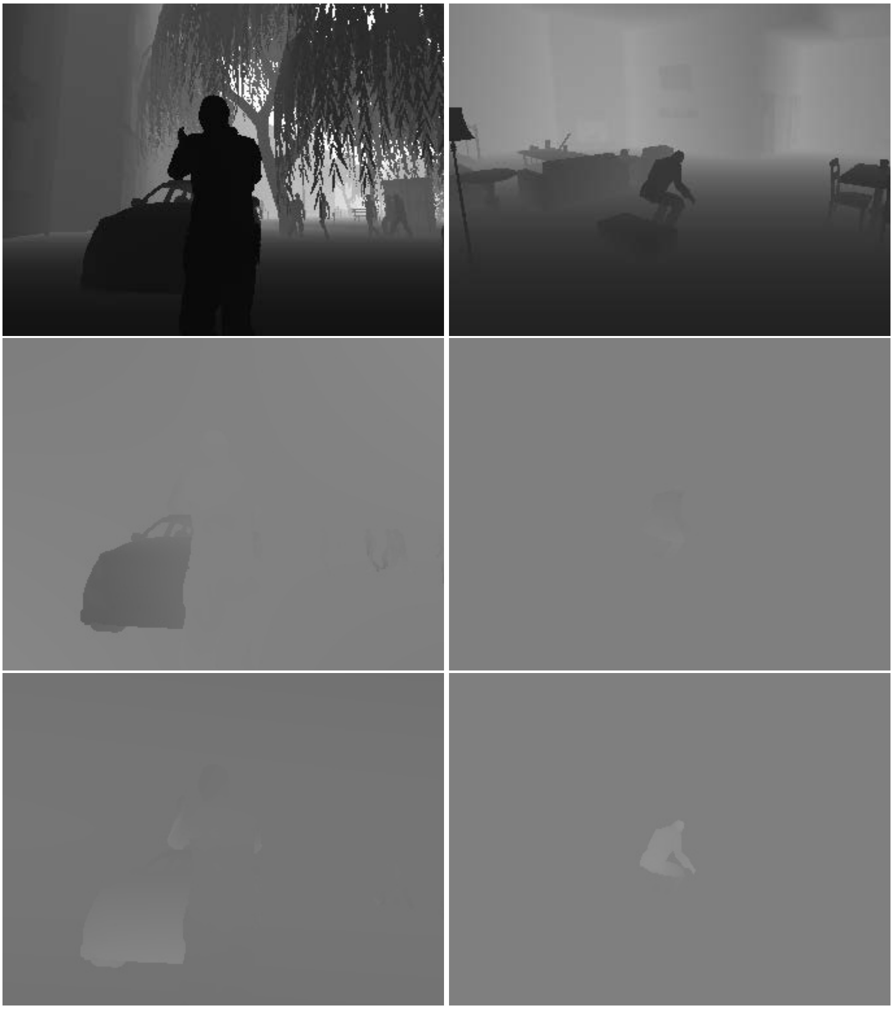


Figure 4.16 – Example frames and data modalities for a synthetic action (car hit, left) and MOCAP-based action (sit, right). From top to bottom: Depth Map, Horizontal Optical Flow, and Vertical Optical Flow. Depth image brightness has been adjusted in this figure to ensure visibility on paper.

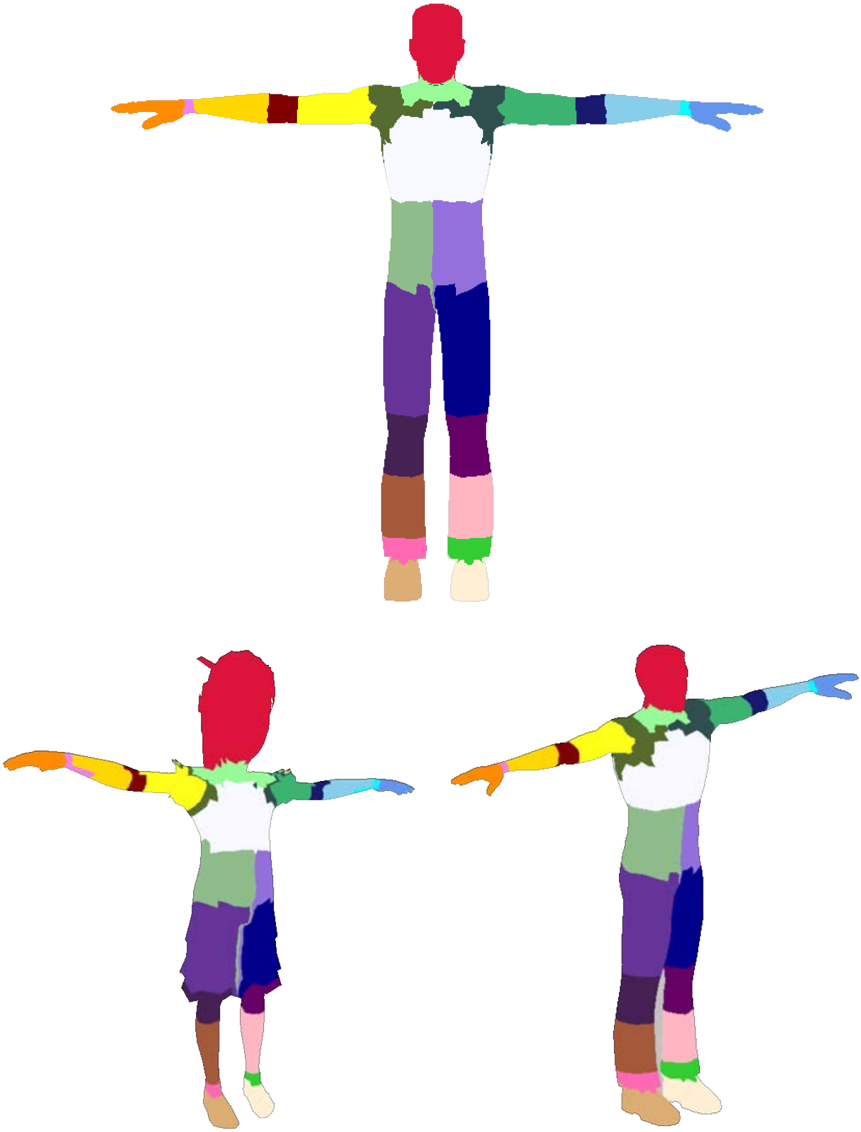


Figure 4.17 – Semantic segmentation ground-truth for human bodies in PHAV. In order to make our approach scalable, body segments are determined automatically for every model through a series of line and distance tests with models in a standardized key position. The spatial resolution of the segments are determined by the resolution of their meshes.

Table 4.4 – Pixel-wise object-level classes in PHAV.

Group		Pixel class	R	G	B	
Virtual KITTI [57]	CityScapes [37]	Road	100	60	100	■
		Building	140	140	140	■
		Pole	255	130	0	■
		TrafficLight	200	200	0	■
		TrafficSign	255	255	0	■
		Vegetation	90	240	0	■
		Terrain	210	0	200	■
		Sky	90	200	255	■
		Car	255	127	80	■
		Truck	160	60	60	■
		Bus	0	139	139	■
		Misc	80	80	80	■
		Tree	0	199	0	■
ADE20k [224]	Indoors	Ceiling	240	230	140	■
		Floor	0	191	255	■
		Chair	72	61	139	■
		Table	255	250	205	■
		Bed	205	92	92	■
		Lamp	160	82	45	■
		Sofa	128	0	128	■
		Window	0	128	0	■
		Door	127	255	212	■
		Stairs	219	112	147	■
		Curtain	230	230	250	■
		Fireplace	233	150	122	■
		Shelf	153	50	204	■
		Bench	245	222	179	■
		Screen	218	165	32	■
		Fridge	255	255	240	■
	Interaction objects	Ball	178	34	34	■
		Baseball Bat	210	105	30	■
		Gun	255	248	220	■
		Golf Club	173	255	47	■
		Hair Brush	224	255	255	■
PHAV-only		Bow	95	158	160	■

Group		Pixel class	R	G	B	
Human	Parts	Head	220	20	60	■
		RightUpperArm	255	255	26	■
		RightLowerArm	255	215	0	■
		RightHand	255	140	0	■
		LeftUpperArm	60	179	113	■
		LeftLowerArm	135	206	235	■
		LeftHand	100	149	237	■
		Chest	248	248	255	■
		RightUpperLeg	102	51	153	■
		RightLowerLeg	164	89	58	■
		RightFoot	220	173	116	■
		LeftUpperLeg	0	0	139	■
		LeftLowerLeg	255	182	193	■
		LeftFoot	255	239	213	■
	Joints	Neck	152	251	152	■
		LeftShoulder	47	79	79	■
		RightShoulder	85	107	47	■
		LeftElbow	25	25	112	■
		RightElbow	128	0	0	■
		LeftWrist	0	255	255	■
		RightWrist	238	130	238	■
		LeftHip	147	112	219	■
		RightHip	143	188	139	■
		LeftKnee	102	0	102	■
		RightKnee	69	33	84	■
		LeftAnkle	50	205	50	■
		RightAnkle	255	105	180	■

Pixel-wise object-level classes in PHAV. Some of the classes have been derived from semantic segmentation labels present in other datasets. These include: CityScapes [37] (mostly for outdoor object classes), Virtual KITTI [57] (which contains a subset of the class labels in CityScapes), and ADE20k [224] (mostly for indoor object classes). The human body has been segmented in 14 parts and 13 joints, for a total of 27 segments. We note that our chosen separation can be combined to recover part separations used in PASCAL-Part [31] and J-HMDB [84] datasets.

Procedural Video Parameters. We also include the internal state of our generator and virtual world at the beginning of the data generation process of each video. This data can be seen as large, sparse vectors that determine the content of a procedurally generated video. These vectors contain the values of all possible parameters in our video generation model, including detailed information about roughly every rigid body, human characters, the world, and otherwise every controllable variable in our virtual scene, including the random seed which will then influence how those values will evolve during the video execution. As such, these vectors include variables that are discrete (*e.g.,* visibility of the clouds), continuous (*e.g.,* x-axis position of the protagonist), piecewise continuous (*e.g.,* time of the day), and angular (*e.g.,* rotation of the Earth). These vectors can therefore be seen as *procedural recipes* for each of our generated videos.

4.4.3 Example frames

In this section, we include random frames for a subset of the action categories in PHAV. These frames show the effect of different variables and motion variations being used (*cf.* Table 4.2). Each frame below is marked with a label indicating the value for different variables during the execution of the video, using the legend shown in Figure 4.18.

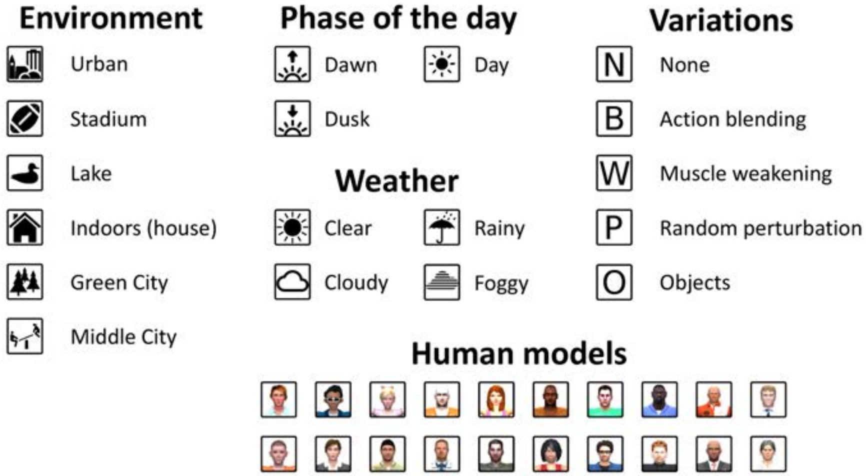


Figure 4.18 – Legend for synthetic action video variations to be used in the frames contained in this section (*cf.* Figures 4.19, 4.20, 4.21, 4.22, and 4.22).

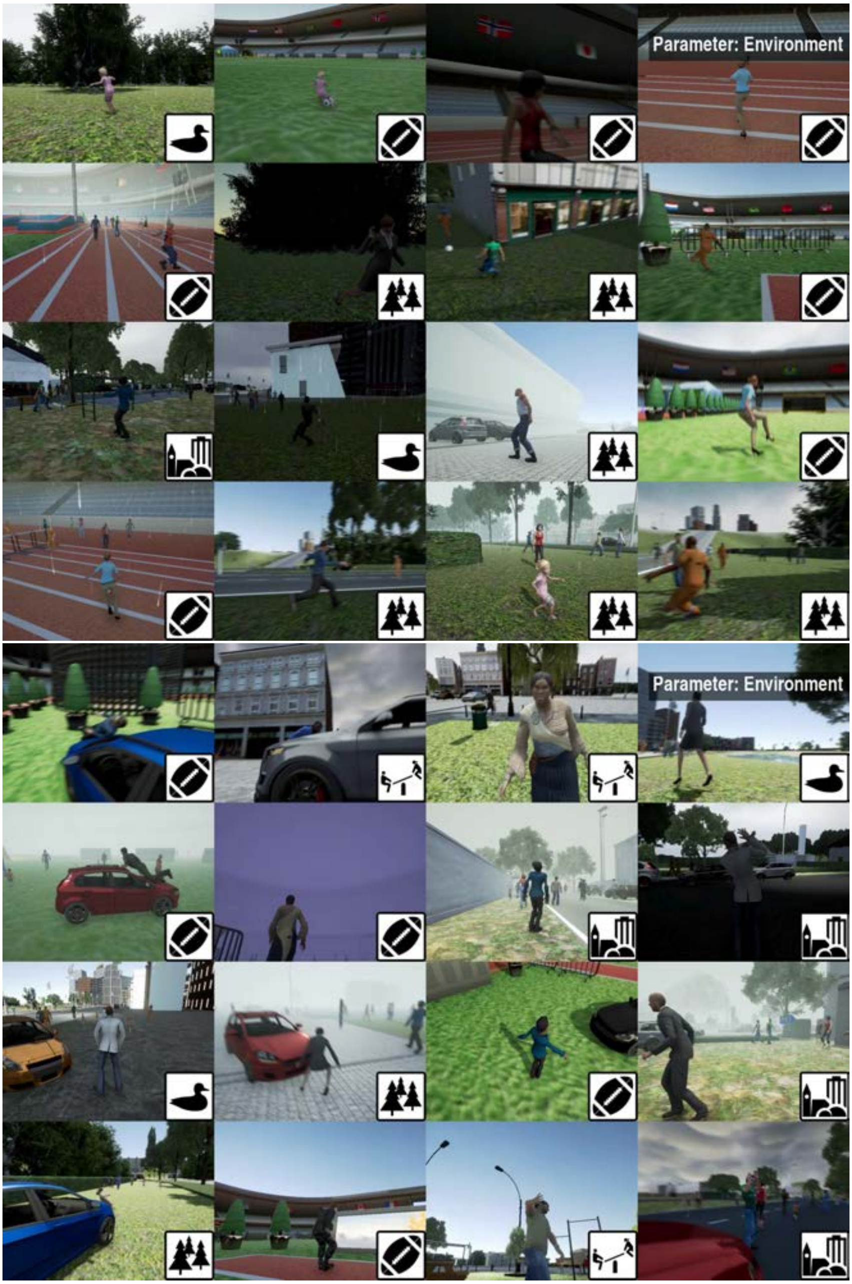


Figure 4.19 – Changing environments. Top: *kick ball*, bottom: *synthetic car hit*.

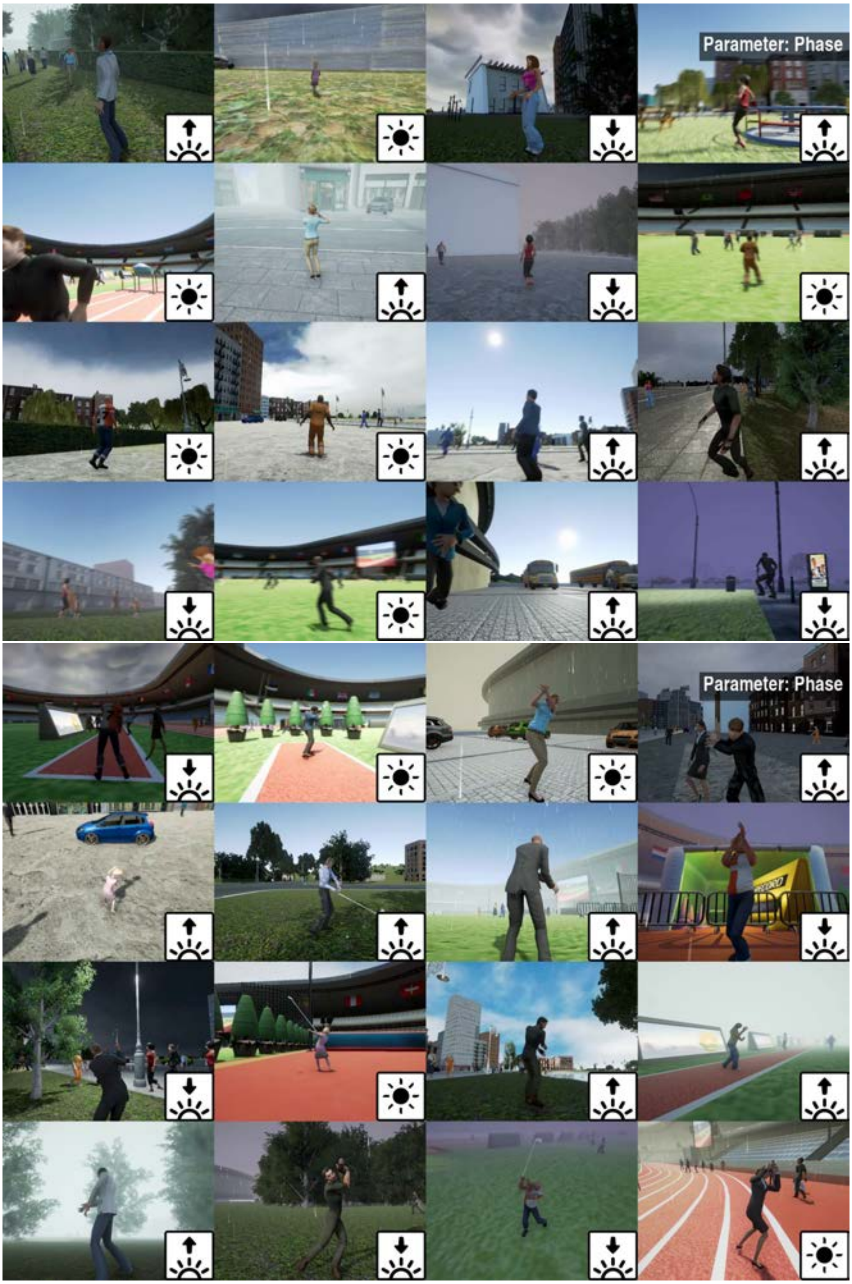


Figure 4.20 – Changing phases of the day. Top: *run*, bottom: *golf*.

