TÉCNICO LISBOA

# UNIVERSIDADE DE LISBOA

# INSTITUTO SUPERIOR TÉCNICO

## Leveraging Context for Multi-Label Action Recognition and Detection in Video

João Guilherme Antunes Martins

**Supervisors:** Doctor Daniel P. Siewiorek
Doctor Asim Smailagic
Doctor Alexandre Bernardino

Thesis approved in public session to obtain the PhD Degree in Electrical and Computer Engineering

2018

Cofinanciado por:
REPÚBLICA PORTUGUESA | CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR
FCT Fundação para a Ciência e a Tecnologia
POCH
PORTUGAL 2020
UNIÃO EUROPEIA Fundo Social Europeu

TÉCNICO LISBOA

Carnegie Mellon University

# UNIVERSIDADE DE LISBOA

# INSTITUTO SUPERIOR TÉCNICO

# CARNEGIE MELLON UNIVERSITY

Leveraging Context for Multi-Label Action Recognition and Detection in Video

João Guilherme Antunes Martins

**Supervisors:** Doctor Daniel P. Siewiorek
Doctor Asim Smailagic
Doctor Alexandre Bernardino

Thesis approved in public session to obtain the PhD Degree in Electrical and Computer Engineering

**Jury**:
Daniel P. Siewiorek, Carnegie Mellon University, Pittsburgh, USA
Asim Smailagic, Carnegie Mellon University, Pittsburgh, USA
Alexandre Bernardino, Universidade Técnica de Lisboa, Portugal
Mário Figueiredo, Universidade Técnica de Lisboa, Portugal
David Held, Carnegie Mellon University, Pittsburgh, USA

**Funding Institutions**:
Fundação para a Ciência e a Tecnologia

2018

Cofinanciado por:
REPÚBLICA PORTUGUESA | CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR
FCT Fundação para a Ciência e a Tecnologia
POCH
PORTUGAL 2020
UNIÃO EUROPEIA Fundo Social Europeu

**Thesis committee:**

Daniel P. Siewiorek, chair

Asim Smailagic

Alexandre Bernardino

Mário Figueiredo

David Held

## Acknowledgments

This thesis would not be possible if not for the support of the CMU-PT portugal program. They are giving chances to people like me, who would never dream of having the opportunity of pursuing a dual PhD degree with as much support as they provide, and for that I wish to express my gratitude. I was the happy recipient of grant SFRH/BD/106068/2015, which funded this work.

I would also like to thank my committee members David Held and Mário Figueiredo for giving direction to the research presented to them. Their valuable input shaped the direction of this thesis in no small way.

To my advisors Asim Smailagic, Daniel Siewiorek and Alexandre Bernardino I wish to thank them for facing every challenge I brought them head on. In five years of challenges and requests I was not told to give up on an idea once. Every difficulty I encountered was met with support, solutions and new challenges, and I can not devise a better way to experience a PhD than this.

Finally, to my grandmother Lourdes, my mother Edna, my brother Rui, and my partner Joana I wish to thank them for their company on my journey. They are simultaneously passenger and driver, sharing in the ride and shaping it at the same time.

## Abstract

This thesis addresses video-based multi-person, multi-label, spatiotemporal action detection and recognition. This is a challenging problem because each person can be performing several actions at the same time (*e.g.* talking and walking), and simultaneously other actors can be performing different actions. We claim that these are problems where the use of contextual information (*e.g.* semantic descriptions of the scene) may lead to significant performance improvements. In this work, we develop several approaches to tackle this problem and validate them in challenging datasets. We propose a framework to integrate and test multiple sources of contextual information in video-based multi-person, multi-label, spatiotemporal action detection and recognition. We highlight six contributions, and that are collected in three publications (at different stages of publication at the time of this writing). The first contribution is a proposed Multisource Video Classification (MVC) framework that allows the combination of several sources of context information, for which we consider four types: actor centric input filtering (a way to focus attention on the actor under analysis but still gather appearance information from the neighborhood), semantic neighbor context (a way to inform the model with the actions performed by nearby agents), object detection (how objects interacting with the actor can inform about its action) and pose data (how high level features extracted from the actor can help the classification process). The second contribution is a foveated approach to actor centric filtering for input selection that weights the appearance information in a decreasing way, from the center to the periphery of the actor bounding box. The third contribution is a novel encoding for the semantic neighbor context and its custom classifier with spatial and temporal dependence. The fourth is a custom Hybrid Sigmoid-Softmax loss function for the multi-class/multi-label case, that combines the cross-entropy loss typical of multi-class problems with the sum-of-sigmoids loss used in the multi label case. The fifth is the application of the developed methods to a challenging dataset with a large number of videos with mul-

tiple agents performing multiple actions, with 80 heterogeneous and highly unbalanced classes. To allow research with reasonable computer power, we have created the mini-AVA, a partition of AVA that maintains temporal continuity and class distribution with only one tenth of the dataset size. The sixth contribution is a collection of ablation studies on alternative actor centric filters and semantic neighbor context classifiers. From this research we achieve a relative mAP improvement of $18.8\%$ using our foveated actor centric filtering, relative mAP improvement of $5\%$ using our semantic neighbor context embedding and models, and a relative mAP improvement of $12.6\%$ using our custom Hybrid Sigmoid-Softmax loss.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| **CNN** | Convolutional Neural Network |
| **IoU** | Intersection over Union |
| **mAP** | mean Average Precision |
| **OF** | Optical Flow |
| **RNN** | Recurrent Neural Network |
| **LSTM** | Long short-term memory |
| **SVM** | Support Vector Machines |
| **GBB** | Gaussian Background Blur |
| **PCA** | Principal Component Analysis |
| **SIFT** | Scale-invariant feature transform |
| **NLP** | Natural language processing |
| **HMM** | Hidden Markov Model |
| **GMM** | Gaussian Mixture Model |

"Facts are meaningless. You could use facts to prove anything that's even remotely true"

*- Homer Simpson*

# Chapter 1

# Introduction

This thesis focuses on action detection in video. We propose a new approach to this classification task by attempting to insert contextual information in the classification procedure to increase performance at a reduced computational cost increase. In this introduction chapter, we first describe why this task and approach are important in Section 1.1. We then explain some essential topics for the action detection task in Section 1.2. Following that, we explain the approach that will be discussed in this thesis in Section 1.3. We list the contributions from this work in Section 1.4 and conclude the introduction by describing the thesis outline in Section 1.5.

## 1.1 Motivation

Recently, video based action detection and recognition has been the main focus of many researchers and companies. Its diverse applications in fields like robotics (*e.g.* identify agents to interact with), self-driving cars (*e.g.* identify pedestrians) and surveillance/security (*e.g.* predict actions)[4] are certainly driving significant efforts. This could be also attributed to the ever increasing computational power availability coupled with the surfacing of bigger and more complex datasets[53]. As the complexity in the action labelling description in-

creases, the computational cost of state-of-the-art classification model becomes prohibitive. With each agent being allowed to perform more than one action at once, the complexity of the output space becomes exponential in the number of actions considered. In this thesis, we address the problem of action detection and recognition in video in scenarios where each subject may be performing several actions at once, with an arbitrary amount of people simultaneously, dubbed Multi-Label Multi-Person Action Detection and Recognition. We study and test how to extract and leverage auxiliary information from video, called context. The main goal is to determine how to leverage this information, how to identify it, and how to rank it in accordance with its cost-benefit trade-off (*i.e.* its computational cost vs its performance increase).

## 1.2 Topic Overview

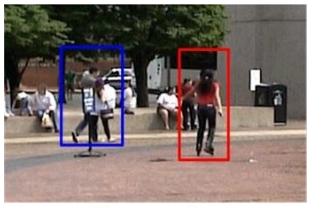
In this section, describe fundamental topics that this thesis hinges on are described. First, we make the distinction between the Action Recognition and Action detection Tasks. Then the differences between Multi-Class and Multi-Label Classification are explained. We then explain the basic concepts behind Convolutional Neural Networks, followed by an explanation of how sequential classification differs from non-sequential classification.. Following that the Multi-Label Multi-Person Spatiotemporal Action Detection and Recognition tasks are explained. Finally, we discuss the concepts of appearance context and semantic context.