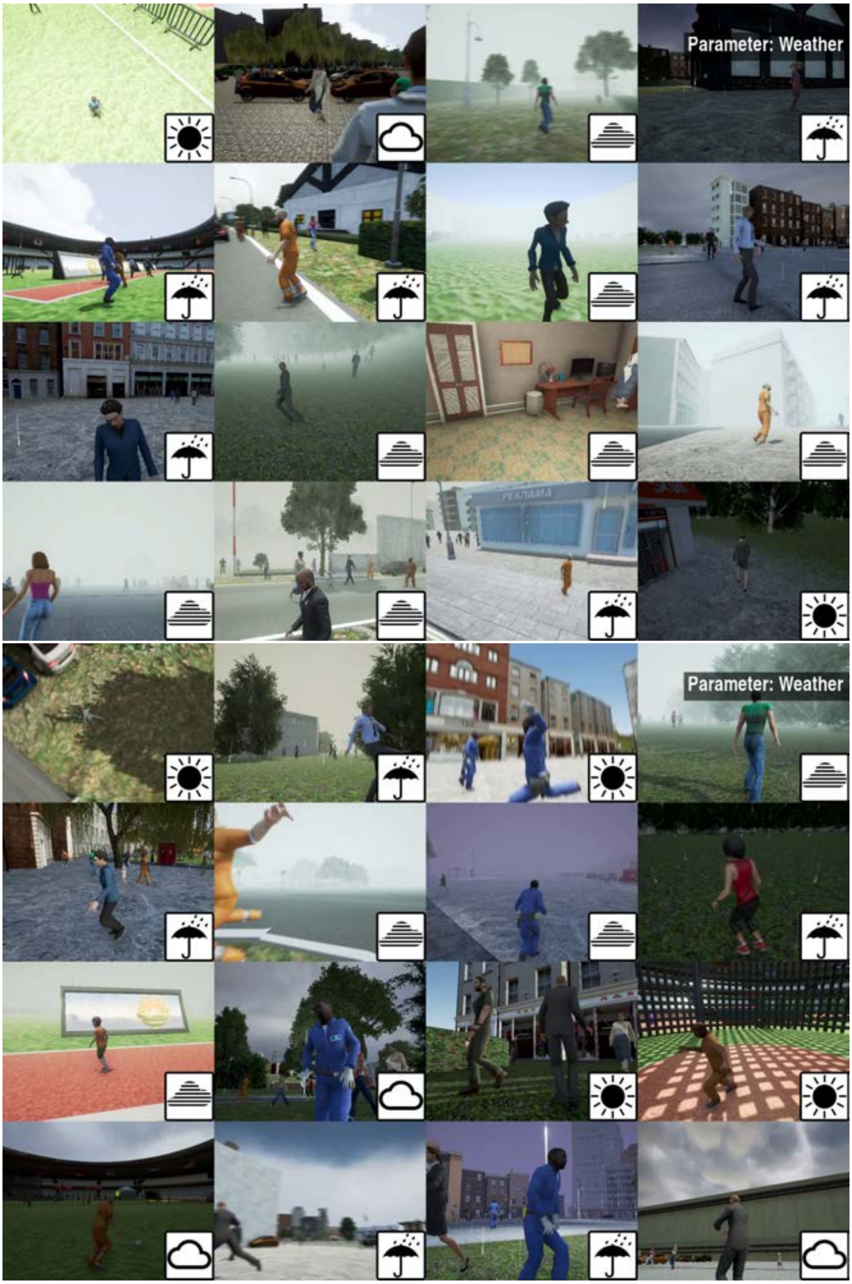


Figure 4.21 – Changing weather. Top: *walk*, bottom: *kick ball*.

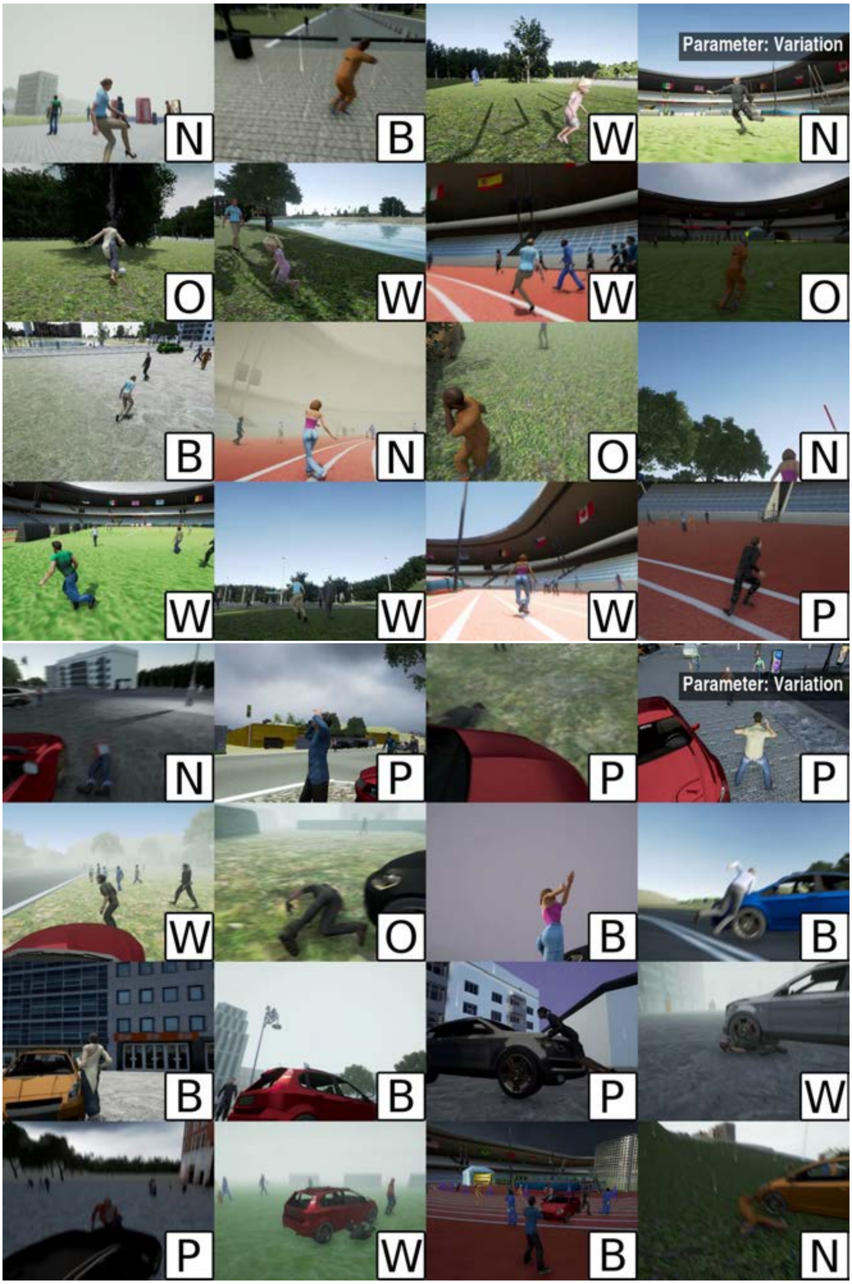


Figure 4.22 – Changing motion variations. Top: *kick ball*, bottom: *synthetic car hit*.

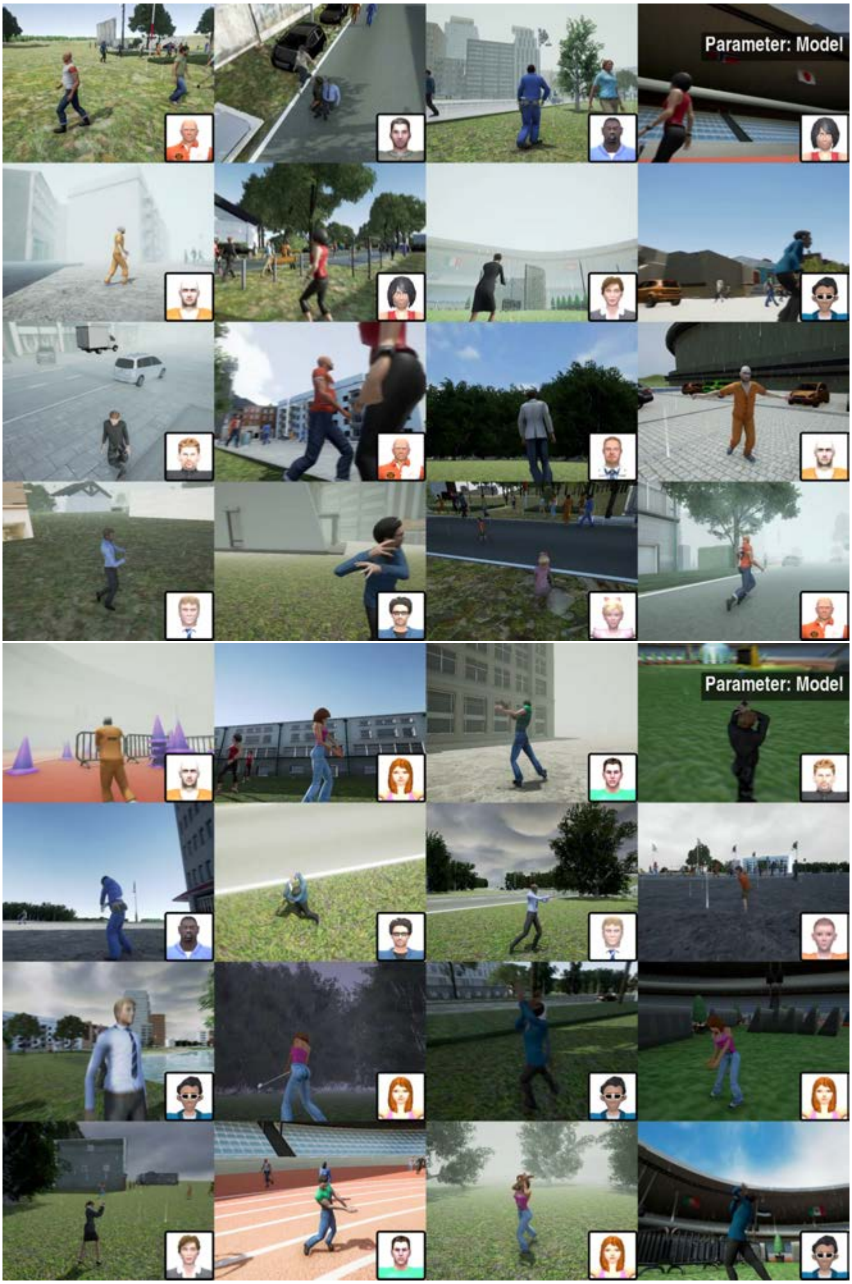


Figure 4.23 – Changing human models. Top: *walk*, bottom: *golf*.

4.5 Summary of the chapter

In the work presented in this chapter, we have introduced PHAV, a large synthetic dataset for action recognition based on a procedural generative model of videos.

Our approach uses techniques from computer graphics (procedural generation) to generate data collections that can be used to train state-of-the-art deep learning models for action recognition. This opens interesting new perspectives for video modeling and understanding, including action recognition models that can leverage algorithmic ground truth generation for optical flow, depth, semantic segmentation, or pose, or the combination with unsupervised generative models like VGAN for dynamic background generation, domain adaptation, or real-to-virtual world style transfer [60, 227].

Although our model does not learn video representations like VGAN [198] (*cf.* Section 2.4), it can generate many diverse training videos thanks to its grounding in strong prior physical knowledge about scenes, objects, lighting, motions, and humans. Our detailed graphical model shows how a complex video generation can be driven through a few simple parameters. We have also shown that generating action videos while still taking the effect of physics into account is a challenging task.

In the next chapter, we will provide quantitative evidence that our procedurally generated videos can be used as a complement to small training sets of manually labeled real-world videos, as well as propose new deep learning models that leverage the extra data modalities present in PHAV.

5 Learning more about the real world with synthetic action videos

Art begins in imitation and ends in innovation.

Mason Cooley

In the previous chapter, we investigated the generation of synthetic training data for action recognition, as it has recently shown promising results for a variety of other computer vision tasks. In this chapter, we introduce two deep multi-task representation learning architectures that are able to leverage synthetic data to learn more about the real world by mixing synthetic and real videos, even if the action categories differ. We also show how to integrate different data modalities and ground-truths during learning, when those sources are available only for the virtual task. Our experiments on the UCF-101 and HMDB-51 benchmarks suggest that combining our large set of synthetic videos with small real-world datasets can boost recognition performance, significantly outperforming state-of-the-art unsupervised generative models of videos which have been fine-tuned for action classification.

5.1 Introduction

DEEP LEARNING for human action recognition in videos is making significant progress, but is slowed down by its dependency on expensive manual labeling of large video collections. Due the lack of large quantities of labeled video data, recent deep learning architectures for action are often pre-trained on large-scale image datasets (*e.g.,* ImageNet, [170]), even for modalities that are exclusive for video (*e.g.,* optical flow, [208]).

Addressing this issue, in Chapter 4 we have proposed an interpretable parametric generative model of human action videos that relies on procedural generation and other computer graphics techniques of modern game engines. We then used it to generate a diverse and physically plausible dataset of human action videos, called PHAV for “Procedural Human Action Videos”, containing a total of 39,982 videos, with more than 1,000 examples for each action of 35 categories.

In this chapter, we now investigate approaches to *leverage* these *synthetic human action videos* to train deep action recognition models. Our experiments suggest that our procedurally generated action videos can complement scarce real-world data. Furthermore, we also look into ways to leverage the multiple data modalities that are available in PHAV (*cf.* Section 4.4.2). Exploring multiple data modalities has been shown to be an effective approach to achieve better and more efficient models in both image-based [25, 76, 86, 128, 210, 214, 218], and video-based action recognition [81, 181, 208, 213, 230], *e.g.,* by taking optical flow as an input for one of the streams in Simonyan *et al.* 's Two Stream architecture [170]. In particular, [84] has shown that having access to ground-truth data for human body parts can lead to significant improvements in the performance of handcrafted action recognition models, while the recent work of [230] has shown that incorporating localized information about human actors, such as joint localization, pose estimation, or parsing of human body parts through an extra modality could also prove beneficial for deep models. This motivates us to explore ways to incorporate such data our deep models for action recognition even though it may be only available in PHAV. We focus in particular on human parsing, *i.e.* segmenting human body parts, because this information is highly relevant for human action recognition.

The **first contribution** of this chapter is a *deep multi-task learning architecture that can learn from real-world and synthetic data sources at the same time.* To allow for generic use, and as predefined procedural action categories may differ from real-world target ones, we propose multi-task learning architectures based on the recent Temporal Segment Network [208] (TSN). We call our first model *Cool-TSN* (*cf.* Figure 5.1) in reference to the “cool world” of [193], as we mix both synthetic and real samples at the mini-batch level during training. Our experiments show that our synthetic human action videos can significantly improve action recognition accuracy, especially with small real-world training sets, in spite of differences in appearance, motion, and action categories.

The **second contribution** of this chapter is a *deep multi-task learning architecture that can learn from multiple data modalities even though some are not available for real-world samples.* To demonstrate the usefulness of the extra modalities in our dataset, which may not be available for target datasets and certainly not at testing time while operating in the real world, we propose an *end-to-end Human Parsing TSN* (HPTSN). We then proceed to give it the same “cool world” treatment as in our Cool-TSN, *i.e.* mixing synthetic and real samples in the same mini-batch. This network uses the semantic segmentation ground-truth from PHAV to induce known useful internal representations [84, 230] inside a deep network to obtain better classification performance. We call our second model *Cool-HPTSN* (*cf.* Figure 5.2), creating a model which is inspired by both the recent DeepLab [31] architecture for semantic segmentation as well as the TSN. We show that these networks can be used to replace the flow stream in Two-Stream Network architectures, which may be advantageous in case flow algorithms are too costly to compute or if the sequence of frames have been taken too far apart and optical flow cannot be computed.

The **third contribution** of this chapter is an *experimental quantitative validation of our dataset* described in Chapter 4. We learn end-to-end action recognition models for real-world target categories by combining a few examples of labeled real-world videos with a large number of procedurally generated videos. We show that, although the synthetic examples differ in statistics and categories, their realism, quantity, and diversity can act as a strong prior and regularizer against overfitting, and enable representation learning with few manually labeled real videos.

A recent alternative to our procedural generative model that also does not require manual video labeling is the unsupervised Video Generative Adversarial Network (VGAN) of [198], which we recall from Section 2.4. Instead of leveraging prior structural knowledge about physics and human actions, the authors view videos as tensors of pixel values and learn a two-stream GAN on 5,000 hours of unlabeled Flickr videos. This method focuses on tiny and short videos, and on capturing scene motion assuming a stationary camera. This architecture can be used for action recognition in videos when complemented with prediction layers fine-tuned on labeled videos. Compared to this approach, our proposal allows to work with any state-of-the-art discriminative architecture, as video generation and action recognition are completely decoupled steps. Moreover, we can decide what specific actions/scenarios/camera-motions to generate, enforcing diversity thanks to our interpretable parametrization. For these reasons, we show in Section 5.4 that, given the same amount of labeled videos, our model achieves nearly two times the performance of the unsupervised features shown in [198].

Regarding our approach for leveraging extra modalities, the closest work to ours is that of [230], who also used a third stream based on semantic segmentation. The authors used a Fast-Net architecture [130] to obtain human body parts from RGB frames, encoded these body parts into RGB images using a mapping of pre-defined RGB values, and then classified these sequences of re-encoded RGB frames using C3D [187] (*cf.* Section 2.3.1). The authors fuse representations from all streams via a Markov chain, obtaining a sequential refinement of the action labels. Our model differs significantly from this work in several aspects: (i) while [230] constructs ground-truth for the segmentation network by approximating part shapes with polygons and ellipses, we leverage a synthetic data source in order to output a more precise pixel-level segmentation of human body parts; (ii) we incorporate uncertainty into the parsing result, and do so in an efficient manner using color blending;

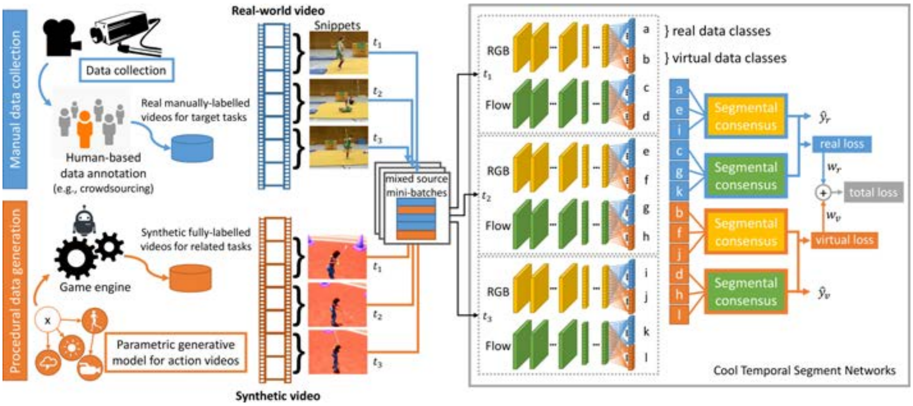

Manual data collection
Data collection
Real manually-labelled videos for target tasks
Human-based data annotation (e.g., crowdsourcing)
Procedural data generation
Synthetic fully-labelled videos for related tasks
Game engine
Parametric generative model for action videos
Real-world video
Snippets
Synthetic video
mixed source mini-batches
RGB
Flow
} real data classes
} virtual data classes
Segmental consensus
real loss
total loss
virtual loss
Cool Temporal Segment Networks

Figure 5.1 – Our "Cool-TSN" deep multi-task learning architecture for end-to-end action recognition in videos.

(iii) we train our models end-to-end. Moreover, we note that the advancements reported in both [230] and in this chapter are not mutually exclusive but in fact, orthogonal: As our experiments show that complementing real with synthetic data leads to observable performance improvements in multiple tasks, this suggests that our approach for multi-task and multi-modality learning for leveraging synthetic data could be used to obtain improved versions of the models presented in [230]. In the same way, the approach for multi-stream chaining described in [230] could also be used to modify our TSN-based architectures to obtain the same sequential refinement of action labels.

The remainder of this chapter is organized as follows: In Section 5.2 and 5.3 we present, respectively, our Cool-TSN and Cool-HPTSN deep learning algorithms for action recognition. We then report our quantitative experiments in Section 5.4, measuring the usefulness of PHAV and the effectiveness of our proposed models. Finally, we draw our conclusions in Section 5.5.

5.2 Cool Temporal Segment Networks

We propose to demonstrate the usefulness of our PHAV dataset via deep multi-task representation learning. Our main goal is to learn end-to-end action recognition models for real-world target categories by combining a few examples of labeled real-world videos with a large number of procedurally generated videos for different surrogate categories in PHAV.

5.2.1 Temporal Segment Networks

Wang *et al.* 's TSN architecture [208] can be seen as an improvement over the original Two-Stream Network (2SN) architecture of [170] (*cf.* Section 2.3.3). Both networks are based on two independent networks, each of which specialized in different modalities from the action videos (spatial and temporal). The TSN incorporates important additions that bring performance improvements over Simonyan *et al.* 's original work: (i) a deeper Inception architecture using modern deep learning improvements such as Batch-Normalization and DropOut, (ii) a better initialization procedure for the temporal stream, and (iii) an explicit approach for capturing long-range temporal structures in a video. This approach, which inspires the name "Temporal Segment Networks", consists in taking a fixed number of k ordered video segments, learning predictions from those segments jointly, and then combining these predictions using a segmental consensus function **G**. For all models described in this chapter, we use the same parameters as [208]: a number of segments $k = 3$, and the consensus function:

$$\mathbf{G} = \frac{1}{k} \sum_{t=1}^{k} \mathscr{F}(T_t; \boldsymbol{W}), \tag{5.1}$$

where $\mathscr{F}(T_t; \boldsymbol{W})$ is a function representing a CNN architecture with weight parameters $\boldsymbol{W}$ operating on short snippet T_t from video segment t. We use this architecture as a base for all of our models proposed in this chapter.

5.2.2 Multi-task learning in a Cool World

Figure 5.1 depicts our learning algorithm inspired by the Two Streams approach of [170], but adapted with the recent state-of-the-art TSN method of [208], and coupled with the "cool worlds" technique of [193], *i.e.* mixing real and virtual data during training. As illustrated in Figure 5.1, the main differences between TSN and our proposed "Cool-TSN" architecture are at both ends of the training procedure: (i) the mini-batch generation, and (ii) the multi-task prediction and loss layers.

Cool mixed-source mini-batches. Inspired by [158, 193], we build mini-batches containing a mix of real-world videos and synthetic ones. Following [208], we build mini-batches of 256 videos divided in blocks of 32 dispatched across 8 GPUs for efficient parallel training using MPI[1]. Each block contains 10 random synthetic videos and 22 real videos in all our experiments with our Cool-TSN, as we observed

[1] https://github.com/yjxiong/temporal-segment-networks

it roughly balances the contribution of the different losses during backpropagation. Note that although we could use our ground-truth synthetic flow for the PHAV samples in the motion stream, we use the same fast optical flow estimation algorithm as [208] (TV-L^1 [219]) for all samples in order to fairly estimate the usefulness of our generated videos.

Multi-task prediction and loss layers. We use a multi-task approach to ensure our architecture can be trained on multiple datasets whose action categories are not required to intersect. Starting from the last feature layer of each stream, our architecture splits into two parallel branches: one of which is used to categorize videos from PHAV and the other which categorizes videos from the real-world dataset. Each branch consists of its own fully-connected prediction, segmental consensus, and softmax loss layers. We obtain the following multi-task loss:

$$\mathscr{L}(\boldsymbol{y}, \boldsymbol{G}) = \sum_{z \in \{real, virtual\}} \delta_{\{y \in C_z\}} w_z \mathscr{L}_z(\boldsymbol{y}, \boldsymbol{G}) \tag{5.2}$$

with

$$\mathscr{L}_z(\boldsymbol{y}, \boldsymbol{G}) = -\sum_{i \in C_z} y_i \left(G_i - \log \sum_{j \in C_z} \exp G_j \right) \tag{5.3}$$

where z indexes the source dataset (real or virtual) of the video, w_z is a loss weight (we use the relative proportion of z in the mini-batch, *i.e.* $w_{real} = 22/32$ and $w_{virtual} = 10/32$), C_z denotes the set of action categories for dataset z, and $\delta_{\{y \in C_z\}}$ is the indicator function that returns 1 when label y belongs to C_z and 0 otherwise. We use standard SGD with backpropagation to minimize our objective, and as every mini-batch contains both real and virtual samples, every iteration is guaranteed to update both the shared feature layers and the separate prediction layers for the real and virtual videos in a common descent direction. We discuss the setting of the hyper-parameters in Section 5.4.

5.3 Cool Human Parsing Temporal Segment Networks

In this section we now show our approach for multi-modality learning with PHAV. Our goal is to obtain end-to-end trainable action recognition models for real-world target categories that leverage multiple data modalities during training, even if some of those modalities are not available in the real-world datasets.

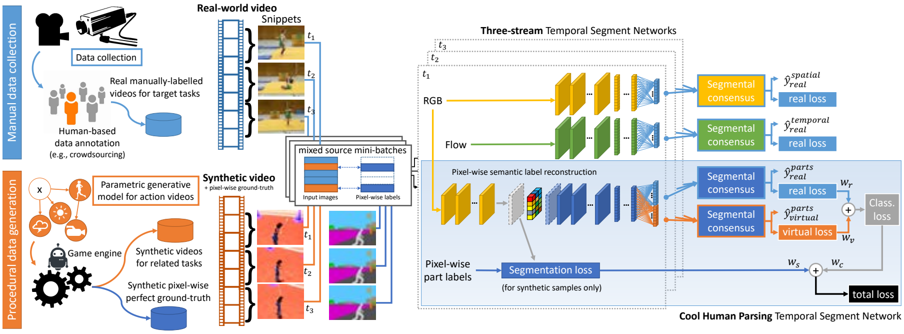

Manual data collection
Data collection
Real manually-labelled videos for target tasks
Human-based data annotation (e.g., crowdsourcing)
Real-world video
Snippets
t_1
t_2
t_3
mixed source mini-batches
Input images
Pixel-wise labels
Procedural data generation
x
Parametric generative model for action videos
Game engine
Synthetic videos for related tasks
Synthetic pixel-wise perfect ground-truth
Synthetic video
+ pixel-wise ground-truth
Three-stream Temporal Segment Networks
RGB
Flow
Pixel-wise semantic label reconstruction
Segmental consensus
$\hat{y}^{spatial}_{real}$
real loss
Segmental consensus
$\hat{y}^{temporal}_{real}$
real loss
Segmental consensus
$\hat{y}^{parts}_{real}$
real loss
w_r
Segmental consensus
$\hat{y}^{parts}_{virtual}$
virtual loss
w_v
Class. loss
Pixel-wise part labels
Segmentation loss
(for synthetic samples only)
w_s
w_c
total loss
Cool Human Parsing Temporal Segment Network

Figure 5.2 – Our full architecture for human action recognition in videos. The third stream can be trained end-to-end from multiple datasets, even when semantic segmentation ground-truth is available for only some of them.

5.3.1 Human Parsing Temporal Segment Networks

Inspired by [84, 230], we extend the TSN with fully-convolutional layers in order to incorporate body part localization as an internal representation within a third stream. Our "HPTSN" (Figure 5.2, highlighted in blue) can be divided in three parts: (i) the human semantic parsing layers; (ii) the pixel-wise semantic color transformation layer; and (iii) the action classification layers.

Human parsing layers. Since we deal with what is essentially a semantic segmentation problem, we base the first part of our architecture on the DeepLab network of [31] with few modifications. Instead of using Atrous Spatial Pyramid Pooling (ASPP), we use a single model with a dilation of 6, which was shown in our experiments to perform well for parsing the PASCAL-Part dataset [32] (*cf.* Section 2.1.1). We also modify it to have the same output resolution of 340 × 256 as expected by Wang *et al.* 's TSN, and to classify each pixel from the input RGB frames into 11 body part categories (*cf.* Table 5.1).

The output of the human parsing layers is a per-pixel probability map $\boldsymbol{X}$ of size (c, h, w), where c is the number of semantic classes, h is the input image height, and w is the input image width. In this work, we take $c = 11$, $w = 340$, and $h = 256$. Each value $\boldsymbol{X}_{ijk} \geq 0$ represents the probability that the pixel at row i, and column j, belongs to class k, with $\sum_{k=1}^{C} \boldsymbol{X}_{ijk} = 1$ for any $1 \leq i \leq w$, $1 \leq j \leq h$.

Pixel-wise semantic color transformation layer. After the human parsing layers have produced class assignment scores for each pixel in a frame, we map the prediction vector for each pixel into an RGB value using a color transformation layer. This layer produces a soft-assignment of colors through color blending. This soft-assignment enables our model to remain fully differentiable and thus amenable to end-to-end training using backpropagation. We also show in the experimental section that using this soft segmentation can have a positive effect in action classification performance.

Our color transform is implemented as a standard cross-correlation layer using 1×1 cross-correlations where the filter weight matrix corresponds to a matrix of RGB colors. The output value of a cross-correlation layer with 1×1 filters, input size (c, h, w) and output (d, h, w) can be described as:

$$\boldsymbol{Y}_j = \boldsymbol{b}_j + \sum_{k=1}^{c} \boldsymbol{W}_{jk} \star \boldsymbol{X}_k \tag{5.4}$$

where $\star$ is the valid 2D cross-correlation operator, b is the bias vector, d is the number of output channels, $1 \leq j \leq d$ is the output channel index. Since we use this convolution to transform c channels to RGB colors, we always consider $d = 3$. The RGB image generated by the color transformation layer can therefore be expressed as:

$$\begin{aligned}
\boldsymbol{Y}^{(r)} &= b^{(r)} + \sum_{k=1}^{11} \boldsymbol{W}_k^{(r)} \star \boldsymbol{X}_k \\
\boldsymbol{Y}^{(g)} &= b^{(g)} + \sum_{k=1}^{11} \boldsymbol{W}_k^{(g)} \star \boldsymbol{X}_k \\
\boldsymbol{Y}^{(b)} &= b^{(b)} + \sum_{k=1}^{11} \boldsymbol{W}_k^{(b)} \star \boldsymbol{X}_k
\end{aligned} \tag{5.5}$$

where $\boldsymbol{W}_k^{(r)}$, $\boldsymbol{W}_k^{(g)}$, and $\boldsymbol{W}_k^{(b)}$ correspond to the amount of red, green, and blue values in the color representation for class k considering a RGB coloring system (*cf.* Tables 4.4 and 5.1 for example representations). Since the RGB image output by the color encoding layer will be used to feed a subsequent classification network, we set the bias vectors b to the negative of the average pixel values in the target dataset. With this, we implement the mean vector subtraction which is expected for the inputs of the pre-trained layers of the TSN.

Table 5.1 – Human body part classes used in our experiments.

Semantic class	Classes in PHAV	Classes in PASCAL-Part	
1	(background)	(background)	🟦
2	Neck Head	Neck Head Left Ear Left Eye Left Eyebrow Right Ear Right Eye Right Eyebrow Mouth Hair Nose	🟥
3	Chest LeftShoulder RightShoulder LeftHip RightHip	Torso	🟩
4	RightUpperArm	Right Upper Arm	🟨
5	RightLowerArm RightElbow RightWrist RightHand	Right Lower Arm Right Hand	🟧
6	LeftUpperArm	Left Upper Arm	🟩
7	LeftElbow LeftLowerArm LeftWrist LeftHand	Left Lower Arm Left Hand	🟦
8	RightUpperLeg	Right Upper Leg	🟪
9	RightLowerLeg RightKnee RightAnkle RightFoot	Right Lower Leg Right Foot	🟥
10	LeftUpperLeg	Left Upper Leg	🟦
11	LeftKnee LeftLowerLeg LeftAnkle LeftFoot	Left Lower Leg Left Foot	🟩

Human body part classes used in our experiments. The first column corresponds to the human parts which are identified (parsed) by our human semantic parsing layers. These parts correspond to different groupings of annotations depending on the dataset from which frames come from (PHAV in the middle, and PASCAL-Part on the right). The last column shows the color coding used in Figure 5.3.

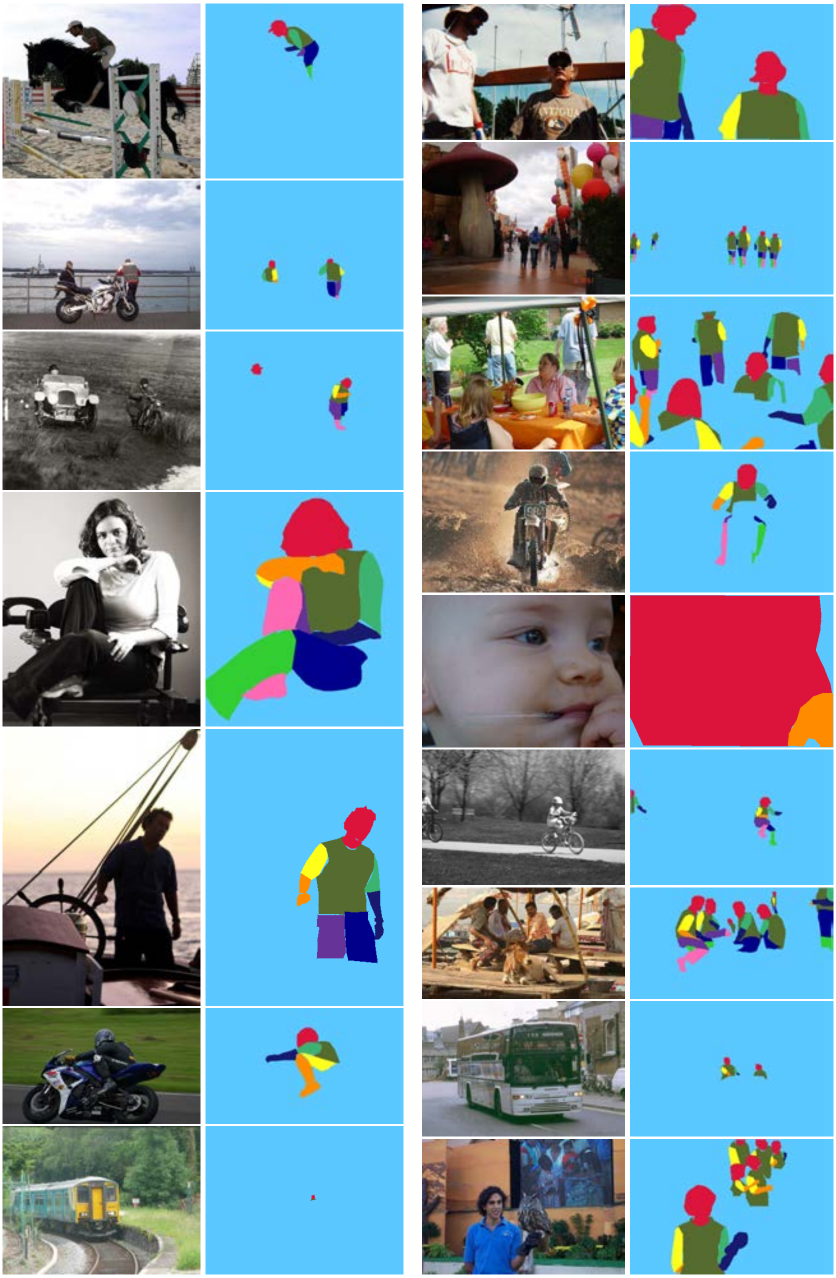


Figure 5.3 – Example images from PASCAL-VOC and their associated body ground-truth using the pixel class labels in Table 5.1. For example images from PHAV, *cf.* Figure 4.15.