### 1.2.1 Action Recognition vs Action Detection

Action recognition[6] (also called action classification) pertains to the task of matching one video to the label that best describes the action being performed. Typically, the video has been trimmed to show the relevant action and nothing else (see Fig. 1.1 (a) ) . In action detection, the number of actions and number of agents performing those actions

are not determined *a priori*. In fact, even a single agent might be performing more than one action. There might also be sections of the video (in either the space domain or the time domain) in which no action is being performed. The goal of action detection[6] is not only to ascertain which actions are being performed, but also to match them with their spatial and/or temporal boundaries. The spatial boundaries are usually in the form of a bounding box (see Fig. 1.1 (b)), selecting the region of the image/video in which the target is enclosed. The temporal boundaries are usually in the form of a time window inside which the detected action is being performed (see Fig. 1.1 (c)). In action detection tasks, videos are usually untrimmed, either containing more than one action being shown (in this case the classifier must identify each action and its temporal bounding boxes) or segments of video with no action of interest (in this case, the classifier must identify in which segments actions of interest are being depicted) or all of those at the same time (*i.e.* multiple actions per videos with segments with no actions as well). This task is intrinsically more complex, as it demands a superset of information to be provided on the output: not only must the classifier determine which action is being depicted it must also determine when (temporal boundary) and where (spatial boundary) it was detected.


(a) Example action recognition



(b) Example action spatial detection

(c) Example action temporal detection

Figure 1.1: Real examples illustrating each type of task. Adapted from https://ghassanalregib.com/demos/, http://www.bu.edu/ids/research-projects/action-recognition/ and https://www.researchgate.net/figure/An-example-of-temporal-action-detection respectively.

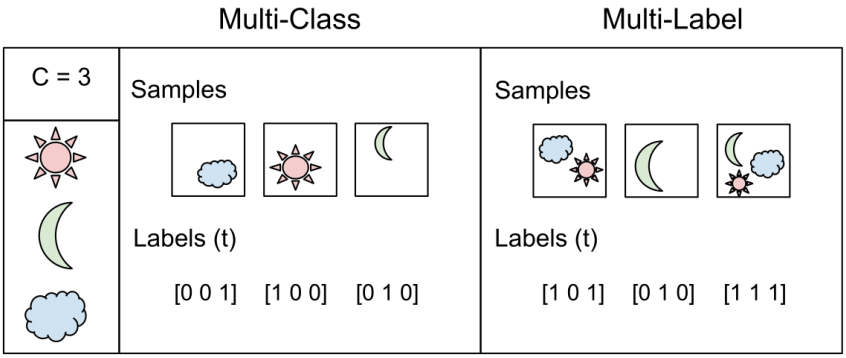



Figure 1.2: Examples of Multi-Class and Multi-Label classification scenarios. In the Multi-Class scenario each sample is attributed one of C labels. The label space is of size C. In the Multi-Label scenario each sample is attributed any combination of labels. The label space is of size $2^C$. (Adapted from https://gombru.github.io/2018/05/23/cross_entropy_loss/)

### 1.2.2 Multi-Class vs Multi-Label tasks

The most common supervised learning scenario is the Multi-class setting. In a problem with C classes, the Multi-Class task needs only to determine which of the C classes should be attributed to a given input. As such, there are C possible output scenarios to be considered. However, in the Multi-Label the space of possible label outputs increases greatly. For a problem with C classes,the label space is comprised of its power set (*i.e.* the label space is comprised of the combination of all possible groups of labels) which spans $2^C$ options. In Fig. 1.2 an example is shown of the difference in labelling complexity.

### 1.2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNN) appear as an approach to do classification on images/video. These media are extremely high-dimensional inputs, and fully connected networks scale very poorly on input size, becoming intractable quickly. As a quick example, if the input were to be an image of size 150 x 150 pixels, that would equate to 22500 inputs. If the image has color, that amount triples (because it would have three channels:

red, green and blue). If we add the time dimension (going from a single image to video) then we have to also multiply that by the number of frames. Finally, in most Neural Network based architectures several layers are employed. It is easy to see how the number of parameters skyrockets in this application. CNNs approach these problems by employing local connectivity, parameter sharing and pooling/subsampling to deal not only with the high dimensionality problem, but also to exploit the underlying 2D (3D for video) topology of pixels (*i.e.* a pixel's positional relationship with its neighboring pixels) as well as invariance properties that can be expected (*e.g.* translation invariance, changes in illumination). The local connectivity property relates to the fact that the weights in a CNN are not connected to all of the input, rather being connected only to a subsection of it, called the receptive field. Note that the same receptive field is used across all channels (for example, if a hidden unit is connected to the top left corner of a color image, it will receive as input the top left corner of all three color channels of that image). This property greatly reduces the total number of parameters necessary for the model. These weights are called the kernels of the CNN. Parameter sharing refers to the fact that each kernel is going to be used with the same weights several times across the input. For each kernel employed, the same weights are going to be convoluted across the image, forming a feature map. Finally pooling/subsampling layers simply take a feature map, and reduce its dimension. The most common pooling layer is max-pooling, in which only the maximum value within a given window is preserved and the rest is discarded. This further reduces the number of parameters of these models.

An example of how a CNN works is now shown. Let $n_C$ denote the number of channels in the input, and $f$ denote the size of our square kernel. Each kernel has then $f * f * n_C$ parameters. Let $n_K$ be the number of kernels. Let $W$ be the width of the input image and $H$ be the input height.

In Fig. 1.3 is an example where the input has only one channel, so its dimension is $W =$

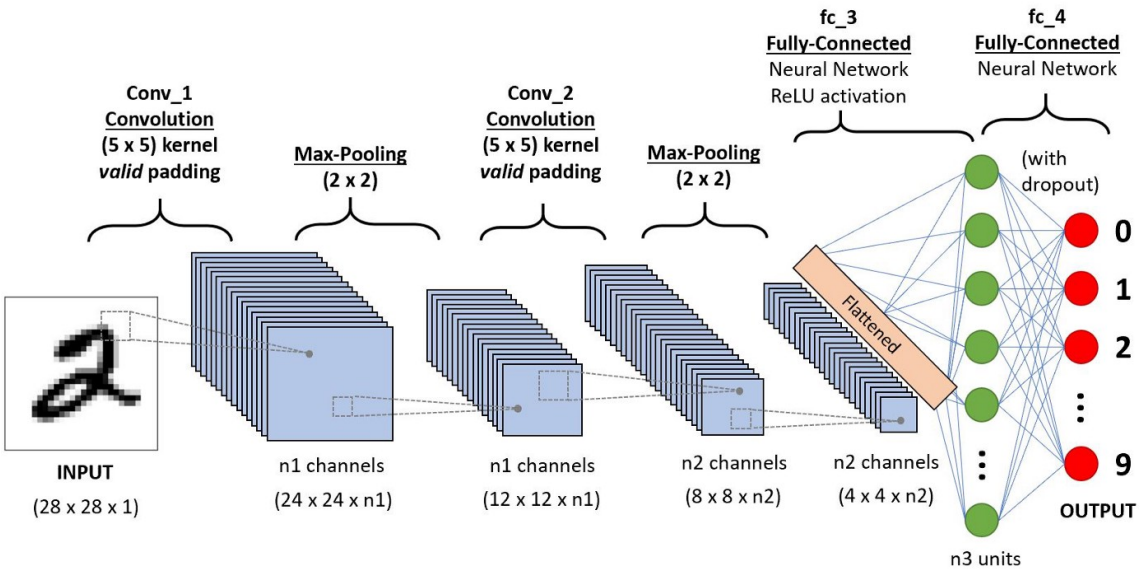



Figure 1.3: Example of a simple CNN architecture. Taken from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

28, $H = 28$, and the full input dimensions are $28 \times 28 \times 1$. The number of kernels is $n_K = n1$ and the kernel size is $5 \times 5 \times 1$. (Since the number of channels is one it will be ommitted for clarity). The size of each dimension of the output is calculated as $\frac{D-F+2\times P}{S} + 1$, where $D$ is the dimension of the input currently being used, $F$ is the dimension of the kernel, P is the amount of zero padding used ( how many zeros are added to the input. In this case, zero, also called valid padding ) and $S$ is the stride (in this case, one). Since in this case both the input image and the kernel are square, the dimension of the output of one kernel convolution in the first layer is 24x24. Stacking all $n1$ kernels, we get the $24 \times 24 \times n1$ dimension found in the image.