Action classification layers. The action classification layers use the same BN-Inception architecture used in [208] to obtain action class labels for each frame. Similar to [208] we also apply random multi-scale cropping, horizontal flipping, multi-scale and aspect-ratio jittering, obtaining frames of dimensions 224 × 224. We note that unlike other models in the literature, these operations are not pre-processing steps but rather are integrated in our architecture: The semantic segmentation network is sensitive to spatial context, and therefore cropping video regions too early would result in a smaller spatial resolution in the middle of our architecture that is difficult to upscale effectively without generating visual artifacts. Therefore, these operations need to be performed inside our network, after the semantic segmentation step but before actual classification.

5.3.2 Multi-modality learning in a Cool World

We now describe our approach for training our Cool-HPTSN model with both real and synthetic data. The main points of our approach are: (i) the mini-batch generation, (ii) the extended multi-modality multi-task prediction and loss layers, and (iii) the initialization of the color transformation layer.

Cool mixed-source and mixed-modality mini-batches. To learn our human parsing network, we again build mini-batches with a mix of real-world and synthetic videos. Based on early experiments, we build mini-batches of 64 videos. Each mini-batch contains 20 synthetic videos and 44 real-world videos.

However, unlike in the case for Cool-TSN, our synthetic and real-world videos now have different associated ground-truths: while the real-world videos have only action class ground-truth labels, the synthetic videos have both action class *and* human parsing ground-truth labels. This means that processing synthetic videos require loading at least two times more images (as human parsing ground-truth is a pixel-wise ground-truth and therefore represented as images, *cf.* Figure 5.3 for example representations) than when processing real-world videos.

Thanks to a flexible base implementation[2], we employ an assymetric division over multiple GPUs to balance memory usage since it is possible to process more real-world videos at the same time than synthetic ones due their different memory requirements. We therefore distribute the 44 real-world videos among other 2 GPUs, and the remaining 20 synthetic videos among 2 other GPUs. Processing synthetic videos also require more memory due an additional computational branch that we introduce in our architecture to incorporate the human parsing objective in our loss function, as we describe below.

[2] https://github.com/yjxiong/tsn-pytorch

Multi-task and multi-modality loss layers. In addition to the two separate branches for target action classes from real and synthetic datasets (*cf.* Equation 5.2), we now introduce a third computational path that branches from the last human parsing layer. First, let us denote the scores as:

$$\boldsymbol{H}_{tijk} = \mathscr{H}(T_t; \boldsymbol{W})_{ijk} \tag{5.6}$$

where $\mathscr{H}$ is a function representing our human parsing layers with weight parameters $\boldsymbol{W}$, giving the probability score of class k for each pixel (i, j), in the input frame T_t from video segment t. Using this notation, we can write our extended multi-task loss as:

$$\mathscr{L}'(y, \mathbf{G}, \boldsymbol{P}) = \mathscr{L}(y, \mathbf{G}) + \delta_{\{y \in C_{virtual}\}} w' \mathscr{L}'(\boldsymbol{P}) \tag{5.7}$$

with

$$\mathscr{L}'(\boldsymbol{P}) = -\sum_{t=1}^{k} \sum_{i=1}^{w} \sum_{j=1}^{h} \sum_{k=1}^{c} \boldsymbol{P}_{tijk} \left(\boldsymbol{H}_{tijk} - \log \sum_{k'=1}^{c} \exp \boldsymbol{H}_{tijk'} \right) \tag{5.8}$$

where $\boldsymbol{P}_{tijk}$ is the ground-truth class for pixel (i, j) for the frame from video segment t, w' is a loss weight for the segmentation task, $C_{virtual}$ denotes the set of action categories for the synthetic videos, and all other variables are defined as in Equations 5.2 and 5.3.

Semantic color encoding. The choice for the colors used in the semantic color transformation layer can have a significant impact on performance. We investigate five different choices for determining those colors.

- Using a fixed set of 11 randomly-chosen colors from the RGB space;
- Partitioning the RGB color space and obtaining 11 color centroids using clustering via EM. We use EM to search for optimal partitionings of the 3-dimensional RGB space, *i.e.* divisions containing partitions that are maximally dissimilar to each other but maximally similar internally [155]. We initialize the algorithm using a uniform division of the RGB space, *i.e.* by partitioning it into 2 × 2 × 2 cells;

- Using a fixed set of colors established by the ISCC-NDB color designation system [93]. This system was created as an effort to standardize color names and includes 10 chromatic and 3 non-chromatic categories (white, black, and grey). We use the values for the 10 chromatic categories and the grey category as reported in [24];
- Using the set of 11 colors from the ISCC-NDB system as the initialization point for the clustering approach described above;

In each case we randomly assign categories to colors (*cf.* Table 5.4 for examples of these encodings). We discuss the performance of each of these approaches in the following experimental section. We then consider the best of these approaches and use it as an initialization point for the color transformation layer in our end-to-end network, therefore fine-tuning those colors to the action classification datasets using backpropagation.

Three-stream architecture. Following [170, 208], we combine the outputs of the RGB, optical flow and human parsing networks using score-level fusion at testing time. To compute the final class prediction for a video, we start by taking 25 equally-spaced segments from it as in [170]. For the RGB and human parsing streams, this corresponds to 25 individual frames, while for the flow stream this corresponds to 25 stacks of 5 horizontal inverleaved with 5 vertical optical flow frames. For RGB and flow, we then extract 10 crops of size 224 × 224 from each frame, corresponding to their 4 corners, their center, and the mirrored version of these. For the human parsing stream, HPTSN performs these crops internally before its action classification layers (*cf.* Section 5.3.1). We compute the action classification scores for each of these crops, averaging per segment. Following [208] we use the average as the consensus function and therefore average the prediction scores across all segments. We then compute the softmax of those averages, obtaining one vector of probabilities for each of the three streams. Finally, we take a weighted average of these probability vectors as the final scores vector for the video.

5.4 Experiments

In this section, we detail our action recognition experiments on widely used real-world video benchmarks. We quantify the impact of multi-task representation learning with our procedurally generated PHAV videos on real-world accuracy, in particular in the small labeled data regime. We then present our experiments with extra modalities only available in PHAV, measuring the impact of supervised training for different parts of our human parsing architecture. We also compare our method with the state of the art on both fully supervised methods and methods based on the fine-tuning of unsupervised generative models of video.

5.4.1 Temporal Segment Networks

In our first experiments (*cf.* Table 5.2), we reproduce the performance of the original TSN on UCF-101 and HMDB-51 using the same learning parameters as in [208]. For simplicity, we use neither cross-modality pre-training nor a third warped optical flow stream like [208], as their impact on TSN is limited with respect to the substantial increase in training time and computational complexity, degrading only by -1.9% on HMDB-51, and -0.4% on UCF-101.

We also estimate performance on PHAV separately, and fine-tune PHAV networks on target datasets. Training and testing on PHAV yields an average accuracy of 82.3%, which is between that of HMDB-51 and UCF-101. This sanity check confirms that, just like real-world videos, our synthetic videos contain both appearance and motion patterns that can be captured by TSN to discriminate between our different procedural categories. We use this network to perform fine-tuning experiments (TSN-FT), using its weights as a starting point for training TSN on UCF101 and HMDB51 instead of initializing directly from ImageNet as in [208]. We discuss learning parameters and results below.

5.4.2 Cool Temporal Segment Networks

In Table 5.2 we also report results of our Cool-TSN multi-task representation learning, (Section 5.2.2) which learns the additional task of classifying videos from PHAV. We stop training after $3,000$ iterations for RGB streams and $20,000$ for flow streams, all other parameters as in [208]. Our results suggest that leveraging PHAV through either Cool-TSN or TSN-FT yields recognition improvements for both modalities in all datasets, with advantages in using Cool-TSN especially for the smaller HMDB-51. This provides quantitative experimental evidence supporting our claim that procedural generation of synthetic human action videos can indeed act as a strong prior (TSN-FT) and regularizer (Cool-TSN) when learning deep action recognition networks.

We further validate our hypothesis by investigating the impact of reducing the number of real world training videos, with or without the use of PHAV. Our results reported in Figure 5.4 and Table 5.3 confirm that reducing training data from the target dataset impacts more severely TSN than Cool-TSN. HMDB displays the largest gaps. We partially attribute this to the smaller size of HMDB and also because some categories of PHAV overlap with some categories of HMDB. Our results show that it is possible to replace half of HMDB with procedural videos and

Table 5.2 – Performance comparison for three target datasets. We show results for the original TSN, our reproduced results, and our two proposed methods for leveraging the extra training data from PHAV.

Target	Model	Spatial (RGB)	Temporal (Flow)	Full (RGB+Flow)
PHAV	TSN	65.9	81.5	82.3
UCF-101	[208]	85.1	89.7	94.0
UCF-101	TSN	84.2	89.3	93.6
UCF-101	TSN-FT	86.1	89.7	94.1
UCF-101	Cool-TSN	86.3	89.9	**94.2**
HMDB-51	[208]	51.0	64.2	68.5
HMDB-51	TSN	50.4	61.2	66.6
HMDB-51	TSN-FT	51.0	63.0	68.9
HMDB-51	Cool-TSN	53.0	63.9	**69.5**

Average mean accuracy (mAcc) across all dataset splits. Note: Wang *et al.* [208] uses TSN with cross-modality training.

Table 5.3 – TSN and Cool-TSN with different fractions of real-world training data.

Fraction of real -world samples	UCF101 (TSN)	UCF101+PHAV (Cool-TSN)	HMDB51 (TSN)	HMDB51+PHAV (Cool-TSN)
1%	25.9	**27.7**	8.1	**12.7**
5%	68.5	**71.5**	30.7	**37.3**
10%	80.9	**84.4**	44.2	**49.7**
25%	89.0	**90.4**	54.8	**60.7**
50%	92.5	**92.7**	62.9	**65.8**
100%	92.8	**93.3**	67.8	**70.1**

Mean Accuracy (mAcc) in split 1 of each respective real-world dataset.

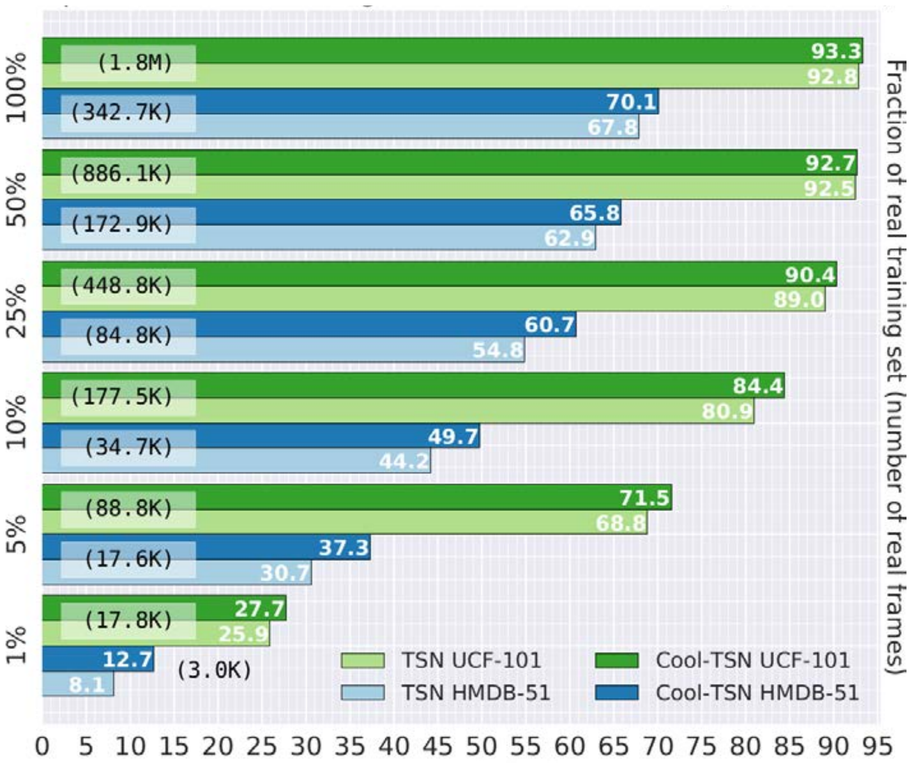


Figure 5.4 – Impact of using subsets of the real world training videos (split 1), with PHAV (Cool-TSN) or without (TSN). Mean Accuracy (mAcc) for RGB+Flow models.

still obtain comparable performance to using the full dataset (65.8 *vs.* 67.8). In a similar way, and although actions differ more, we show that reducing UCF-101 to a quarter of its original training set still yields a Cool-TSN model that rivals the state of the art [170, 207, 211]. This shows that our procedural generative model of videos can indeed be used to augment different small real-world training sets and obtain better recognition accuracy at a lower cost in terms of manual labor.

Figure 5.5 shows the performance of each network stream separately. The first and second plot in the figure show the RGB and optical flow streams respectively. One can see that the optical flow stream has a higher contribution than the RGB stream to the performance of our Cool-TSN, including when using very low fractions of the real data. This confirms that our generator is indeed producing plausible motions that help learn both the virtual and real-world data sources.

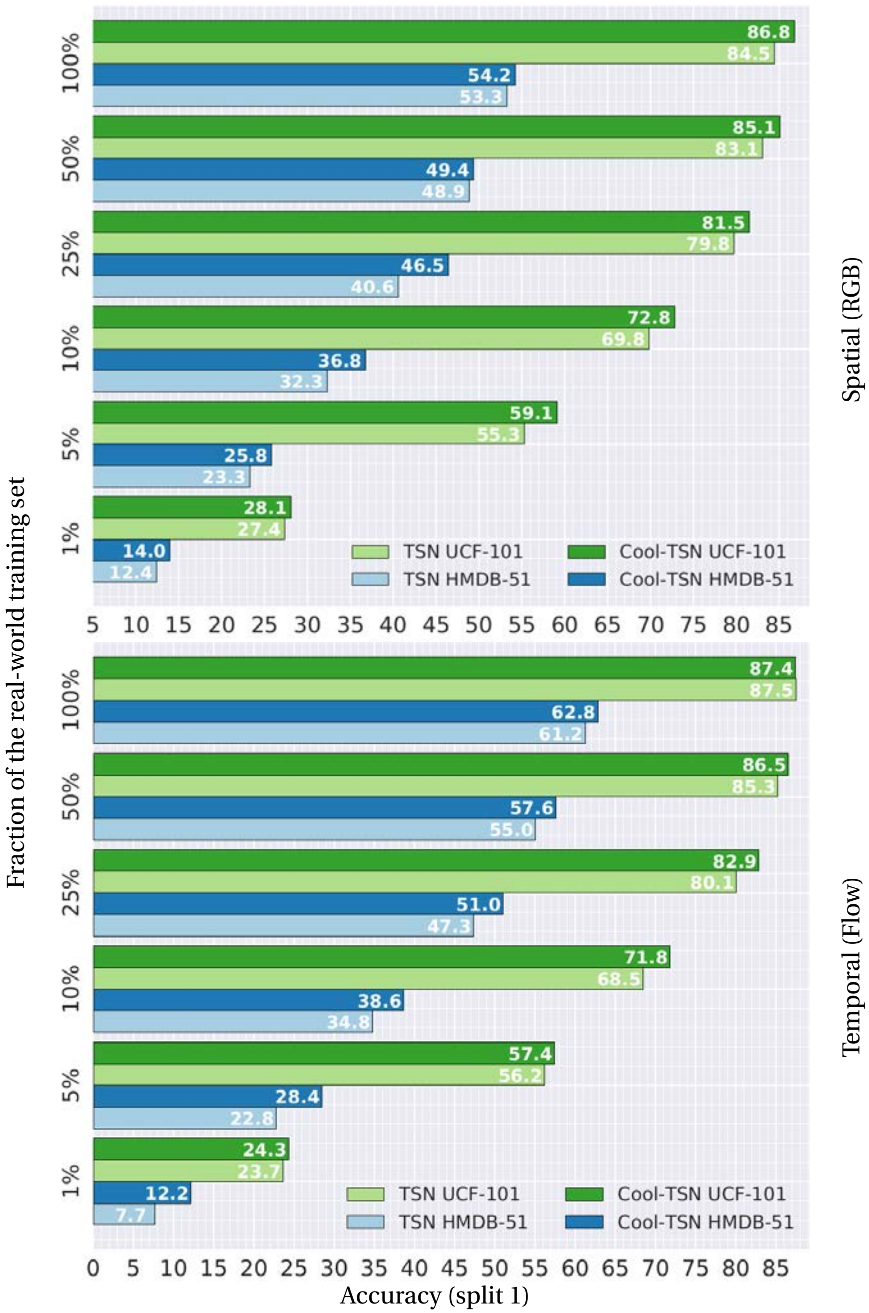
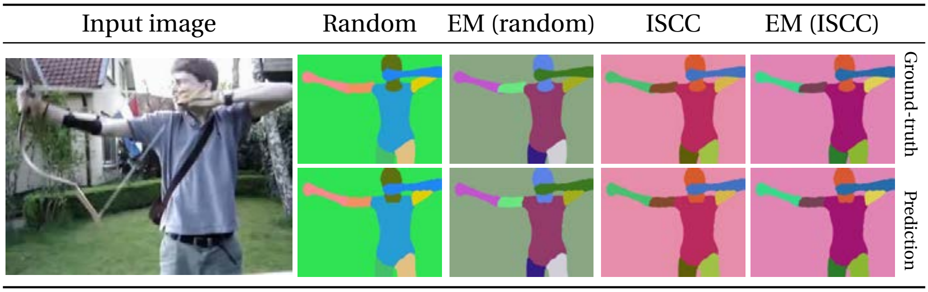


Figure 5.5 – TSN and Cool-TSN results for different amounts of real-world training data, for each separate stream, and for each dataset.

Table 5.4 – Color encoding strategies for initializing the human parsing stream.



Example frame from HMDB-51 with ground-truth puppet annotations from J-HMDB and human body parts as predicted by our human parsing layers. We recall that J-HMDB is a subset of the HMDB-51 dataset with additional annotations, including human body parts which have been annotated with the help of a puppet model manually positioned on top of the human performing the action (*cf.* Section 2.1.1).

5.4.3 Human Parsing Temporal Segment Networks

Wang *et al.* [208] has shown how using better initialization procedures for additional streams can have a significant impact in performance. We start our experiments with the human parsing stream by pre-training our fully convolutional layers in the PASCAL-Part [32] and PHAV datasets. We then evaluate the different techniques for initializing our pixel-wise semantic color transformation layer as discussed in Section 5.3.2 (*cf.* Table 5.4) on the first split of HMDB-51. We do not use soft-max probabilities to achieve color blending in those experiments, to focus instead on the performance impact of the different encodings. We also do not use end-to-end training, but learn our architecture in stages: first we learn the fully convolutional layers, we run the color transform, then learn the classification layers. In those settings, the configuration of our architecture used in those experiments corresponds to a plain application of a fully convolutional network, a colorization of the input frames according to the semantic labels predicted for their pixels using a particular encoding, followed by a spatial-stream (RGB) TSN. We do not include PHAV when training the action classification layers.

When training our human parsing layers, we use mostly the same parameters as the improved model in [32]: We use an initial learning rate of 10^{-3} (10^{-2} for the last pixel classification layer), a "poly" learning rate policy (the learning rate is multiplied by $(1 - \frac{iter}{max_i ter})^{power}$ at every iteration, where we use $power = 0.9$), a batch size of 10, random scaling, and initialize our weights from models pre-trained on MS-COCO [109]. We also train our models for a total of 40,000 iterations, and perform random horizontal flipping. When training our action classification layers, we use the same settings as in Section 5.4.1, but with a batch size of 64 videos.

Table 5.5 – Classification accuracy for different color encodings.

Color encoding	Training dataset for human parsing layers	Training dataset for action classif. layers	HMDB-51 %mAcc
Random	PASCAL-Part	HMDB-51	39.7
	PHAV	HMDB-51	40.8
	PASCAL-Part + PHAV	HMDB-51	45.2
EM (random)	PASCAL-Part	HMDB-51	40.2
	PHAV	HMDB-51	40.5
	PASCAL-Part + PHAV	HMDB-51	45.5
ISCC	PASCAL-Part	HMDB-51	40.3
	PHAV	HMDB-51	41.6
	PASCAL-Part + PHAV	HMDB-51	44.9
EM (ISCC)	PASCAL-Part	HMDB-51	41.4
	PHAV	HMDB-51	40.9
	PASCAL-Part + PHAV	HMDB-51	**46.3**

Mean Accuracy (mAcc) on the first split of HMDB-51 after colorization using the respective encoding. Models which have been trained on a combination of PASCAL-Parts and PHAV, while using an unsupervised refinement of the ISCC–NBS color designation system, give the best results.

We show our results in Table 5.5. As can be seen, training our human parsing layers exclusively on either PHAV or PASCAL-Parts results in comparable performance, and training on the combination of those two datasets gives consistently better results for all color encodings we consider. This gives us further evidence for the usefulness of our dataset: just as was the case for the Cool-TSN, our procedural videos can be used to replace real-world samples and still be used to obtain models of comparable performance, as well as augment real-world training sets to obtain better recognition accuracy with less manual effort.

Additionally, these experiments show that the synthetic semantic segmentation ground-truth contained in our PHAV dataset (and easily collected as part of the synthetic data generation process) can be used to improve action classification models based on semantic body part labels. Moreover, the colorization scheme based on optimizing the color centroids from the ISCC standard using EM outperforms the alternatives we considered. We therefore use this color encoding in all our further experiments.

5.4.4 Cool Human Parsing Temporal Segment Networks

We train our Cool-HPTSN models using different learning rates for each layer, basing our choices on the existing literature. Following [31], we use a learning rate of $lr_f = 10^{-3}$ for the first human parsing layers, $lr_p = 10^{-2}$ for the last (prediction) human parsing layer. Following [208] we use $lr_a = 10^{-3}$ for the action classification layers. We then use $w' = 0.1$ (*cf.* Equation 5.7) to give more emphasis to the action classification task, *i.e.* weighting the action classification task 10 times higher than the human parsing task.

We start our experiments by comparing our architecture to existing models in the literature, using the single-stream pose model of [230] as a baseline (*cf.* Table 5.6, first row). We reproduce the best results from Table 5.5 (*cf.* Table 5.6, second row), which corresponds to a hard segmentation version (*i.e.* without soft-labels) of our full architecture. Our results are already higher than [230], which we attribute to the better segmentation and classification models upon which our architecture is based.

Next, we enable the soft-labels, but freeze the fully convolutional and color transformation layers. This corresponds to a soft segmentation version of our full architecture. Using soft-labels, our model presents a small decrease in performance in HMDB-51 but significantly better performance in UCF-101 (*cf.* Table 5.6, third row). This demonstrates that, besides serving to keep our model fully differentiable, the use of a soft-label decision layer can have a positive impact in the performance of classification models built on top of segmentation-inspired representations.

Then, we unfreeze all layers in our architecture, and train our model end-to-end (*cf.* Table 5.6, fourth row). This approach results in superior performance for both HMDB-51 and UCF-101, surpassing all previous results and baselines. This demonstrates that our proposed architecture can effectively learn common representations for synthetic and real-world samples even though semantic segmentation ground-truth is only available for the synthetic tasks.

However, we note that our choices for model hyper-parameters had been mostly based on existing models in the literature, which may not have been optimal given our architecture. On the other hand, conducting a full hyper-parameter search, including the exploration of different learning rates for our multiple layers results in a combinatorial explosion in terms of possibilities to be explored. For this reason, instead of attempting a full hyper-parameter search, we conduct a small *cæteris paribus* analysis to measure the importance of fine-tuning middle layers in our network. We conduct this analysis by holding the weights of certain layers in our architecture fixed, while at the same time still letting gradients propagate across all layers. We report our results in the two last lines of Table 5.6.

Our results show that fixing the color transformation or mean subtraction layers

Table 5.6 – Cool-HPTSN results for HMDB-51 and UCF-101.

	Dataset (%mAcc)	
Method	HMDB-51	UCF-101
Pose [230]	36.0	56.9
Hard segmentation	46.3	59.1
Soft segmentation	45.3	64.0
End-to-end	48.0	67.5
Fixed mean subtraction	51.0	70.9
Fixed color transform	**51.2**	**72.5**

Mean Accuracy (mAcc) for split 1 of each dataset.

can have a large impact in performance, where fixing the color transformation to the matrix of weights learned through EM (ISCC) leads to the best results. We hypothesize the that learning of this layer could be improved by adding a color clustering objective into our objective function (*cf.* Equations 5.7 and 5.8), *i.e.* by maximizing inter-class variance while decreasing intra-class variance between semantic segmentation pixel classes directly within our model, which we leave as future work. Nonetheless, we show that propagating gradients between classification and segmentation layers results in better classification accuracy than independently-trained models (as in [230]) in all cases.

Next, we measure the impact of combining our Cool-HPTSN with TSNs for RGB and optical flow, and evaluate the performance of our full three-stream architecture. We display our results in Table 5.7. Our results show that our model can bring improvements when combined with either stream, with again HMDB-51 displaying the largest gaps. In the case of UCF-101, our model brings noticeable improvements when combined with the RGB stream, but only comparable effects when combined with the flow or both RGB+Flow.

Moreover, we observe that the SP stream is more complementary to RGB than optical flow. The gap in performance between RGB and SP+RGB is of 5.9 for HMDB-51 and 2.2 for UCF-101, whereas the gap between Flow and SP+Flow is of 5.6 and 0.1 for the same datasets, in respective order. We believe that the redundancy of the flow and parsing streams is at least partially due to the fact that they are both invariant to appearance, which has been shown to be one of the most important features captured by optical flow in action recognition algorithms [167]. We show examples comparing semantic segmentation and optical flow frames in Table 5.9. In

Table 5.7 – Cool-HPTSN results when merging with RGB and Flow streams.

		Dataset (%mAcc)	
Architecture	Stream	HMDB-51	UCF-101
TSN	RGB	53.3	84.5
TSN	Flow	61.2	87.5
Cool-HPTSN	SP	51.6	72.5
Cool-HPTSN	SP+RGB	59.2	86.7
Cool-HPTSN	SP+Flow	66.8	87.4
TSN	RGB+Flow	67.8	**92.8**
Cool-HPTSN	RGB+Flow+SP	**70.1**	92.1

Mean Accuracy (mAcc) for split 1 of each dataset.

addition to contrasting these different modalities, the examples shown in Table 5.9 also depict certain success and failure cases for our segmentation network, demonstrating it works even with grayscale images (8th row), but fails when subjects are at a distance and unconventional poses (9th row).

5.4.5 Comparison to the state of the art

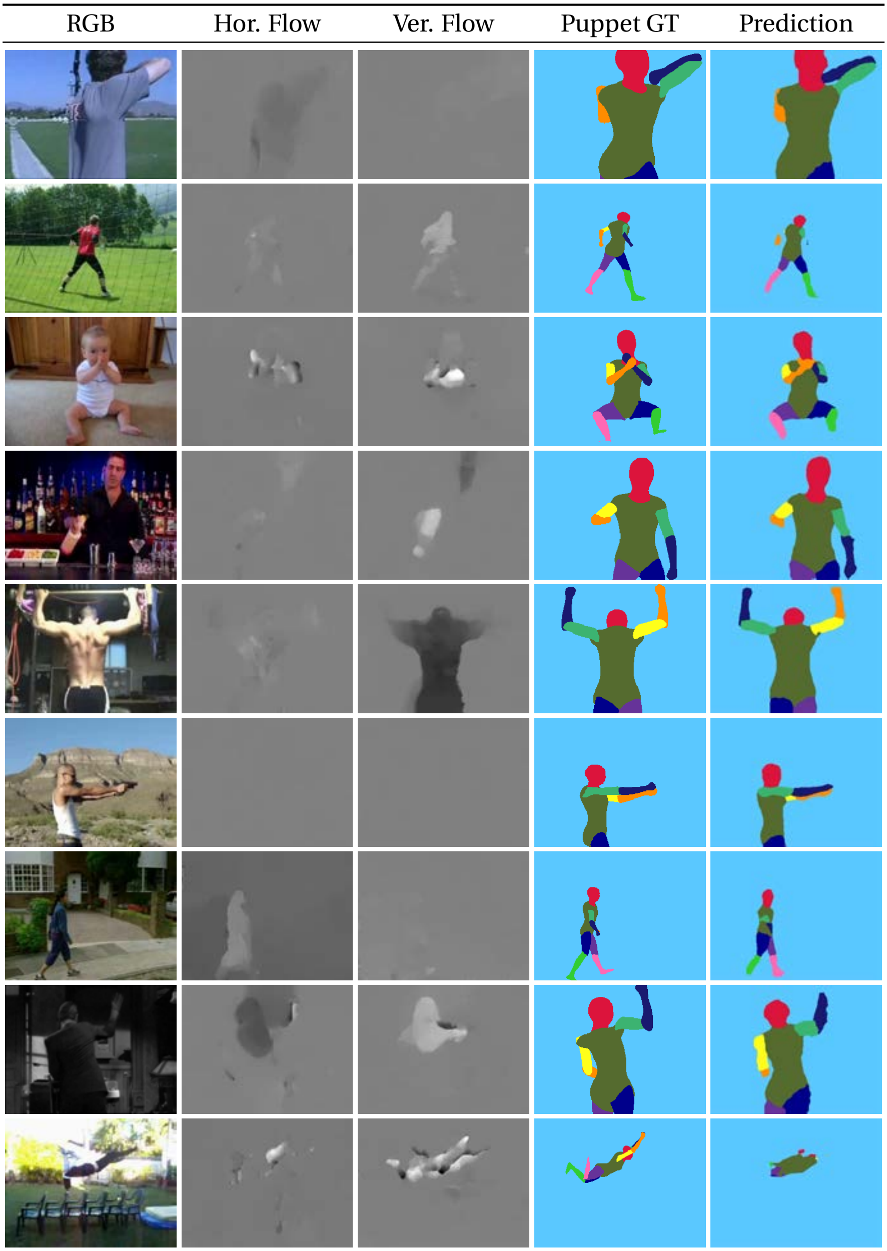
In this section, we compare our model with the state of the art in action recognition (Table 5.8). We separate the current state of the art into works that use one or multiple sources of training data (such as by pre-training, multi-task learning or model transfer). We note that all works that use multiple sources can potentially benefit from PHAV without any modifications. Our results indicate that our methods are competitive with the state of the art, including methods [91, 187] that use much more manually labeled training data like the Sports-1M dataset [91]. Our Cool-TSN approach also leads to better performance than the current best generative video model VGAN [198] on UCF101, for the same amount of manually labeled target real-world videos. We note that while VGAN's more general task is quite challenging and different from ours, [198] has also explored VGAN as a way to learn unsupervised representations useful for action recognition, thus validating our comparison.

Table 5.8 – Comparison against the state of the art* in action recognition.

	Method	UCF-101 %mAcc	HMDB-51 %mAcc
One source	iDT+FV [204]	84.8	57.2
	iDT+StackFV [141]	-	66.8
	iDT+SFV+STP [203]	86.0	60.1
	iDT+MIFS [101]	89.1	65.1
	VideoDarwin [51]	-	63.7
Multiple sources	2S-CNN [170]	88.0	59.4
	TDD [207]	90.3	63.2
	TDD+iDT [207]	91.5	65.9
	C3D+iDT [187]	90.4	-
	Actions~Trans [211]	92.0	62.0
	2S-Fusion [49]	93.5	69.2
	Hybrid-iDT [43]	92.5	70.4
	TSN-3M [208]	94.2	69.4
	VGAN [198]	52.1	-
	Cool-TSN	94.2	69.5
	Cool-HPTSN	92.4	68.8

*State of the art at the time of publication of work contained in this chapter. Average Mean Accuracy (mAcc) across all dataset splits.

Table 5.9 – Example frames from HMDB-51 with puppet annotations from J-HMDB.



We recall that J-HMDB is a subset of the HMDB-51 dataset with additional annotations, including human body parts which have been annotated with the help of a puppet model manually positioned on top of the human performing the action (*cf.* Section 2.1.1).

5.5 Summary of the chapter

In this chapter, we have proposed two deep learning architectures for action recognition and used them to validate the usefulness of our PHAV dataset proposed in Chapter 4. Our models can leverage multiple data sources during learning, mixing real-world and synthetic videos, while at the same time being able to leverage data modalities that are exclusive to the virtual domain. We provide quantitative evidence that the procedurally generated videos in PHAV can be used as a complement to small training sets of manually labeled real-world videos, and that our ground-truth annotations are accurate enough to be used as different modalities when learning complementary tasks for action recognition. Importantly, we have shown that we do not need to generate training videos for particular target categories fixed a priori. Instead, we show that data containing surrogate categories that have been defined procedurally improve representation learning for potentially unrelated target actions that might have only few real-world training examples.

6 Conclusions and future work

> The only time you don't fail is the last time you try anything – and it works.
>
> William Strong

In the past chapters we have presented our investigations and experiments on methods based on handcrafted features, and methods based on the end-to-end learning of deep hierarchical models directly from raw data. In this chapter we summarize the work conducted in this thesis and present our conclusions, giving directions for future work and listing the main outcomes of this research.

6.1 Conclusions

IN THIS DISSERTATION, we have explored the problem of human action recognition in videos with a specific focus on data-efficient models. We initially focused on approaches using handcrafted features, given their persistent success in action recognition at the commencement of this investigation. At that time, deep learning had already become the state-of-the-start approach for image recognition, methods that used handcrafted features still outperformed deep methods for video understanding.