In typical CNN architectures, convolution layers are followed by pooling layers, and this is repeated for as many times as desired. Near the output layers the feature maps are flattened into a vector (see Fig 1.3) to be classified by fully connected output layers. The

early layers of this type of architecture are capturing the feature map activations (calculating where each kernel is activated in the image to extract patterns), with increasingly more complex patterns being captured as the layers get deeper in the architecture (simple edges and shapes in the early layers vs a face or an eye in late layers, for example).

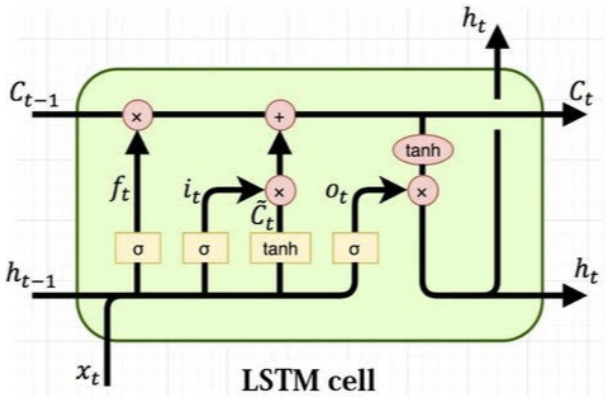
### 1.2.4 Sequential Classification

Sequential classification introduces the notion of memory to classification systems. Instead of taking pairs of inputs/outputs and processing them by themselves, sequential classification methods retain a representation of past learning instances and use it to inform the current classification task. This is achieved in several different ways, *e.g.* in Hidden Markov Model (HMM)[7] the transition probabilities between two output states is learned, and the current classification step is a function of the last output state in addition to the current input. In Recurrent Neural Network (RNN)[8] an internal representation of all the classification steps done so far is kept and used to inform each new classification instance.

One distinction that is important to make is Sequence Classification *vs* Sequential Classification. This is best done using an example: a classifier that received a video volume (*i.e.* a sequence of frames) as input, and classifies it (*e.g.* a 3D-CNN) is not performing sequential classification, it is merely classifying a sequence. Sequential classification requires past classification instances to have an influence on the current classification instance, via the concept of memory. Classifying one sequence taking into account nothing but that sequence is not enough.

The sequential classification paradigm comes naturally when doing classification in any task that includes a time domain, like action classification in video. It is intuitive to expect past actions to be pertinent in determining current actions.

When dealing with long sequences, propagating internal states through long chains of timesteps can lead to a hard to balance problem: How much influence should information




$$(1)\quad i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$$
$$(2)\quad f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$$
$$(3)\quad o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$$
$$(4)\quad \tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$
$$(5)\quad C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$
$$(6)\quad h_t = \tanh(C_t) * o_t$$

Figure 1.4: Internal structure of an LSTM cell. Adapted from[1]

that was processed a long time ago have over the current classification step? Classical RNN architectures simply let its influence decay across time. Long short-term memory (LSTM)[9] architectures possess internal gates that model the internal representation learned to discern what to forget and what to remember.

In Figure 1.4 an example is shown of the internal structure of an LSTM cell, along with the equations used to calculate the internal state propagation, where $x_t$ is the input received at timestep $t$, $h_{t-1}$ and $h_t$ are the internal states at timesteps $t-1$ and $t$, respectively. $C_{t-1}$ and $C_t$ are the cell states used to calculate the internal states using the equations that will be explained shortly. $f_t$ is the result passed by the forget gate. This gate consists of a sigmoid layer that combines the input $x_t$ and the previous internal state $h_t-1$ using learned weights ($U^f$ and $W^f$) to determine which information to forget (*i.e.* which entries to delete from the cell state vector). This is shown in the second equation in Figure 1.4. This is the gate responsible by learning which information found in the past no longer has predictive power for current and future classification steps (a unique feature of LSTM architectures). The following step is to decide which new information is saved in the next cell state $C_t$. This process has two steps. First, the input gate decides which entries in the cell state should be updated using a sigmoid function receiving the input ($x_t$), the previous internal

state ($h_{t-1}$) and learned weights ($U^i$ and $W^i$). Then, the candidate values to be inserted are calculated using a hyperbolic tangent function using the same values but with separate weights. (equations one and four in 1.4). Then, the new cell state is calculated by forgetting the valued defined by the forget gate and refreshing the values defined by the input gate (equation five in 1.4). This cell state will be passed to the next LSTM unit classifying the next timestep. Finally, the output has to be calculated. Note that the output is used both as an output for deeper layers (the upward $h_t$ arrow in 1.4) but also as the next internal state (the $h_t$ arrow pointing right in 1.4). The output is calculated using a gate that applies a sigmoid function to the input ($x_t$) and previous internal state ($h_{t-1}$) along with learned weight matrices $U^o$ and $W^o$ (see equation three in 1.4). The output is then calculated by multiplying the values found by this gate with a hyperbolic tangent function applied on the new cell state $C_t$.

The weights learned to model how the internal and cell states decide which information to store and which information to delete are a property unique to LSTMs. These weights and gates are what allow LSTMs to classify long sequences without vanishing gradient problems regarding long term dependencies in input data sequences.

### 1.2.5 Multi-label Multi-Person Spatiotemporal Action Detection and Recognition

With the previous topics defined, the task that is tackled in this work can now be defined.

Action Recognition has the goal of matching one video with one label. In this task, the goal is to identify the action being shown. Given the same video, the goal of action detection would be to determine the actions being performed, but also when (in which frames?) and where (in which pixels?). This task deals with untrimmed videos, where discerning in which segments of the video the action is being performed is also part of the task.

If the video does not have the constraint of having only one person the task now becomes more complex, since more than one action must be identified, along with its corresponding time and space segmentations. If each person is allowed to perform multiple actions the problem becomes multi-label.

So, for the multi-person, multi-label, spatiotemporal action detection and recognition task in video, the goal is to receive as input a video, and to return a list of actions coupled with bounding boxes in both space and time, depicting every action of every person depicted in the video.

### 1.2.6 Appearance Context and Semantic Context

The term context is severely overloaded, and its use is widespread among different applications. Just in the action recognition/detection community it is used to represent vastly different concepts, and a clear cut definition is not available. For example, while Sun *et al.*[10] and Zhang *et al.*[11] both use context (in action recognition) to refer to motion descriptors, Liu *et al.*[12] use it to refer to the hidden state of an LSTM trained on Skeleton data. Wu *et al.*[13] use context to mean feature descriptors learned from Gaussian Mixture Model (GMM). Han *et al.*[14] use the word context to refer to a description of the objects present in a scene.

Although these definitions do not match, their meaning can be inferred to mean *any information that can be extracted from video data that is not the actual pixel values.*. This definition agrees with the usage in all of the works mentioned above, as well as our own usage. We define context to mean whichever information can be extracted from the raw RGB pixels to aid in the classification procedure.

In this thesis, two types of context to use in Action Detection are studied: Appearance Context and Semantic Context. IWedenote appearance context as the information that can be leveraged by manipulating the RGB frames from video to establish a hierarchy between regions. Specifically for Action Detection, a task that is largely human-centric,

Appearance Context is used to direct the classifiers to the regions on interest in the input space,*i.e.* the humans performing the actions to detect. In particular, in the multi-person scenario, directing classifiers to the regions of interest for the current classification task yields significant performance gains.

Semantic Context leverages the information that can be extracted from the actions that do not belong to the classification target. In the multi-person scenario, where multiple people are performing actions that must be detected, the actions depicted in proximity may be related. A richer input description of the target can be obtained by informing a classifier of the action universe surrounding the target. Due to the fact that this representation is semantic (*i.e.* using an embedding), this classification approach has a very low-dimensional input space, making it very light in computational requirements, while possessing significant predictive power.

## 1.3 Approach

The scrutiny of the video understanding community on the action detection task shows not only the many applications of developing an accurate human action centric classifier, but also the complexity of the task. It is far from a solved problem, where even the most computationally intensive models exhibit modest performance. The multi-person multi-label constraints introduced by this type of task come aggregated with challenges like high output dimension (see Section 1.2.2, actions that are harder to distinguish (*e.g.* picking up an object vs putting down an object[3]), and requiring a more detailed understanding of the video than the action recognition problem (*i.e.* having to determine a multitude of actions per video, as opposed to classifying the high level thematic of the video[15]).

When ImageNet[16] appeared, image classification performance increased dramatically. The features learned from data greatly outperformed hand-crafted features and the methods in the state-of-the-art quickly converged to CNNs. However, such a dramatic paradigm

shift has yet to be discovered for the video understanding problem[17]. 3D-CNN models[18][19] are currently the approach with the best performance, but the gap between data-driven feature extraction and hand-crafted features is surmountable. Also, the performance of 2D-CNN models applied to still video frames shows competitive performance with 3D-CNN's, despite not incorporating any type of temporal information modelling. Furthermore, 2D-CNN models have fewer parameters to train, making them computationally more tractable. Architectures such as two-stream 2D-CNN's ([20]), that take RGB and Optical Flow as input, offer a good balance between having temporal information encoding on optical flow along with RGB frames, but still not requiring the immense computational power that 3D-CNN models need.

However, recent datasets like the AVA dataset[3] have increased the complexity of the video understanding task by providing multiple persons and labels per video instance, making it a Multi-Person, Multi-label Spatiotemporal Action detection problem (See Sections 1.2.2 and 1.2.4). The increase in difficulty posed by these new datasets is exemplified in Table 1.1.

| mAP@0.5IoU | UCF101-24 (2013) | JHMDB (2013) | AVA (2018) |
|---|---|---|---|
| Action Detection | 76.3% | 76.7% | 15.6% |
| Actor Detection | 84.8% | 92.8% | 75.3% |
| Gap | 8.5% | 16.1% | 59.7% |

Table 1.1: Taken from[3]. In the results in this table we can not only see how much more challenging the action detection task is in the AVA dataset, the only one of these three that imposes Multi-Label, Multi-person constraints. In addition to that, the actions to be detected in the AVA (*e.g.* opening a door) dataset are much more granular than those in UCF101-24 (*e.g* playing bowling)

We highlight from the information in Table 1.1 that the increase in difficulty is mainly on the action detection task, and not on the actor detector task. We use this as motivation to focus on the action detection task, and leave the actor detector task to other works.

As the action labels get progressively more granular (*i.e.* holding an object *vs* playing

basketball) the use of extra sources of information (named *context* in this domain) is a promising way to improve classification, as proven by our experiments.

The use of context is very promising as a way to improve classification without involving the training of new network architectures for feature extraction. This allows for a performance increase at a relatively cheap computational cost. As the action detection task includes more than one subject simultaneously, the classification procedure now is burdened with an extra constraint: in the input (*e.g.* an RGB frame or an RGB volume) there will be regions that pertain to one target, and regions that do not. Let there be two people in an RGB volume that are performing two different actions. Not only must the classifier detect which actions are being performed, it must also determine which regions of the input will be useful for the classification task (*i.e.* **where** is our target performing the action we are interested in detecting) and which regions will lead the classifier awry (*i.e.* where are the actions that are not helpful for the classification task). To have different regions of the input space showcasing different output options is a challenge unique to action detection, not found in action recognition. The difficulty in data region selection is a limitation found on state-of-the-art two-stream CNN approaches.

In this thesis, the approach of using context to aid state-of-the-art classification is proposed. The main advantages of this approach are threefold: 1) the performance increase comes at almost no extra computational cost (since it implies no extra training or extra layers in the classifiers), 2) this approach can be inserted into almost all the state-of-the-art architectures (the use of extra information can be used in conjunction with the state-of-the-art approaches), 3) the classification step is done by leveraging more information than using only RGB or RGB and Optical Flow. However, it is still unclear where to source context, and a reliable way to test context sources is not available. Without knowing if a context source has predictive power, it is unwise to include it in a system, only to have it yield worse performance. Also, if it does indeed have predictive power, is the perfor-

mance/computational cost trade-off favourable?

We address both the context source testing and data region selection problems. For the context source testing problem we propose our Multisource Video Classification (MVC) framework, a **flexible** over 20 different possible configurations), **extensible** (more context sources can be added without re-structuring), **modular** (each context source can be combined or isolated independently) framework. Two context sources are combined together in experiments: Appearance Context, and Semantic Neighbor Context.

For data region selection we propose that our Appearance Context be used with CNN based classifiers. Our Appearance Context is based on actor centric filtering, and ensures that the classification models we employ use more detailed information in the location of the target than in the surrounding background. This way, in the classification step, data that pertains to the target action is highlighted and everything else is demoted to background. Suppose that in one single RGB frame two people are present. One person is watching TV and the other person is reading a newspaper. If attempting to simultaneously classify this scenario, pixels that depict the person watching TV do not help in classifying the ”reading a newspaper” action. Likewise, pixels depicting the newspaper do not help in classifying the ”watching TV” action. Our appearance context filters highlight the relevant pixels for the current classification step and downplay all others. This avoids ”cross-contamination” of the classification step by other actions being depicted.This is particularly useful in cases where different actors performed different actions at the same time. This mechanism addresses the difficulty in data region selection directly.

Our Semantic Neighbor Context approach is a semantic classification procedure. This is a classification step that uses an embedding and network architecture we designed, and attempts to classify the actions of the target based on the actions of other people in the video (*i.e.* his neighbors). The intuition behind it is: if there is more than one person performing (at least) one action in a video, then those actions might be related. Know-

ing the action of the neighbor(s) of our target might inform our classification procedure about the actions of our target. An easy example is found considering a video of two people talking. Let those people be called John and Mary. If John is listening, it should be easy to conclude that Mary is talking. This is an example of a simple binary relationship between two actions. But there might be action relationships involving more than two people. There might also be relationships with a temporal dependence (*e.g.* opening a door then closing it). In our Semantic Neighbor Context approach, we develop an embedding that takes into account the actions of an arbitrary number of neighbors across space (*i.e.* neighbors are ranked according to their distance to our target) and time (*i.e.* the actions of the neighbors of our target done in the past, in the present frame, and in future frames are all encoded). This makes our Semantic Neighbor Context approach an Embedded input sequential multi-label classification problem. We tackle this problem using an LSTM architecture designed and trained from scratch.

The results obtained show the benefits of using actor centric filters - yielding a relative mAP performance increase of $18.8\%$ - as well as semantic neighbor context (see Section 3.2) - yielding a relative mAP performance increase of $5\%$. In the future, using more context sources like human pose features or object detection could be incorporated for additional performance gains.

Although not included in the proposed MVC framework, we have also researched and published a study on how to leverage human pose features for action classification. These could be included in the MVC framework at a future date, along with the LSTM networks designed for the purpose.

## 1.4 Contributions

There are six main contributions in this work:

- A flexible, extensible, modular, state-of-the-art Framework for testing context sources, allowing for the comparison of different context sources and establishing performance/computational cost trade-off;
- Fovea actor centric filtering, a filtering method mimicking human vision. This technique can be applied with any classification approach, and resulted in significant performance increases on our tests with a state-of-the-art two-stream CNN (18.8% relative performance increase in video mAP on the AVA dataset)
- Semantic neighbor context encoding (relative mAP improvement of 5%) and its custom LSTM classifier with spatial and temporal dependence;
- A custom loss function - Hybrid Sigmoid-Softmax - to be used with the AVA dataset that takes into account this dataset's unique labelling structure (three different label groups, one group comprised of mutually exclusive labels, the other two groups can have between zero and three active labels);
- miniAVA, a partition of the AVA dataset that maintains temporal continuity, keeps the same approximate distribution of the data, while being only 10% of the full dataset;
- Ablation studies on alternative actor centric filtering approaches as well as ablation studies on semantic neighbor context classifiers.

Three research papers have resulted from this work:

- BMVC 2018: Paper accepted at describing the research on human pose features for action classification[21];
- ECCV 2020: Paper submitted describing the fovea actor centric filtering and ablation studies on actor centric filtering;

- Paper describing the research on semantic neighbor context applications for action detection, currently under preparation.