In Chapter 3, we identified that both handcrafted and deep learning approaches had complementary strengths and weaknesses, and proposed to *combine the best* *of both worlds via a single hybrid classification architecture* consisting in applying sequentially the iDT handcrafted features, the unsupervised FV representation, unsupervised or supervised dimensionality reduction, and supervised deep layers. We performed an extensive experimental analysis on a variety of datasets, showing that our hybrid architecture yields data efficient, transferable models of small size that still outperform much more complex deep architectures trained end-to-end on millions of images and videos. We have also shown that our hybrid models can differentiate between some fine-grained action groups, confirming that the unsupervised video-level FV representation contains fine-grained information about the input video, information that may be more successfully exploited by our more complex (deeper) hybrid models. Most results presented in this chapter have been published in [43].

The success of handcrafted approaches in contrast to deep learning may be attributed in part to the increased amount of prior knowledge incorporated in the former. By definition, handcrafted features are created based on human expertise and therefore aggregate a number of previous human-drawn conclusions and interpretations about the data being handled [12]. The feature engineering process that gives origin to those features can therefore be seen as a technique for embedding prior knowledge on machine learning models directly. Nonetheless, supervised training data for video has typically been scarce, including at the outset of this research. For this reason, we decided to explore an alternative approach for achieving more data-efficient models: data synthesis. Instead of incorporating prior knowledge within a machine learning model, we investigated how to incorporate human knowledge in the *training data used to feed those models.*

In Chapter 4, we have shown that it is possible to create an interpretable parametric generative model of human action videos leveraging the strong *a priori* knowledge about the real world that is stored in the physic engines of modern video-game development systems [121]. Using this model, we generated a diverse, realistic, and physically plausible dataset of human action videos called PHAV, for "Procedural Human Action Videos". It contained a total of 39,982 videos, with more than 1,000 examples for each action of 35 categories and 8 data modalities. Even though this dataset used MOCAP sequences as a base, we have shown that our approach was not limited to a pre-defined set of animations: we presented techniques to select, divide and combine MOCAP sequences, as well as modify them at generation time through limited-time physics simulations. Going further, we have shown that it was possible to programatically define 14 new synthetic action categories with the aid of ragdoll physics and puppet manipulation. This dataset has been made publicly available for download in [42].

In Chapter 5, we provided quantitative evidence that our procedurally generated videos can be used to create better and more data-efficient models for action recognition. To this end, we had introduced *two novel deep multi-task representation learning architectures* that are able to leverage synthetic data to learn more about the real world, mixing synthetic and real videos, even if the action categories do not entirely align. We had also shown how to integrate different data modalities and ground-truths during learning, even when they are only available for one of the tasks. We have used our models to provide quantitative evidence that the procedurally generated videos in PHAV can be used as a complement to small training sets of manually labeled real-world videos, and that our ground-truth annotations are accurate enough to improve the learning of different data modalities for action recognition. Importantly, we proved that it was not necessary to generate training videos for particular target categories fixed a priori in order to achieve better models for action recognition. Instead, we have shown that through a multi-task framework it is possible to learn better representations for target action classes that might have only few real-world training examples, even when there only partial overlap between synthetic and real-world classes, including when some of these classes have been completely procedurally defined.

Our experiments on the UCF-101 and HMDB-51 benchmarks suggest that combining our large set of synthetic videos with small real-world datasets can boost recognition performance. We show that our parametric generative model for action videos can generate videos that when combined with real-world datasets result in significantly better performance than alternative approaches based on unsupervised generative models such as VGAN [198]. Although our model does not learn video representations like VGAN, we have shown how it can generate many diverse training videos thanks to its grounding in strong prior physical knowledge about scenes, objects, lighting, motions, and humans. Most results from this chapter have been published in [43].

6.2 Comparison to the state of the art

In the various chapters of this dissertation, we have compared the performance of our proposed models against the state-of-the-art performance at the time the contents of each of those chapters had been published. In this section, we present a comparison of these models against the state of the art at the time this dissertation has been completed. Table 6.1 presents the main works we compare against organized by category (cf. Table 2.4) and in chronological order within each block. The models we have proposed in this dissertation are shown at the bottom.

Table 6.1 – Comparison against the state of the art in action recognition.

		UCF-101	HMDB-51	Hollywood2	High-Five	Olympics	ActivityNet
	Method	%mAcc	%mAcc	%mAP	%mAP+	%mAP	%mAP
HANDCRAFTED	iDT+FV [204]	84.8 [205]	57.2	64.3	-	91.1	-
	SDT-ATEP [56]	-	41.3	54.4	62.4	85.5	-
	iDT+FM [140]	87.9	61.1	-	-	-	-
	RCS [75]	-	-	73.6	71.1	-	-
	iDT+SFV+STP [203]	86.0	60.1	66.8	69.4	90.4	-
	iDT+MIFS [101]	89.1	65.1	68.0	-	91.4	-
	VideoDarwin [51]	-	61.6	69.6	-	-	-
	VideoDarwin+HF+iDT [51]	-	63.7	73.7	-	-	-
	iDT+TBC [200]	-	-	63.8	**78.1**	-	-
DEEP-BASED	2S-CNN [170]$^{\text{IN}}$	88.0	59.4	-	-	-	-
	2S-CNN+Pool [127]$^{\text{IN}}$	88.2	-	-	-	-	-
	2S-CNN+LSTM [127]$^{\text{IN}}$	88.6	-	-	-	-	-
	Objects+Motion(R*) [81]$^{\text{IN}}$	88.5	61.4	66.4	-	-	-
	Comp-LSTM [176]$^{\text{ID}}$	84.3	44.0	-	-	-	-
	C3D+SVM [187]$^{\text{S1M,ID}}$	85.2	-	-	-	-	-
	FSTCN [182]$^{\text{IN}}$	88.1	59.1	-	-	-	-
	Actions~Trans [211]	92.0	62.0	-	-	-	-
	2S-Fusion [49]	93.5	69.2	-	-	-	-
	3-TSN [208]$^{\text{IN}}$	94.0	68.5	-	-	-	86.9
	9-TSN [209]$^{\text{IN}}$	94.9	71.0	-	-	-	**87.9**
	DHRP [50]	91.4	66.9	**76.7**	-	-	-
	I3D [22]$^{\text{K}}$	**97.9**	**80.2**	-	-	-	-
	CMSN (C3D) [230]	91.1	69.7	-	-	-	-
	CMSN (TSN) [230]	94.1	-	-	-	-	-
HYBRID	iDT+StackFV [141]	-	66.8	-	-	-	-
	TDD [207]$^{\text{IN}}$	90.3	63.2	-	-	-	-
	TDD+iDT [207]$^{\text{IN}}$	91.5	65.9	-	-	-	-
	CNN-hid6 [221]$^{\text{S1M}}$	79.3	-	-	-	-	-
	CNN-hid6+iDT [221]$^{\text{S1M}}$	89.6	-	-	-	-	-
	C3D+iDT+SVM [187]$^{\text{S1M,ID}}$	90.4	-	-	-	-	-
	Best from state of the art	97.9 [22]	80.2 [22]	76.7 [50]	78.1 [200]	91.4 [101]	87.9 [208]
	Our best FV+SVM	90.6	67.8	69.1	71.0	92.8	66.5
	Our best hybrid	92.5	70.4	72.6	76.7	**96.7**	72.5
	Our best Cool-TSN	94.2	69.5	-	-	-	-
	Our best Cool-HPTSN	92.4	68.8	-	-	-	-

Methods are organized by category (*cf.* Table 2.4) and sorted in chronological order in each block. IN: uses ImageNet. S1M: uses Sports-1M. K: uses Kinetics. ID: uses private internal data ([187] pre-trains models on an internal dataset referred to as I380K, whereas [176] uses additional 300h of unrelated YouTube videos).

6.3 Future work

We believe our results open interesting new perspectives to design even more data-efficient models, both hybrid and with deep learning approaches. A line of further research with hybrid methods would be to incorporate explicit temporal dynamics in these models using Long Short Term Memory recurrent networks (LSTMs) [176] and related methods [34, 35]. We also believe there is still much that could be gained by cross-dataset training *i.e.* pre-training on more datasets, or using a multi-task loss as we have explored in Chapter 5.

However, we also believe that more promising lines of research may reside within deep learning approaches. For instance, there is still much to be explored in terms of hyper-parameter values in our proposed architectures. Furthermore, our approaches could also benefit from incorporating ideas from more recent models such as Carreira *et al.* 's Inflated 3D Convolutional Networks[22], or the Markov-chain approach for modality fusion in [230].

Regarding our approach for procedural action video generation, one interesting direction for further research would be to explore the other data modalities we generate (*cf.* Section 4.4.2). For instance, we hypothesize it should be possible to leverage multiple data modalities from different data sources under the recent Learning Using Privileged Information (LUPI) paradigm of [191], as well as the Generalized Distillation (GD) of [111]. Another interesting direction would be the combination with unsupervised generative models like VGAN [198] for dynamic background generation, domain adaptation, or real-to-virtual world style transfer [60, 227].

The generation of synthetic data also gives access to a extra information about the generated samples and the generation process itself. Our PHAV dataset includes sparse vectors (*cf.* end of Section 4.4.2) capturing detailed information about the status of our procedural generator at the start of each video, which can be regarded as their *procedural recipes*. One promising line of work would be the exploitation of the multiple parametric variations in our synthetic data, possibly by proposing methods to drive data generation, *e.g.,* by adapting the Image Transformation Pursuit (ITP) method of [139], the Adaptive Weighted (AW-)SGD of [15], or methods from curriculum learning [13] for this purpose.

Another line of work in this same direction would be to explore methods from deep reinforcement learning [161, 189], and active learning [225] to drive the generation. We hypothesize that it should be possible to devise an active learning method where an agent could sample the entire space of possible synthetic videos that our generator can produce, with each observation corresponding to a video procedural parameter that could be lazily-evaluated to generate a training video on-the-fly.

6.4 Patents

The execution of this work originated the following patents:

- César Roberto de Souza, Adrien Gaidon, Eleonora Vig, and Antonio Manuel López Peña. **System and method for video classification using a hybrid unsupervised and supervised multi-layer architecture**. U.S. Patent Application 15/240,561, filed February 22, 2018.

6.5 Scientific articles

The work presented in this dissertation has been published in the following venues:

6.5.1 International conferences

- César Roberto de Souza, Adrien Gaidon, Eleonora Vig, and Antonio Manuel López Peña. **Sympathy for the Details: Dense Trajectories and Hybrid Classification Architectures for Action Recognition**. The 14th European Conference on Computer Vision (ECCV), Amsterdam, Netherlands, 2016.
- César Roberto de Souza, Adrien Gaidon, Yohann Cabon, and Antonio Manuel López Peña. **Procedural Generation of Videos to Train Deep Action Recognition Networks**. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, USA, 2017.

6.5.2 Workshops and events

- César Roberto de Souza, Adrien Gaidon, Yohann Cabon, and Antonio Manuel López Peña. **Procedural Generation of Videos to Train Deep Action Recognition Networks**. CVPR Vision Meets Cognition Workshop, 2017.

6.6 Contributed datasets

The work described in this thesis has lead to the contribution of the following dataset to the public:

- **Procedural Human Action Videos (PHAV) dataset**: A dataset of synthetic procedurally generated human action recognition videos. http://adas.cvc.uab.es/phav/

6.7 Challenges

The techniques described in this dissertation have been used in the following challenges:

- César Roberto de Souza, Adrien Gaidon, Eleonora Vig, and Antonio Manuel López Peña. **The Xerox Research Centre Europe submission to the ActivityNet Large Scale Activity Recognition Challenge 2016**. CVPR ActivityNet Large Scale Activity Recognition Challenge Workshop, 2016. Ranked 8$^{\text{th}}$ according to Top-1 and 9$^{\text{th}}$ according to mAP for the untrimmed video classification task in the challenge. We give more details about our submission and extended results for our approach in Appendix A.

6.8 Scientific dissemination

We have disseminated this scientific knowledge acquired during the execution of this work in the form of demonstrations and talks in multiple events and venues:

6.8.1 Demonstrations

- César Roberto de Souza, Adrien Gaidon, and Antonio Manuel López Peña. **Realistic Virtual Worlds and Human Actions for Video Understanding**, Neural Information Processing Systems (NIPS), Barcelona, Spain, 2016.

- César Roberto de Souza, Adrien Gaidon, and Antonio Manuel López Peña. **Procedural Human Action Videos (PHAV)**. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, USA, 2017

6.8.2 Talks

- César Roberto de Souza. **State-of-the-art action recognition with dense trajectories and modern bag of words**. Xerox Research Centre Europe (XRCE) Computer Vision Seminars, Meylan, France, 2016.

A ActivityNet Challenge 2016

In this appendix, we describe our method for untrimmed action recognition whose results have been submitted to the ActivityNet Challenge 2016. Our method is based on data-augmentation and feature fusion techniques for video-level Dense Trajectories and C3D features, audio-level MFCC features, and frame-level ImageNet features.

A.1 Challenge

The ActivityNet Large Scale Activity Recognition Challenge was first organized as a half-day workshop held during CVPR 2016, with the goal of advancing and simulating research in computer vision and, more specifically, human action understanding. This challenge was focused in the recently published at the time ActivityNet benchmark.

The ActivityNet [73] dataset (release 1.3) contains 19,994 untrimmed videos distributed over 200 activity classes. The dataset has been divided into three sets, two with publicly available ground-truth labels (training and validation) and one where the ground-truth labels have been withheld for the purpose of the challenge and are not publicly available (testing). Since this dataset is distributed using Youtube, not all videos originally included in the dataset remain available to download. At the time we downloaded the dataset, it contained 9902 training videos, 4856 validation videos, and 4988 test videos (*cf.* Section 2.1.1 and Tables 2.1 and 2.2 in the main chapters of this dissertation for more details).

A.2 Submission

Our method is based on the improved Dense Trajectories (iDT) pipeline of Wang & Schmid [204]. However, we employ modifications to both the beginning (data preprocessing) and end (feature fusion) of this pipeline. First, we preprocess the videos from the 1.3 version of the ActivityNet to reduce their size, downscaling them to a height of 244 pixels while keeping the aspect ratio. Then, we augment the training data using DAFS, *i.e.* using frame-skipping and horizontal mirroring. Afterwards, we proceed to extract information from both their audio and video streams.

From the audio stream, we extract a set of 40-dimensional MFCC audio features for each video. From the video stream, we extract Trajectory shape (Traj) [201], HOG [38], HOF [39], horizontal and vertical MBH components [201] descriptors along trajectories obtained by median filtering dense optical flow, using the same parameters given in [204]. We subsample the trajectories from each video, keeping only 10% of the originally extracted trajectory descriptors. We apply the RootSIFT normalization [3] (ℓ_1 normalization followed by square-rooting) to all video descriptors.

Next, we randomly sample 256,000 trajectories and MFCC vectors from the pool of training videos to learn the vocabularies needed for feature encoding. Before learning the GMMs, we apply PCA to the descriptors, reducing their dimensionality by a factor of 2. Afterwards, we concatenate the PCA-transformed video descriptors with their respective $(x, y, t) \in \mathbb{R}^3$ coordinates.

We learn one separate GMM per descriptor channel. After the vocabularies have been created, we use them to create Fisher Vector (FV) [147, 150] encodings for each local descriptor in each descriptor channel, combining these encodings into a per-channel, video-level representation using sum-pooling. We then apply power normalization [150] (signed-square-rooting followed by ℓ_2 normalization) to those per-channel FVs. Next, we concatenate all channels together and reapply this same normalization [101].

Finally, we learn separate probability-calibrated SVMs for a) iDT+MFCC Fisher Vectors b) video-level C3D features [187], and c) image-level Shuffle ImageNet (SIM) features [119], the last two of which were provided by the challenge[1]. We concatenate the probability outputs of each of those SVMs, use the concatenated vector as a global feature vector, learn a fourth SVM on top of this global feature vector, and use this final SVM to predict the final scores for each video (this approach is sometimes referred to as *Stacked Generalization* [212]).

A.3 Results

Using the method described above, we achieved the 8th position in the challenge according to top-1 classification (78.5%), and the 9th position according to mAP (82.6%) for the classification task as calculated by the computer servers of the challenge organizers. Ground-truth class labels are not publicly available for this set. In our experience, validation set accuracy was a strong predictor of testing set accuracy, with a typical gap of 3-4 percent points (p.p.) between a result obtained in the validation set and its subsequent measurement in the testing set.

A.4 Pipeline

Equation A.1 defines the pipeline for our submission to the 2016 ActivityNet Challenge using the notation introduced in Section 2.2.1. As can be seen, the pipeline is divided into four main paths: The path for dense trajectories features, the path for MFCC features, the path for C3D features, and the path for ImageNet Shuffle features. The first paths to be merged are the ones for dense trajectories and MFCC, with their merging happening at the representation-level. Then, this combined path merges with the C3D and SIM paths at the score-level using a third SVM which is learned on the fusion of scores from each path.

[1] http://activity-net.org/challenges/2016/

Table A.1 – Performance of different iDT pipelines for ActivityNet's validation set.

Pipeline	ActivityNet (validation set) %mAP
iDT+SFV+STP	57.56
iDT	60.89
iDT+STA	60.94
iDT+DN	62.27
iDT+STA+DN	62.36
iDT+MIFS	62.77
iDT+HF	63.26
iDT+STA+MIFS+DN	64.12
iDT+HF+DN	64.83
iDT+DAFS	64.96
iDT+STA+MIFS+DN	65.12
iDT+DAFS+DN	66.49
iDT+DAFS+DN+PCA	66.49
iDT+DAFS+DN+PCA+L2	66.92
iDT+DAFS+DN+PCA+L2+MFCC+C3D	72.76
iDT+DAFS+DN+PCA+L2+MFCC+C3D+SIM+W	**79.09**

Results are ordered by mAP as measured in ActivityNet's validation set. iDT: Improved Dense Trajectories; SFV: Spatial Fisher Vector; STP: Spatio-Temporal Pyramids; STA: Spatio-Temporal Augmentation; MIFS: Multi-skIp Feature Stacking; HF: Horizontal Flipping; DN: Double-Normalization; DAFS: Data Augmentation Feature Stacking; *without Trajectory descriptor; PCA: Principal Component Analysis after FV encoding; L2: ℓ_2 normalization after PCA; MFCC: Mel Frequency Cepstral Coefficients; C3D: Frame-based featurs from C3D model; SIM: ImageNetShuffle ConvNet features; W: Score-level fusion with SVM.

$$\left\{\begin{array}{c} SIM-26-29\\ C3D-10-14-20-26-29\\ \left\{\begin{array}{c} MFCC\\ \left\{\begin{array}{l}\left\{\begin{array}{c}0\\1\\2\end{array}\right\}\\ \left\{\begin{array}{c}0\\1\\2\end{array}\right\}\quad -4\end{array}\right\}-6-\left\{\begin{array}{c}\left\{\begin{array}{c}Traj-\mathbf{S}-8-9-10\\xyt\end{array}\right\}-\mathbf{F}\\ \left\{\begin{array}{c}HOG-\mathbf{S}-8-9-10\\xyt\end{array}\right\}-\mathbf{F}\\ \left\{\begin{array}{c}HOF-\mathbf{S}-8-9-10\\xyt\end{array}\right\}-\mathbf{F}\\ \left\{\begin{array}{c}MBHx-\mathbf{S}-8-9-10\\xyt\end{array}\right\}-\mathbf{F}\\ \left\{\begin{array}{c}MBHy-\mathbf{S}-8-9-10\\xyt\end{array}\right\}-\mathbf{F}\end{array}\right\}\end{array}\right\}-14-21-23-24-\mathbf{R}-23-24-26-29-10\end{array}\right\}-\mathbf{C}-26-29-31 \tag{A.1}$$

Bibliography

[1] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. YouTube-8M: a large-scale video classification benchmark. *Computing Research Repository (CoRR)*, arXiv:abs/1609.08675, September 2016.

[2] Robin Aly, Relja Arandjelovic, Ken Chatfield, Matthijs Douze, Basura Fernando, Zaid Harchaoui, Kevin Mcguiness, Noël O'Connor, Dan Oneata, Omkar Parkhi, Danila Potapov, Jerome Revaud, Cordelia Schmid, Jochen Schwenninger, David Scott, Tinne Tuytelaars, Jakob Verbeek, Heng Wang, and Andrew Zisserman. The AXES submissions at TrecVid 2013. *TREC Video Retrieval Evaluation Workshop*, November 2013.

[3] Relja Arandjelovic and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2911–2918. IEEE, June 2012.

[4] Relja Arandjelovic and Andrew Zisserman. All about VLAD. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1578–1585, June 2013.

[5] Maryam Asadi-Aghbolaghi, Albert Clapes, Marco Bellantonio, Hugo Jair Escalante, Víctor Ponce-López, Xavier Baró, Isabelle Guyon, Shohreh Kasaei, and Sergio Escalera. A survey on deep learning based approaches for action and gesture recognition in image sequences. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition (FG)*, pages 476–483. IEEE, May 2017.

[6] Joan Marc Llargues Asensio, Juan Peralta, Raul Arrabales, Manuel González Bedia, Paulo Cortez, and Antonio López. Artificial intelligence approaches for the generation and assessment of believable human-like behaviour in virtual characters. *Expert Systems With Applications*, 41(16):7281–7290, 2014.

[7] Mathieu Aubry, Daniel Maturana, Alexei Efros, Bryan Russell, and Josef Sivic. Seeing 3D chairs: exemplar part-based 2d-3d alignment using a large dataset

of CAD models. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3762–3769. IEEE, June 2014.

[8] Mathieu Aubry and Bryan C. Russell. Understanding deep features with computer-generated imagery. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2875–2883. IEEE, December 2015.

[9] Artem Babenko and Victor Lempitsky. Aggregating local deep features for image retrieval. In *Proceedings of the International Conference on Computer Vision (ICCV)*, volume 2015, pages 1269–1277. IEEE, December 2015.

[10] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Action classification in soccer videos with long short-term memory recurrent neural networks. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 154–159. Springer Berlin Heidelberg, 2010.

[11] Nicolas Ballas, Li Yao, Chris Pal, and Aaron C. Courville. Delving deeper into convolutional networks for learning video representations. In *Proceedings of the International Conference on Learning Representations*, March 2016.

[12] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 35(8):1798–1828, 2013.

[13] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the International Conference on Machine Learning*, pages 1–8, 2009.

[14] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006.

[15] Guillaume Bouchard, Théo Trouillon, Julien Perez, and Adrien Gaidon. Online learning to sample. *Computing Research Repository (CoRR)*, arXiv:abs/1506.09016, 2015.

[16] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the International Conference on Machine Learning*, pages 111–118. Omnipress, 2010.

[17] Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal on Machine Learning Research*, 2:499–526, March 2002.

[18] William Brendel and Sinisa Todorovic. Learning spatiotemporal graphs of human activities. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 778–785. IEEE, November 2011.

[19] Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(20):88–97, 2009.

[20] Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. A naturalistic open source movie for optical flow evaluation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 611–625. Springer-Verlag, 2012.

[21] Carnegie Mellon Graphics Lab. Carnegie Mellon University motion capture database, 2016.

[22] João Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the Kinetics dataset. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733. IEEE, July 2017.

[23] Matthew P. Carter. *Computer Graphics: Principles and practice*, volume 22. Wiley Subscription Services, Inc., A Wiley Company, February 1997.

[24] Paul Centore. sRGB centroids for the ISCC-NBS colour system. Munsell Colour Science for Painters, May 2016.

[25] Alexandros Andre Chaaraoui, José Ramón Padilla-López, Pau Climent-Pérez, and Francisco Flórez-Revuelta. Evolutionary joint selection to improve human action recognition with rgb-d devices. *Expert Systems With Applications*, 41(3):786–794, 2014.

[26] Jose M. Chaquet, Enrique J. Carmona, and Antonio Fernández-Caballero. A survey of video datasets for human action and activity recognition. *Computer Vision and Image Understanding*, 117(6):633–659, 2013.

[27] Ken Chatfield, Victor Lempitsky, Andrea Vedaldi, and Andrew Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *Proceedings of the British Machine Vision Conference (BMVC)*, volume 2, pages 76.1–76.12, November 2011.

[28] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2014.

[29] Chenyi Chen, Ari Seff, Alain L. Kornhauser, and Jianxiong Xiao. DeepDriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2722–2730, December 2015.

[30] Liang-Chieh Chen, Sanja Fidler, and Raquel Urtasun. Beat the MTurkers: Automatic image labeling from weak 3D supervision. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3198–3205. IEEE, June 2014.

[31] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 40(4):834–848, April 2018.

[32] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1979–1986. IEEE, June 2014.

[33] François Chollet. Keras: Deep learning library for Theano and TensorFlow. https://keras.io, 2015.

[34] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *Computing Research Repository (CoRR)*, arXiv:abs/1412.3555v1, December 2014.

[35] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *Proceedings of the International Conference on Machine Learning*, 2015.

[36] Ramazan Gokberk Cinbis, Jakob Verbeek, and Cordelia Schmid. Approximate Fisher Kernels of non-iid image models for image categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 38(6):1084–1098, June 2016.

[37] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele.

The Cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223. IEEE, 2016.

[38] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 886–893. IEEE, June 2005.

[39] Navneet Dalal, Bill Triggs, and Cordelia Schmid. Human detection using oriented histograms of flow and appearance. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 428–441. Springer-Verlag, 2006.

[40] César Roberto De Souza. The Accord.NET Framework. A framework for scientific computing. Available in: http://accord-framework.net, 2014.

[41] César Roberto De Souza, Adrien Gaidon, Yohann Cabon, and Antonio Manuel López. Procedural generation of videos to train deep action recognition networks. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2594–2604. IEEE, July 2017.

[42] César Roberto De Souza, Adrien Gaidon, Yohann Cabon, and Antonio Manuel Lopez Pena. Procedural human action videos - post-processed rgb frames. Torrent files for post-processed RGB frames. Links available at: http://adas.cvc.uab.es/phav/, 2017.

[43] César Roberto De Souza, Adrien Gaidon, Eleonora Vig, and Antonio Manuel López. Sympathy for the details: Dense trajectories and hybrid classification architectures for action recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 697–716. Springer International Publishing, 2016.

[44] Jonathan Delhumeau, Philippe–Henri Gosselin, Hervé Jégou, and Patrick Pérez. Revisiting the VLAD image representation. In *Proceedings of the ACM International Conference on Multimedia (ACM-MM)*, pages 653–656, Barcelona, Spain, 2013. ACM.

[45] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, volume 39, pages 677–691. IEEE, April 2017.

[46] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazırba, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks, December 2015.

[47] Arjan Egges, Arno Kamphuis, and Mark Overmars, editors. *Motion in Games*, volume 5277 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[48] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision (IJCV)*, 88(2):303–338, June 2010.

[49] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1933–1941. IEEE, June 2016.

[50] Basura Fernando, Peter Anderson, Marcus Hutter, and Stephen Gould. Discriminative hierarchical rank pooling for activity recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1924–1932. IEEE, June 2016.

[51] Basura Fernando, Efstratios Gavves, M. José Oramas, Amir Ghodrati, and Tinne Tuytelaars. Modeling video evolution for action recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5378–5387. IEEE, June 2015.

[52] Stephen Few. Multivariate analysis using parallel coordinates. *Perceptual Edge*, pages 1–9, September 2006.

[53] Adrien Gaidon, Zaid Harchaoui, and Cordelia Schmid. Actom sequence models for efficient action detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3201–3208. IEEE, June 2011.

[54] Adrien Gaidon, Zaid Harchaoui, and Cordelia Schmid. Recognizing activities with cluster-trees of tracklets. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 30–1, 2012.

[55] Adrien Gaidon, Zaid Harchaoui, and Cordelia Schmid. Temporal localization of actions with actoms. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 35(11):2782–2795, November 2013.

[56] Adrien Gaidon, Zaid Harchaoui, and Cordelia Schmid. Activity representation with motion hierarchies. *International Journal of Computer Vision (IJCV)*, 107(3):219–238, May 2014.

[57] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4340–4349, 2016.

[58] Quentin Galvane, Marc Christie, Christophe Lino, and Rémi Ronfard. Camera-on-rails: Automated computation of constrained camera paths. *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 151–157, 2015.

[59] Chuang Gan, Naiyan Wang, Yi Yang, Dit Yan Yeung, and Alexander G. Hauptmann. DevNet: A deep event network for multimedia event detection and evidence recounting. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2568–2577. IEEE, June 2015.

[60] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, June 2016.

[61] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks, May 2010.

[62] Yunchao Gong, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 392–407. Springer International Publishing, 2014.

[63] Philippe-Henri Gosselin, Naila Murray, Hervé Jégou, and Florent Perronnin. Revisiting the Fisher Vector for fine-grained classification. *Pattern Recognition Letters*, 49:92–98, 2014.

[64] Martin Guay, Rémi Ronfard, Michael Gleicher, and Marie-Paule Cani. Adding dynamics to sketch-based character animations. *Sketch-Based Interfaces and Modeling*, pages 27–34, 2015.

[65] Martin Guay, Rémi Ronfard, Michael Gleicher, and Marie-Paule Cani. Space-time sketching of character animation. *ACM Transactions on Graphics*, 34(4):118:1–118:10, July 2015.

[66] Ralf Haeusler and Daniel Kondermann. Synthesizing real world stereo challenges. In *Proceedings of the German Conference on Pattern Recognition*, pages 164–173. Springer Berlin Heidelberg, 2013.

[67] Vladimir Haltakov, Christian Unger, and Slobodan Ilic. Framework for generation of synthetic ground truth data for driver assistance applications. In *Proceedings of the German Conference on Pattern Recognition*, pages 323–332. Springer Berlin Heidelberg, 2013.

[68] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. SynthCam3D: Semantic understanding with synthetic indoor scenes. *Computing Research Repository (CoRR)*, arXiv:abs/1505.00171, 2015.

[69] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. Understanding real world indoor scenes with synthetic data. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4077–4085. IEEE, June 2016.

[70] John A Hartigan and Manchek A Wong. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[71] Tal Hassner. A critical review of action recognition benchmarks. In *Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops*, pages 245–250, June 2013.

[72] Hironori Hattori, Vishnu Naresh Boddeti, Kris M. Kitani, and Takeo Kanade. Learning scene-specific pedestrian detectors without real data. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3819–3827. IEEE, June 2015.

[73] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. ActivityNet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–970. IEEE, June 2015.

[74] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin Wilson. CNN architectures for large-scale audio classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 131–135, March 2017.

[75] Minh Hoai and Andrew Zisserman. Improving human action recognition using score distribution and ranking. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, pages 3–20. Springer International Publishing, 2015.

[76] Jian-Fang Hu, Wei-Shi Zheng, Jianhuang Lai, and Jianguo Zhang. Jointly learning heterogeneous features for RGB-D activity recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 39(11):2186–2200, November 2017.

[77] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1647–1655, July 2017.

[78] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456. JMLR.org, July 2015.

[79] Ferris Jabr. The reading brain in the digital age: The science of paper versus screens. *Scientific American*, 11:1–19, April 2013.

[80] Mihir Jain, Herve Jegou, and Patrick Bouthemy. Better exploiting motion for better action recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2555–2562. IEEE, June 2013.

[81] Mihir Jain, Jan Van Gemert, and Cees G. M. Snoek. What do 15,000 object categories tell us about classifying and localizing actions? In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 46–55. IEEE, June 2015.

[82] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. On the burstiness of visual elements. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1169–1176, June 2009.

[83] Hervé Jégou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Pérez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 34(9):1704–1716, September 2012.

[84] Hueihan Jhuang, Juergen Gall, Silvia Zuffi, Cordelia Schmid, and Michael J. Black. Towards understanding action recognition. *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 3192–3199, December 2013.

[85] Shuiwang Ji, Ming Yang, Kai Yu, and Wei Xu. 3D convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 35(1):221–231, Jan 2013.

[86] Chengcheng Jia and Yun Fu. Low-rank tensor subspace learning for rgb-d action recognition. *IEEE Transactions on Image Processing*, 25(10):4641–4652, October 2016.

[87] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. IEEE, June 2009.

[88] Y.-G. Jiang, J Liu, a Roshan Zamir, I Laptev, M Piccardi, M Shah, and R Sukthankar. THUMOS Challenge: Action recognition with a large number of classes, 2013.

[89] Biliana Kaneva, Antonio Torralba, and William T. Freeman. Evaluation of image features using a photorealistic virtual world. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2282–2289, November 2011.

[90] Vadim Kantorov and Ivan Laptev. Efficient feature extraction, encoding and classification for action recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2593–2600, June 2014.

[91] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732. IEEE, June 2014.

[92] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The Kinetics human action video dataset. *Computing Research Repository (CoRR)*, arXiv:abs/1705.06950, May 2017.

[93] Kenneth L. Kelly. The ISCC-NBS method of designating colors and a dictionary of color names. *National Bureau of Standards circular 533*, 1955.

[94] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Proceedings of the International Conference on Learning Representations*, May 2015.

[95] Kishore Konda, Xavier Bouthillier, Roland Memisevic, and Pascal Vincent. Dropout as data augmentation. In *Proceedings of the International Conference on Learning Representations*, June 2015.

[96] Josip Krapac, Jakob Verbeek, and Frédéric Jurie. Modeling spatial layout with Fisher vectors for image categorization. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1487–1494, November 2011.

[97] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[98] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[99] Hilde Kuehne, Huei-Han Jhuang, Estibaliz Garrote-Contreras, Tomaso Poggio, and Thomas Serre. HMDB: a large video database for human motion recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2556–2563, November 2011.

[100] Tian Lan, Yuke Zhu, Amir Roshan Zamir, and Silvio Savarese. Action recognition by hierarchical mid-level action elements. *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 4552–4560, December 2015.

[101] Zhenzhong Lan, Ming Lin, Xuanchong Li, Alexander G. Hauptmann, and Bhiksha Raj. Beyond gaussian pyramid: Multi-skip feature stacking for action recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 204–212. IEEE, June 2015.

[102] Michael S Langer and Heinrich H Bülthoff. Depth discrimination from shading under diffuse lighting. *Perception*, 29(6):649–660, June 2000.

[103] Ivan Laptev. On space-time interest points. *International Journal of Computer Vision (IJCV)*, 64(2–3):107–123, September 2005.

[104] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, June 2008.

[105] Svetlana Lazebnik, Cordelia Schmid, Jean Ponce, Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2169–2178. IEEE Computer Society, 2010.

[106] Quoc V. Le, Will Y. Zou, Serena Y. Yeung, and Andrew Y. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3361–3368. IEEE, June 2011.

[107] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Handwritten digit recognition with a back-propagation network. In *Proceedings of Neural Information Processing Systems (NIPS)*, volume 2, pages 396–404. MIT Press, 1989.

[108] Adam Lerer, Sam Gross, and Rob Fergus. Learning physical intuition of block towers by example. In *Proceedings of Machine Learning Research*, volume 48, pages 430–438, June 2016.

[109] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C.Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 740–755. Springer International Publishing, 2014.

[110] Jingen Liu, Benjamin Kuipers, and Silvio Savarese. Recognizing human actions by attributes. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3337–3344. IEEE, June 2011.

[111] David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. *Computing Research Repository (CoRR)*, arXiv:abs/1511.03643:1–10, 2015.

[112] Richard Maclin and David Opitz. An empirical evaluation of bagging and boosting. In *AAAI*, pages 546–551, 1997.

[113] Javier Marín, David Vázquez, David Gerónimo, and Antonio M. López. Learning appearance in virtual scenarios for pedestrian detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 137–144, June 2010.

[114] Marcin Marszalek, Ivan Laptev, and Cordelia Schmid. Actions in context. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2929–2936. IEEE, June 2009.