## 1.5 Thesis Outline

In Chapter 2 the relevant previous work approaches to similar problems are described. Approaches for Action Recognition and Detection in Video, both based in handcrafted features and deep learning, are explained and related to the proposed approach. Also described are the most important works in the field of Multi-Label Sequential Classification with Embedded inputs, and related with our Semantic Neighbor Context approach.

In Chapter 3, the context sources proposed in this thesis - Appearance Context and Semantic Neighbor Context are explained. Also explained are alternative approaches and their limitations, especially when used with CNN-based classifiers, along with their artifact insertions.

In Chapter 4 the challenges addressed by the proposed Multisource Video Classification (MVC) framework regarding context selection, testing and fusion in the action detection task. We then describe how our MVC framework approaches these challenges in a novel way.

In Chapter 5, we describe the test bed used for evaluating the various branches of the MVC framework, as well as the experiments ran, evaluation procedures, and metrics used.

In Chapter 6, we describe the results obtained by the MVC framework, along with the baselines established, performance expectation, and its analysis.

This thesis is concluded in Chapter 7, discussing the insights to be taken from this work. Finally, some future work considerations are provided.

# Chapter 2

# Previous work

The work presented in this thesis relates to two seemingly unrelated fields of machine learning: action detection and recognition in video and sequential multi-label classification. This work describes how to use sequential classification techniques to enhance action detection and recognition performance. As such, the relevant related work on both fields will be presented here.

We separate the approaches to action detection and recognition in video in two categories: those based on handcrafted features, and data-driven approaches. After the performance increase achieved by CNNs in the domain of image recognition[16], it was believed that the advances in performance would also show improvement in action recognition[22]. However, many handcrafted methods still outperformed CNN-based methods regardless of the growing interest in the use of convolutions for action recognition over temporal stacks[23]. Many network-based approaches[18,2] still use an averaging with handcrafted feature based predictions in order to improve performance. Early network-based approaches[24] tried using single 2D CNNs and their results were worse than traditional methods for two reasons: the learned features did not capture either motion and long temporal context and existing datasets (*e.g.* UCF101[15]) were not large or varied enough for the

networks to learn such features.

The work presented in this thesis also deals with sequential multi-label classification. The most common approaches to this problem follow a similar workflow: first define a vector representation for the input[25,26], then use this representation to train an RNN[27,28]. More recent approaches[29,30] found a new way to tackle this problem by using attention based models and circumventing the challenging training procedure imposed by RNN architectures.

## 2.1 Action Recognition in Video - Handcrafted Features

Classification approaches that rely on handcrafted features typically follow the same blueprint: design (usually interpretable) features that describe a region of the input video. Such features are calculated/extracted from the raw video, and linked across space and time, in order to encompass all the information in the video. Then, these features are encoded into a fixed length feature vector that will be fed to a classifier, typically, Support Vector Machines (SVM) or Naïve Bayes.

Csurka *et al.*[31] define a collection of relevant image patterns to use as features, and aggregate their occurrence into a fixed length vector using a histogram to form a bag of keypoints. These bags are then used as descriptors for the whole video, and classified using SVM or Naïve Bayes.

Wand and Schmid[32] estimates camera motion from the video and uses it to correct dense trajectories extracted from the video. These trajectories are then processed (dimensionality reduction with Principal Component Analysis (PCA), orientation correction), aggregated into histograms and then classified. Once again, classification is done with SVMs.

Sánchez *et al.*[33], Perronnin *et al.*[34], and Jégou *et al.*[35] show approaches that aggregate features using the Fisher Kernel method to create a description of the video. Scale-invariant feature transform (SIFT) descriptors are collected to describe the image, and ag-

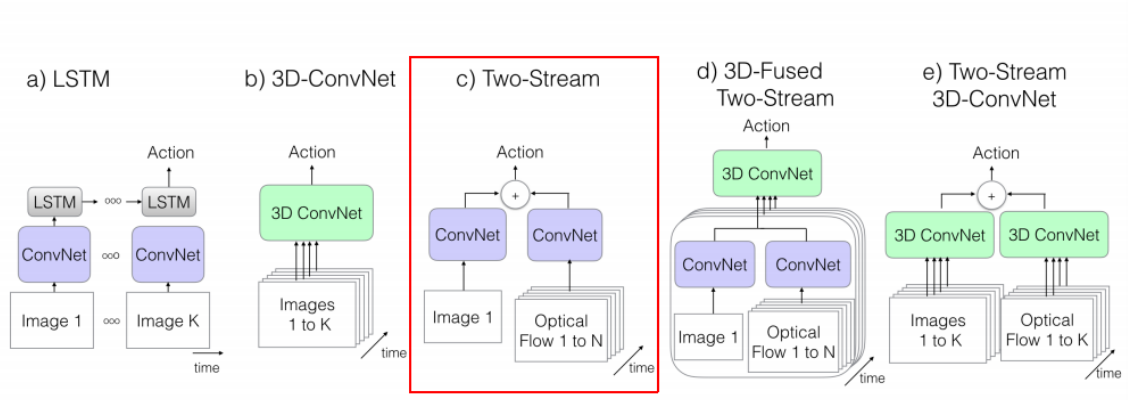



Figure 2.1: Examples of typical action recognition approaches with CNNs as shown in[2]. Approaches a) and b) tend to lack explicit motion features like Optical Flow while approaches d) and e) tend to be computationally intensive. The Two-Stream approach presents an interesting trade-off of providing competitive results with the state-of-the-art 3D-CNN approaches while being much less computationally demanding. See Sec. 2 for more details. Image taken from[2] with permission.

gregated using the Fisher Kernel method to create the Fisher Vector. This vector is then used as a proxy for the video and classified using an SVM.

## 2.2 Action Recognition and Detection in Video - Deep Learning Approaches

After data-driven approaches became more powerful, the use of CNN architectures in action recognition quickly became the state of the art[18]. More recent action recognition approaches differ in how they employ temporal information[2]. We describe five main categories of approaches in this sense: LSTM models[36,37], 3D-ConvNet models[18], Two-Stream architectures[20], 3D-Fused Two-Stream architectures[38] and Two-Stream 3D-ConvNet Architectures[2].

We explain the idea of the approaches in Fig. 2.1: LSTM networks ( approach group a) in Fig. 2.1), (*e.g* Beyond Short Snippets[36], LRCN[37]) model sequences[39] of trained feature-

maps extracted with CNNs to capture long term temporal information. These approaches have the largest temporal receptive fields.

Second,3D-CNNs ( approach group b) in Fig. 2.1), (*e.g* C3D[18,19]) built on earlier single-stream 2D approaches[24], but instead used 3D convolutions on video volumes.

Two-Stream CNN( approach group c)( in Fig. 2.1), (*e.g* Two-Stream[20], Two-Stream Fusion[40], TSN[41]) introduce the use of an additional 2D-CNN stream trained on explicit motion features like optical flow to encode short temporal context. Many aspects of this original approach have been further augmented (*e.g* ActionVLAD[42], HiddenTwoStream[43]). This approach presents an interesting trade-off: it is capable of achieving competitive results with methods requiring far more computational resources like 3D-CNNs.

Finally, Fused Two-Streams, and Two-Stream 3D-ConvNet,( approach group d) and e) respectively in Fig. 2.1) (*e.g* I3D[2], S3D[38], R(2+1)D[44]) are currently the best performing approaches. They extend two-stream approaches in the time dimension as well, either through 3-D convolutions or by alternating between spatial 2-D convolutions with 1-D temporal convolutions. As such, these approaches tend to be more computationally intense than all others.

Two fundamental insights to be extracted when looking at these approaches is that allowing the network to learn temporal features is fundamental for good performance in video classification; and that incorporating explicit motion features like optical flow is beneficial even in 3-D CNNs.

The best methods[45,46,47,48,49,50] submitted to the AVA[3] challenge (currently the most challenging multi-label multi-person spatiotemporal action detection dataset) are showing that the best performing approaches all use prohibitively costly models and training procedures. All of these methods pretrain their models not only on AVA[3], but also on Kinetics[2], an extremely large dataset, made up of over 300,000 video clips. The pre-training procedures can last over 200,000 epochs, before even starting the fine tuning for the actual

task. Also, the complexity of the models keeps increasing (*p.e.*[45] combines seven models of $\approx 33M$ parameters). Dealing with this task without requiring such prohibitive computational power is a necessity.