[115] Francisco Massa, Bryan C. Russell, and Mathieu Aubry. Deep exemplar 2D-3D detection by adapting from real to rendered views. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6024–6033. IEEE, June 2016.

[116] Pyry Matikainen, Rahul Sukthankar, and Martial Hebert. Feature seeding for action recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1716–1723, November 2011.

[117] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016.

[118] Stephan Meister and Daniel Kondermann. Real versus realistically rendered scenes for optical flow evaluation. In *CEMT*, pages 1–6, March 2011.

[119] Pascal Mettes, Dennis C Koelma, and Cees G M Snoek. The ImageNet shuffle: Reorganized pre-training for video event detection. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, pages 175–182. ACM, 2016.

[120] Gavin Miller. Efficient algorithms for local and global accessibility shading. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 319–326, New York, New York, USA, 1994. ACM Press.

[121] Ian Millington. *Game Physics Engine Development, Second Edition: How to Build a Robust Commercial-Grade Physics Engine for Your Game*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2010.

[122] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with deep

reinforcement learning. In *Proceedings of Neural Information Processing Systems Workshops*, 2013.

[123] Steven Molnar. Efficient supersampling antialiasing for high-performance architectures. Technical Report TR91-023, North Carolina University at Chapel Hill, The address of the publisher, 1991.

[124] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the International Conference on Machine Learning*, pages 807–814. Omnipress, 2010.

[125] Sanath Narayan and Kalpathi R. Ramakrishnan. Hyper-fisher vectors for action recognition. *Computing Research Repository (CoRR)*, arXiv:abs/1509.08439, 2015.

[126] Joe Yue–Hei Ng, Fan Yang, and Larry S. Davis. Exploiting local features from deep networks for image retrieval. In *Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops*, pages 53–61. IEEE, June 2015.

[127] Joe Yue-Hei Ng, Matthew J. Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4694–4702. IEEE, June 2015.

[128] Siqi Nie, Ziheng Wang, and Qiang Ji. A generative restricted boltzmann machine based method for high-dimensional motion data modeling. *Computer Vision and Image Understanding*, 136:14–22, 2015.

[129] Juan Carlos Niebles, Chih Wei Chen, and Li Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 392–405. Springer Berlin Heidelberg, 2010.

[130] Gabriel L. Oliveira, Wolfram Burgard, and Thomas Brox. Efficient deep models for monocular road segmentation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4885–4891, October 2016.

[131] Dan Oneata, Jakob Verbeek, and Cordelia Schmid. Action and event recognition with fisher vectors on a compact feature set. *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1817–1824, December 2013.

[132] Dan Oneata, Jakob Verbeek, and Cordelia Schmid. The LEAR submission at THUMOS 2014. Technical report, INRIA Grenoble - Rhône-Alpes Research Centre, 2014.

[133] Naveen Onkarappa and Angel D. Sappa. Synthetic sequences and ground-truth flow field generation for algorithm validation. *Multimedia Tools and Applications*, 74(9):3121–3135, May 2015.

[134] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1717–1724, 2014.

[135] Paul Over, George Awad, Travis Rose, and Jon Fiscus. TRECVID 2008 – goals, tasks, data, evaluation mechanisms and metrics. *Evaluation*, pages 1–32, 2009.

[136] Jeremie Papon and Markus Schoeler. Semantic pose using deep networks trained on synthetic RGB-D. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 774–782, December 2015.

[137] Alonso Patron-Perez, Marcin Marszalek, Ian Reid, and Andrew Zissermann. Structured learning of human interaction in TV shows. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 34(12):2441–2453, December 2012.

[138] Alonso Patron-Perez, Marcin Marszalek, Andrew Zisserman, and Ian Reid. High Five: Recognising human interactions in TV shows. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 50.1–50.11, 2010. doi:10.5244/C.24.50.

[139] Mattis Paulin, Jérôme Revaud, Zaid Harchaoui, Florent Perronnin, and Cordelia Schmid. Transformation pursuit for image classification. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3646–3653. IEEE, September 2014.

[140] Xiaojiang Peng, Limin Wang, Xingxing Wang, and Yu 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.

[141] Xiaojiang Peng, Changqing Zou, Yu Qiao, and Qiang Peng. Action recognition with stacked fisher vectors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 581–595. Springer International Publishing, 2014.

[142] Xingchao Peng, Baochen Sun, Karim Ali, and Kate Saenko. Learning deep object detectors from 3D models. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1278–1286, December 2015.

[143] Bojan Pepik, Michael Stark, Peter Gehler, and Bernt Schiele. Teaching 3D geometry to deformable part models. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3362–3369. IEEE, June 2012.

[144] Ken Perlin. Real time responsive animation with personality. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):5–15, March 1995.

[145] Ken Perlin and Gerry Seidman. Autonomous digital actors. In *Motion in Games*, pages 246–255. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[146] Florent Perronnin, Zeynep Akata, Zaid Harchaoui, and Cordelia Schmid. Towards good practice in large-scale learning for image classification. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3482–3489, June 2012.

[147] Florent Perronnin and Christopher Dance. Fisher kernels on visual vocabularies for image categorization. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, June 2007.

[148] Florent Perronnin and Diane Larlus. Fisher vectors meet neural networks: A hybrid classification architecture. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3743–3752. IEEE, June 2015.

[149] Florent Perronnin, Yan Liu, Jorge Sanchez, and Herve Poirier. Large-scale image retrieval with compressed fisher vectors. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3384–3391. IEEE, June 2010.

[150] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the Fisher kernel for large-scale image classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 143–156. Springer Berlin Heidelberg, 2010.

[151] Alessandro Pieropan, Giampiero Salvi, Karl Pauwels, and Hedvig Kjellstrom. Audio-visual classification and detection of human manipulation actions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3045–3052. IEEE, September 2014.

[152] Leonid Pishchulin, Arjun Jain, Christian Wojek, Mykhaylo Andriluka, Thorsten Thormählen, and Bernt Schiele. Learning people detection models from few training samples. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1473–1480. IEEE, June 2011.

[153] Tomaso Poggio and Simon Smale. The mathematics of learning: dealing with data. *Notices of the American Mathematical Society*, 50(May):537–544, 2003.

[154] Ronald Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976–990, June 2010.

[155] Terry Regier, Paul Kay, and Naveen Khetarpal. Color naming reflects optimal partitions of color space. *Proceedings of the National Academy of Sciences*, 104(4):1436–1441, 2007.

[156] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Koltun Vladlen. Playing for data: Ground truth from computer games. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 102–118. Springer International Publishing, 2016.

[157] Tobias Ritschel, Thorsten Grosch, and Hans-Peter Seidel. Approximating dynamic global illumination in image space. *Proceedings of the 2009 symposium on Interactive 3D graphics and games - I3D '09*, page 75, 2009.

[158] German Ros, Laura Sellart, Joanna Materzyska, David Vázquez, and Antonio M. López. The SYNTHIA dataset: a large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3234–3243. IEEE, June 2016.

[159] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, December 2000.

[160] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, December 2015.

[161] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *Computing Research Repository (CoRR)*, arvix:abs/1511.06295, 2015.

[162] Jorge Sánchez, Florent Perronnin, and Teófilo de Campos. Modeling the spatial layout of images beyond spatial pyramids. *Pattern Recognition Letters*, 33(16):2216–2223, 2012.

[163] Scott Satkin, Jason Lin, and Martial Hebert. Data-driven scene understanding from 3D models. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2012.

[164] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. In *Proceedings of the International Conference in Pattern Recognition*, volume 3, pages 32–36. IEEE, 2004.

[165] David Sculley. Web-scale k-means clustering. In *Proceedings of the 19th International Conference on World Wide Web*, pages 1177–1178. ACM, 2010.

[166] Jeremy Selan. Cinematic color. *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 1–54, 2012.

[167] Laura Sevilla-Lara, Yiyi Liao, Fatma Guney, Varun Jampani, Andreas Geiger, and Michael J Black. On the integration of optical flow and action recognition. *Computing Research Repository (CoRR)*, arXiv:abs/1712.08416, 2017.

[168] Alireza Shafaei, James J. Little, and Mark Schmidt. Play and learn: Using video games to train computer vision models. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 26–1, September 2016.

[169] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipmanand, and Andrew Blake. Real-time human pose recognition in parts from a single depth image. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1297–1304. IEEE, June 2011.

[170] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 568–576. Curran Associates, Inc., 2014.

[171] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Computing Research Repository (CoRR)*, arXiv:abs/1409.1556, September 2014.

[172] Elena Sizikova1, Vivek K. Singh, Bogdan Georgescu, Maciej Halber, Kai Ma, and Terrence Chen. Enhancing place recognition using joint intensity - depth analysis and synthetic data. In *Proceedings of the European Conference on*

Computer Vision Workshops, pages 901–908. Springer International Publishing, 2016.

[173] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *Computing Research Repository (CoRR)*, arXiv:1212.0402, November 2012.

[174] Tiago Sousa, Nick Kasyan, and Nicolas Schulz. Secrets of cryengine 3 graphics technology. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2011.

[175] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal on Machine Learning Research*, 15:1929–1958, 2014.

[176] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using LSTMs. *Proceedings of Machine Learning Research*, 37:843–852, July 2015.

[177] Bernhard Steiner. Post processing effects. Institute of Graphics and Algorithms, Vienna University of Technology. Bachelour's thesis, August 2011.

[178] Hao Su, Charles R. Qi, Yangyan Yi, and Leonidas Guibas. Render for CNN: viewpoint estimation in images using CNNs trained with rendered 3D model views. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015.

[179] Hao Su, Fan Wang, Yangyan Yi, and Leonidas Guibas. 3D-assisted feature synthesis for novel views of an object. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2677–2685, December 2015.

[180] Baochen Sun and Kate Saenko. From virtual to reality: Fast adaptation of virtual object detectors to real domains. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2014.

[181] Lin Sun, Kui Jia, Kevin Chen, Dit Yan Yeung, Bertram E Shi, and Silvio Savarese. Lattice long short-term memory for human action recognition. *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2147–2156, October 2017.

[182] Lin Sun, Kui Jia, Dit-Yan Yeung, and Bertram E. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 4597–4605, December 2015.

[183] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9. IEEE, June 2015.

[184] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826. IEEE, June 2016.

[185] Geoffrey R. Taylor, Andrew J. Chosak, and Paul C. Brewer. OVVV: Using virtual worlds to design and evaluate surveillance systems. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, June 2007.

[186] Sinisa Todorovic. Human activities as stochastic kronecker graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 7573 LNCS, pages 130–143. Springer, Berlin, Heidelberg, 2012.

[187] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3D convolutional networks. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4489–4497. IEEE, December 2015.

[188] Saburo Tsuji, Akira Morizono, and Shinichi Kuroda. Understanding a simple cartoon film by a computer vision system. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 2 of *IJCAI*, pages 609–610. Morgan Kaufmann Publishers Inc., 1977.

[189] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *Computing Research Repository (CoRR)*, arXiv:abs/1509.06461(2):1–5, 2015.

[190] Herwin Van Welbergen, Ben JH Van Basten, Arjan Egges, Zs M Ruttkay, and Mark H Overmars. Real time character animation: A trade-off between naturalness and control. *Proceedings of the Eurographics*, pages 45–72, 2009.

[191] Vladimir Vapnik. Learning using privileged information: Similarity control and knowledge transfer. *Journal on Machine Learning Research*, 16:2023–2049, 2015.

[192] David Vazquez, Antonio M. López, Javier Marín, Daniel Ponsa, and David Gerónimo. Virtual and real world adaptation for pedestrian detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 36(4):797–809, April 2014.

[193] David Vázquez, Antonio M. López, Daniel Ponsa, and Javier Marín. Cool world: domain adaptation of virtual and real worlds for human detection using active learning. In *Proceedings of Neural Information Processing Systems Workshops*, 2011.

[194] Ramakrishna Vedantam, Xiao Lin, Tanmay Batra, C. Lawrence Zitnick, and Devi Parikh. Learning common sense through visual abstraction. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2542–2550, December 2015.

[195] V.S.R. Veeravasarapu, Rudra Narayan Hota, Constantin Rothkopf, and Ramesh Visvanathan. Model validation for vision systems via graphics simulation. *Computing Research Repository (CoRR)*, arXiv:1512.01401, 2015.

[196] V.S.R. Veeravasarapu, Rudra Narayan Hota, Constantin Rothkopf, and Ramesh Visvanathan. Simulations for validation of vision systems. *Computing Research Repository (CoRR)*, arXiv:1512.01030, 2015.

[197] V.S.R. Veeravasarapu, Constantin Rothkopf, and Ramesh Visvanathan. Model-driven simulations for deep convolutional neural networks. *Computing Research Repository (CoRR)*, arXiv:1605.09582, 2016.

[198] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Proceedings of Neural Information Processing Systems (NIPS)*, 2016.

[199] Michalis Vrigkas, Christophoros Nikou, and Ioannis A. Kakadiaris. A review of human activity recognition methods. *Frontiers in Robotics and AI*, 2:1–28, 2015.

[200] Hanli Wang, Yun Yi, and Jun Wu. Human action recognition with trajectory based covariance descriptor in unconstrained videos. In *Proceedings of the ACM International Conference on Multimedia (ACM-MM)*, MM, pages 1175–1178, New York, NY, USA, 2015. ACM.

[201] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng Lin Liu. Action recognition by dense trajectories. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3169–3176. IEEE, June 2011.

[202] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Dense trajectories and motion boundary descriptors for action recognition. *International Journal of Computer Vision (IJCV)*, 103:60–79, May 2013.

[203] Heng Wang, Dan Oneata, Jakob Verbeek, and Cordelia Schmid. A robust and efficient video representation for action recognition. *International Journal of Computer Vision (IJCV)*, 119(3):219—-238, September 2016.

[204] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 3551–3558, December 2013.

[205] Heng Wang and Cordelia Schmid. LEAR-INRIA submission for the THUMOS workshop. In *ICCV workshop on action recognition with a large number of classes*, volume 2, pages 8–11, Jan 2013.

[206] Heng Wang, Muhammad Muneeb Ullah, Alexander Klaser, Ivan Laptev, and Cordelia Schmid. Evaluation of local spatio-temporal features for action recognition. *Proceedings of the British Machine Vision Conference (BMVC)*, pages 124.1–124.11, September 2009.

[207] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4305–4314. IEEE, June 2015.

[208] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 20–36. Springer International Publishing, 2016.

[209] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks for action recognition in videos. *Computing Research Repository (CoRR)*, arXiv:abs/1705.02953, 2017.

[210] Pichao Wang, Wanqing Li, Zhimin Gao, Yuyao Zhang, Chang Tang, and Philip Ogunbona. Scene flow to action map: A new representation for RGB-D based action recognition with convolutional neural networks. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 416–425. IEEE, July 2017.

[211] Xiaolong Wang, Ali Farhadi, and Abhinav Gupta. Actions ~ Transformations. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016.

[212] David H Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[213] Zuxuan Wu, Yu gang Jiang, Xi Wang, Hao Ye, and Xiangyang Xue. Multi-stream multi-class fusion of deep networks for video classification. In *Proceedings of the ACM International Conference on Multimedia (ACM-MM)*, MM, pages 791–800. ACM, 2016.

[214] Zuxuan Wu, Xi Wang, Yu-Gang Jiang, Hao Ye, and Xiangyang Xue. Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. In *Proceedings of the ACM International Conference on Multimedia (ACM-MM)*, MM, pages 461–470, 2015.

[215] Jiaolong Xu, Sebastian Ramos, David Vázquez, and Antonio M. López. Domain adaptation of deformable part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 36(12):2367–2380, December 2014.

[216] Jiaolong Xu, David Vázquez, Antonio M. López, Javier Marín, and Daniel Ponsa. Learning a part-based pedestrian detector in a virtual world. *T-ITS*, 15(5):2121–2131, October 2014.

[217] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Proceedings of Neural Information Processing Systems (NIPS)*, 27:3320–3328, November 2014.

[218] Mengyang Yu, Li Liu, and Ling Shao. Structure-preserving binary representations for RGB-D action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 38(8):1651–1664, August 2016.

[219] Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime tv-l1 optical flow. In *Proceedings of the 29th DAGM Conference on Pattern Recognition*, pages 214–223. Springer-Verlag, 2007.

[220] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *Computing Research Repository (CoRR)*, arXiv:abs/1311.2901, 2013.

[221] Shengxin Zha, Florian Luisier, Walter Andrews, Nitish Srivastava, and Ruslan Salakhutdinov. Exploiting image-trained CNN architectures for unconstrained video classification. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 60.1–60.13, 2015.

[222] Jianguo Zhang, Marcin Marszalek, Svetlana Lazebnik, and Cordelia Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International Journal of Computer Vision (IJCV)*, 73(2):213–238, June 2007.

[223] Yuanjie Zheng, Stephen Lin, Chandra Kambhamettu, Jingyi Yu, and Sing Bing Kang. Single-image vignetting correction. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, volume 31, pages 2243–2256, December 2009.

[224] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5122–5130. IEEE, July 2017.

[225] Shusen Zhou, Qingcai Chen, and Xiaolong Wang. Active deep networks for semi-supervised sentiment classification. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1515–1523. Association for Computational Linguistics, 2010.

[226] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: Many could be better than all. *Artificial Intelligence*, 137:239–263, May 2002.

[227] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the International Conference on Computer Vision (ICCV)*. IEEE, 2017.

[228] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3357–3364, May 2017.

[229] C. Lawrence Zitnick, Ramakrishna Vedantam, and Devi Parikh. Adopting abstract images for semantic scene understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 38(4):627–638, April 2016.

[230] Mohammadreza Zolfaghari, Gabriel L. Oliveira, Nima Sedaghat, and Thomas Brox. Chained multi-stream networks exploiting pose, motion, and appearance for action classification and detection. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.