## 2.3 Embedded Input Multi-label Sequential Classification

In this work, action classification/detection approaches are enhanced by using embedded input sequential classification techniques. This is a fairly niche problem, and, outside of Natural language processing (NLP) not often studied. In NLP, it has several applications like question answering, language translation, and language understanding[30].

Classical approaches to this task can be summarized in the following steps:

1. Design a vector representation for words/characters, called an embedding. This can be done in several different ways like, word frequency counting[51], or neural network based approaches[52][25][26]. Although there are variations, typically these word representations will have the nice property that words that are similar in meaning should be close (in Cartesian distance) in the vector space representation[25].

2. Use this word embedding to train a bi-directional RNN for the target task[27][28][53][54].

Vaswani *et al.*[29] in his 2017 paper revolutionized the way the sequence translation task was tackled by proposing the Transformer model, that departs from the recurrent networks school of thought, to using simple feed-forward networks with self-attention modules. This architecture allows the network to identify which parts of the input sequence should correspond to each module of the output sequence via its multi-head attention layers. These layers allow the correspondence between output tokens and input tokens to be established more directly than their RNN counterparts, since the n-th output token does not have to backpropagate its information query through a chain of hidden state variables.

Instead, the n-th output token provides the attention layers with queries regarding the input, making the lookup much simpler.

Devlin *et al.*[30] then presented the world with BERT, a novel architecture that beat the state-of-the-art in almost every NLP task. Their innovations mostly come from a new training paradigm, where instead of training the model to intake the sequence left-to-right (*e.g.* an LSTM), right-to-left (*e.g.* an LSTM), or left-to-right combined with right-to-left (*e.g.* a bi-directional LSTM), the training is done on the whole sentence at once, essentially making it non-directional. By inputting the whole sentence at once, and randomly masking words in the sentence to either be a word missing token, a random word, or the correct word, and then demanding that their classifier predicts the correct word. This is done in a self-supervised fashion, where the training corpus is comprised of complete sentences that are then altered. This novel architecture was based on the Transformer[29] model, and uses its self-attention modules.

# Chapter 3

# Appearance and Semantic Context

This thesis focuses on leveraging extra sources of information (named context sources) for multi-person, multi-label, spatiotemporal action detection and recognition in video.

Action Recognition is a task with the goal of matching a video depicting a person performing an action with one label, describing that action. Given the same video, the goal of action detection would be to determine where (which pixels in the video?) and when (in which frames is the action being performed) the action happens, without the requirement of actually knowing which action is being performed. Action detection and recognition ties both tasks by aiming to know when, where and what is being shown in the video. With more than one person (multi-person) per video the task becomes more complex, since now it is required to also know more than one action, and the segmentations in both space and time associated with those actions. If each person can be depicting multiple actions in a video (simultaneously or not), the problem becomes multi-label.

So, for the multi-person, multi-label, spatiotemporal action detection and recognition task in video, the goal is to receive as input a video, and to return a list of actions coupled with bounding boxes in both space and time, depicting every action of every person depicted in the video.

This is a very challenging problem, where even the most complex and sophisticated models (*e.g.*.[32]) are showing modest performance (state-of-the-art currently achieves $34.25$ mAP in the AVA[3] dataset). As described in 2, the best performing approaches all use prohibitively costly models and training procedures with pre-training procedures lasting over 200,000 epochs and very complex models (*p.e.*[45] combines seven models of $\approx 33M$ parameters). Dealing with this task without requiring such prohibitive computational power is a necessity. In this thesis, I advocate leveraging context to boost performance without the computational overhead required by the state-of-the-art.

However, caution must be employed when using context sources. **If a context source's predictive power is low, the overall performance of the system using it might decrease**. Also, even if the source has some predictive power, the **performance increase/computational cost trade-off** must also be taken into consideration.

To address this problem, in this work we describe an architecture we denote Multi-source Video Classification Framework, enabling the testing of context sources, independently and in combination, in a flexible, modular and extensible fashion.

## 3.1 Actor Centric Filtering

Actor Centric filtering contributes to improve classification performance by using more detailed information in the location of the target than in the surrounding background. As such, the features correlated to the action performed by the target person will have a stronger role in the classification than the "distracting features" of the other actors. This is particularly useful in cases where different actors performed different actions at the same time (*i.e.* in the same frame), as is the case of the AVA dataset. Two alternative approaches are possible (see Fig. 3.1=: the first, named Crop filter, would be to crop the area outside that region and the second, named GBB (Gaussian Background Blur) would be to simply apply a Gaussian blur with a Gaussian kernel to everything outside that region. The Crop

filter approach is in effect what is happening when a network following the architecture defined by Faster R-CNN[55] is used, since the feature maps passed to the classifier layers of the network are cropped around the RoI chosen by the region proposal network.

Despite the goal of the actor centric filters, the Crop filter, having no background context at all, might hinder the classifier (i.e an abundance of blue might help a classifier guess the *swim* action). Additionally, for both the GBB and the Crop filter, regions with large contrasts exist (i.e in the crop filter there is a sharp transition from the actor centric region to black and in the GBB there is a sharp transition from the relevant region to a blurred region).

To overcome these artifact insertion limitations we propose an artificial foveal vision filter from[56] inspired by human foveal vision[57][58]. This provides a smooth blur transition between the bounding box and the background (See Fig. 3.1). Firstly, a Gaussian pyramid is built with increasing levels of blur. The image $g_{k+1}$ can be obtained from $g_k$ via convolution with 2D isotopic Gaussian filter kernels with progressively higher $\sigma_k = 2^{k-1}\sigma_1$ standard deviations for each $k$th level. Next, a Laplacian pyramid[58] is computed from the difference between adjacent Gaussian levels. Finally, exponential weighted kernels are multiplied by each level of the Laplacian pyramid to emulate a smooth fovea:

$$k(u, v, f_{kx}, f_{ky}) = e^{-\left(\frac{(u-u_0)^2}{2f_{kx}^2} + \frac{(v-v_0)^2}{2f_{ky}^2}\right)}, \qquad 0 \leq k \leq K \tag{3.1}$$

where $f_{0x} = \frac{1}{2}w$, $f_{0y} = \frac{1}{2}h$, and $f_{kx} = 2^k f_{0x}$, $f_{ky} = 2^k f_{0y}$ is used to define the fovea intensity at the $k$-th level and the fovea is centered at $u_o = x + w/2$, $v_o = y + h/2$, given $(x, y, h, w)$ where $x, y$ are top-left corner coordinates and $h, w$ are the height and width of the bounding box.

In Figure 3.1 an example of the actor centric filters used in this work is shown, highlighting the difference in the abrupt transition boundaries introduced by the approaches

of GBB and Crop filtering versus the smooth transition in the Fovea filter.

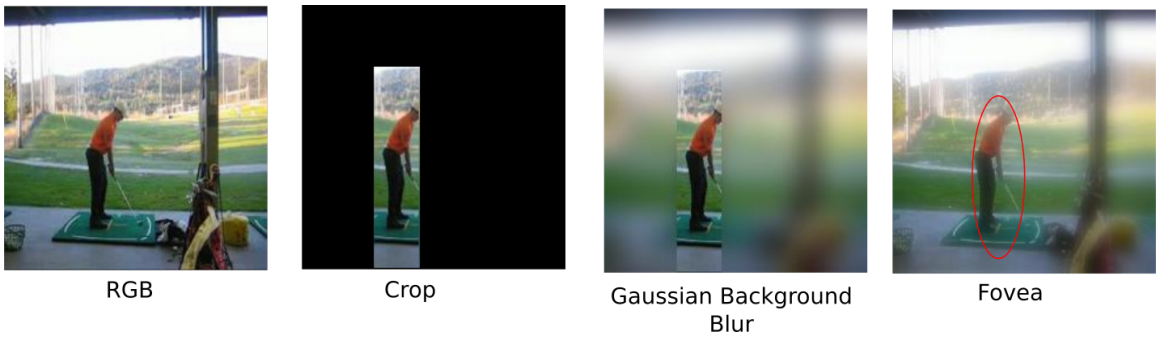



Figure 3.1: Comparison of the different actor centric filters. We highlight how the Fovea filter induces less artifacts in the image when compared with GBB and Crop filtering.

## 3.2 Semantic Neighbor Context

The multi-label characteristic of the action detection task greatly increased the difficulty faced by classifiers. However, it also allows for new classification approaches to surface. Using human interactions as inspiration, We propose the following classification methodology: create an encoding describing the actions of neighbors of the target across space and time to predict the actions of the target. Using an example: imagine two people talking. At any point in time one person will be talking while the other person will be listening. Suppose the target is currently *talking*. Solely from the information that the target's neighbor is *listening* it should be reasonable to predict that the target is currently talking. This is a simple example of a binary example, fairly contained in time. It is also an intuitive relationship between actions. But what about actions that negatively affect one another? For example, if the target's neighbor is swimming, it is unlikely that the target is sleeping or sitting down. So there also exist "negative" relationships between actions. Other possibilities would be groups of more than two actions, or actions that precede or succeed one another. The ideal scenario would be to simply aggregate as many action groups as

possible, correctly ranking them according to space and time in relation to the target person/frame combo, and create a classifier that infers all these relationships automatically.

With this in mind, we propose the following encoding: let $T$ be the number of timesteps considered by the encoder (for the target timestep $t_0$ we encode the neighbor actions from $t_{-T}$ to $t_T$). This creates a time window of length $2T+1$. Let $N$ be the number of neighbors to be considered. For each timestep, the closest $N$ neighbors will have their actions encoded. The distance between the target and a neighbor is calculated by the euclidean distance between the centers of their corresponding bounding boxes. Finally, let C be the number of classes considered by the encoding. In that case, given a target person and a timestep, our encoding creates a vector of size $(2T + 1) \times N \times C$.

We process the data resulting from this encoding with an LSTM network designed and trained specifically for this problem. The two best performing LSTM architectures are shown in Fig. 3.2. Model A has two LSTM layers at the input, one receives a sequence of the past timesteps and the present timestep, and the other receives the present timestep and the future timesteps. These two LSTM layers are fused into another LSTM that encodes the full sequence that is then finally used as an encoding for classification. The LSTM model B has the same pair of LSTM layers at the input but fuses them immediately into a fully connected layer, without resorting to an intermediate LSTM to process the whole sequence.

This type of problem is called an Embedded Input Multi-label Sequential Classification. This is a fairly niche problem, and only found focus in the NLP community. In Section 2.3, I describe the various approach methodologies followed in recent years. Although the current approach to beat is BERT[30], it is not possible to apply it to this specific task. The reason for this limitation is that BERT's architecture is based on the self attention modules of Transformer[29]. These modules leverage the unique properties of Language Model Encodings in which words that are close semantically are also close in the vector space in

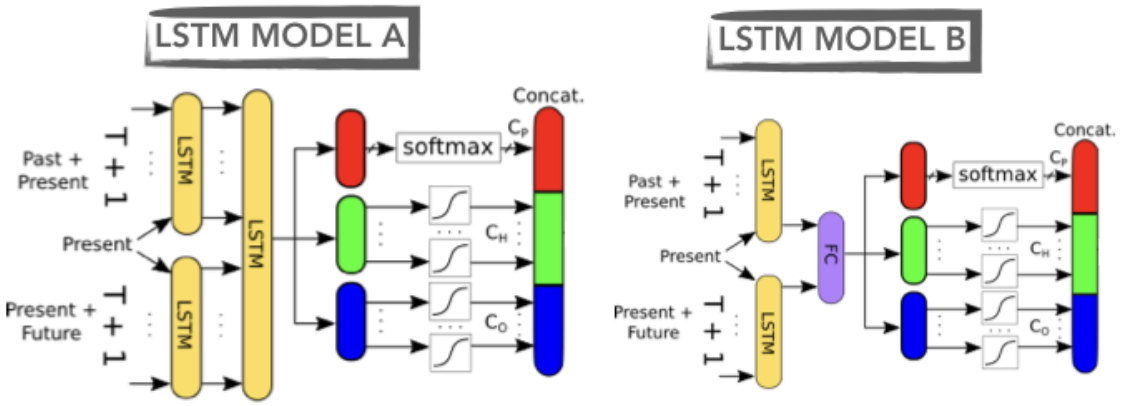



Figure 3.2: Our two best LSTM models for semantic neighbor context classification. While model A encodes the input sequence fully into a 2-layer LSTM, model B fuses the past+present and present+future sequences by concatenation into a fully connected layer.

which they are encoded. Defining the "semantic closeness" of actions to classify is part of the task. Since our encoding ranked the actions based on neigbhor-target distance (both spatial and temporal) and not semantic distance, we are unable to use the same attention modules.

This is why the bidirectional LSTM approach was used instead.

## 3.3 Skeletal Context Source

Action classification resorting to skeletal data is a common approach. However, using skeletal data as a sequence to perform action detection and recognition is more challenging, mainly because of the pre-processing involved in dealing with incomplete skeletons, joint tracking over time and space, and the ability to deal with a more complex label space (1 label per frame vs 1 label per video).

In the work published at BMVC 2018[21], we present ablation tests determining optimal network architectures and testing choices for action detection and action recognition on the

AHA[21] dataset. The AHA dataset is a curated dataset of 21 subjects performing exercises designed to promote active ageing, including both RGB and skeletal data. This dataset is published in that paper and is available online. In the experiments described, incomplete skeletons are dealt with by approximating its position from skeletons on other frames. Tests are also run showing classification done with both joint positions as well as velocities and accelerations. For recognition, we trained models on the positions of the joints achieving 88.2% $\pm$ 0.077 accuracy, and on joint positions and velocities, achieving 91% $\pm$ 0.082 accuracy. The segmentation baseline achieved an accuracy of 88.29% in detecting actions at frame level. Although this approach has not yet been applied to the datasets contemplated in the MVC framework this is one of the main approaches we intend to research in future work.

## 3.4 Extension to Object Detection Context Source

Using the information pertaining to objects in the scene to aid in the action classification context has been studied before, although infrequently. Han *et al.*[14] present one such approach. The intuition behind this is that certain classes of actions can be classified by the existence of objects and their spatial relationship with the target. For example, classifying that the target is *holding an object* can be achieved by determining that an object is present very close or inside the target's own bounding box (even in cases where the label is object agnostic, as is the case in the AVA dataset).

We have not done research in this topic in the past. The approach to take for the initial research on this is to use an off-the-shelf object detector, encode the object type and position into a vector (similar to the approach taken in the semantic neighbor context, see Sec. 3.2) and design a network to classify the action the target is performing taken as input the position and type of the $N$ nearest objects to him, in space and time.

# Chapter 4

# MVC Framework

In this chapter, the proposed context source testing and fusion frame work is described, the Multisource Video Classification Framework. The datasets used for testing are also described, as well as the reasons for choosing those datasets. Finally, the proposed Hybrid Sigmoid-Softmax loss is explained to deal with multi-label scenarios where a subset of the labels is mutually exclusive.

## 4.1 Multisource Video Classification (MVC) Framework

In this Section, the Multisource Video Classification (MVC) framework that enables context source fusion and testing is described. Leveraging information from extra sources to improve the performance of action detection and recognition in multi-person, multi-label, spatiotemporal action detection and recognition is a promising approach, as shown in this thesis. However, it is of crucial importance to be able to gauge the performance of a context source. If the source has no predictive power, the overall performance of the classification system might decrease. To be able to discern the quality of a context source, our MVC framework needs to fulfill the following requirements:

- Flexible - It should allow contributions of any context source to the overall performance of the system to be tested, either independently or in any combination, with as little re-training as possible.
- Extensible - It should be possible to add new context sources without re-structuring. The new sources should also take advantage of the flexibility properties that the pre-existing context sources have.
- Modular - Each context source should be able to be added or removed independently from the others.

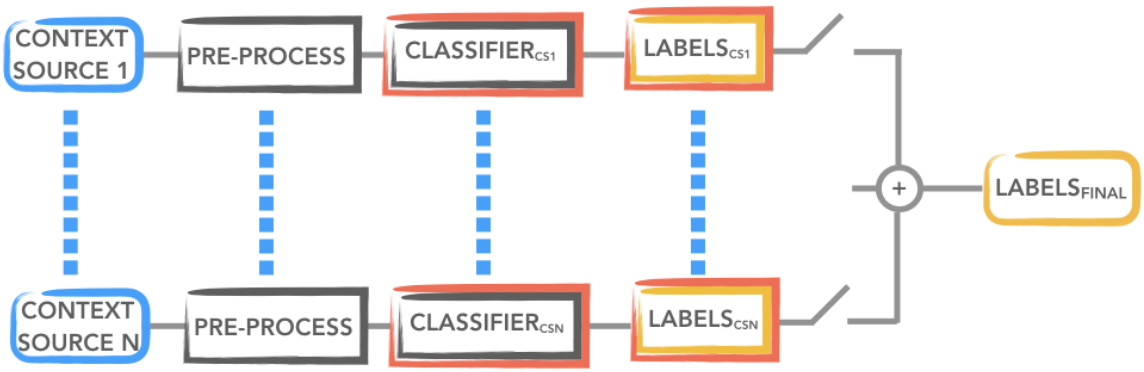



Figure 4.1: The format followed by the proposed MVC framework. Each context source is pre-processed and classified individually. Each set of labels obtained from each each source can be included or excluded from the final fusion step in order to assess its contribution to the overall classification score. The color code on the boxes is blue for input data, black for classifiers, red for original contributions, and yellow for labels.

In Fig. 4.1 the format followed by the MVC framework is shown. For each context source, independent pre-processing techniques are applied, and the processed data is fed to a classifier that takes only that context source into account. The labels obtained from that context source are then fused with any number of context sources to achieve the final labels. The fusion is achieved via weighted averaging of prediction scores of each context source.

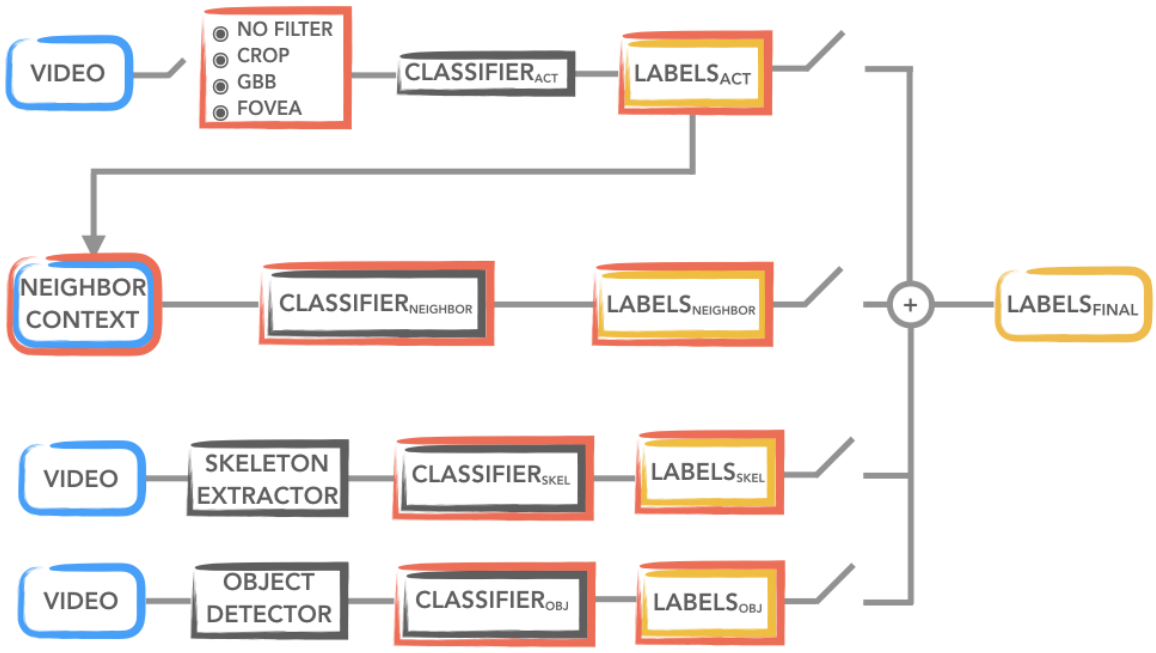



Figure 4.2: The context sources considered in the MVC framework: actor centric filtered RGB and OF frames extracted from the input video, semantic neighbor context information, in which a classifier attempts to predict the actions of the target based on the actions of his neighbors (in space and time), skeletal information, and objects. The color code on the boxes is blue for input data, black for classifiers, red for original contributions, and yellow for labels.

In Fig. 4.2 the specific context sources considered in the proposed MVC framework are exemplified. The four context sources are: actor centric filtered videos, semantic neighbor context data, human pose features and object detection. The actor centric context source is handled by filtering the RGB and Optical Flow (OF) images extracted from the video using one of four possible filters: no filter, cropping the images around the target actor, GBB - blurring the background with a gaussian kernel, and fovea filtering, a smooth filtering function that does not introduce edges/artifacts into the image (more info in Section 3.1). This context source enhances the classification by establishing a hierarchy in the video pixels used for input. In a video with two people performing two different sets of actions the classification task becomes impossible if no hierarchy is established. For example, in a

video where two people are visible, one person is standing, while the other one is sitting. The pixels depicting the person standing are not helpful in the classification step tasked with determining the actions of the person sitting. The appearance filters direct the classification to the right person without discarding the informantion that may be leveraged from the background. Classification is done using a deep two-stream CNN (see Fig. 2.1 and Section 4.3.4) with weights available and pre-trained on UCF101. This approach provides a good balance between classification performance and computational cost, by leveraging explicit temporal information in the OF stream, but not requiring 3-D convolutions like 3D-CNN based models.

For the semantic neighbor context source, the input is generated from the predicted labels for the neighbors of the target actor and classified with an LSTM developed for this purpose (more information on this on Section 3.2). The third context source is the skeletal information extracted from the target actor performing the action. This information can be obtained either during the data collection task, for example if a Kinect was used, or extracted from the 2D video using a tool like OpenPose[59,60,61,62]. The classifier for this is also trained and created specifically for this task[21]. The fourth context source are the objects detected in the image. This part is not yet developed, but the approach to be used is to employ an off-the-shelf object detector to create a vector depicting the objects detected and their positions/sizes and then create a classifier that attempts to predict the target's actions. Finally, the context sources can be fused or used individually as shown by the "+" signal on the right side of the diagram in conjunction with the switches found at the end of each context source line.

## 4.2 Combining the Context Sources

One important limitation of this context source is that it requires knowing the actions of the neighbors. It is important to be able to improve classification even when ground truth

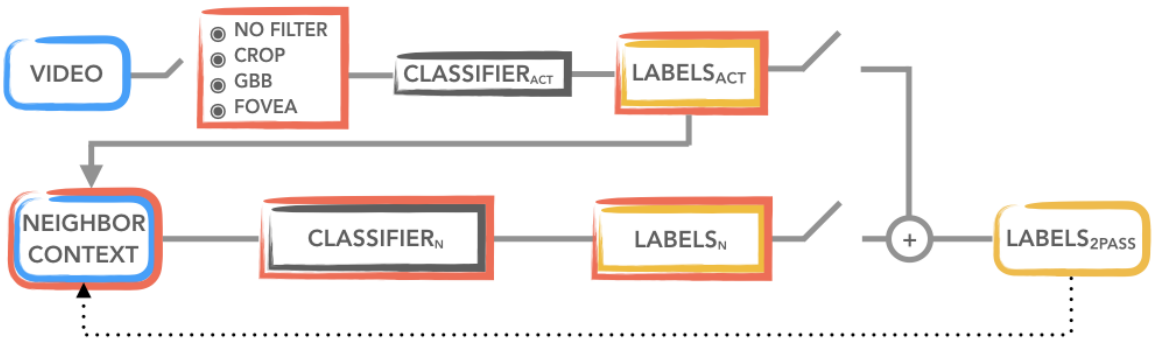



Figure 4.3: Example of how the two-pass scheme works. From the actor centric filtering method, we get $Labels_{Act}$. From these labels we generate the semantic neighbor context and feed that encoding to the classifier trained for that encoding, getting $Labels_N$. Fusing $Labels_{Act}$ with $Labels_N$ we get the labels from a two pass scheme. If we take those $Labels_{2pass}$ and generate a new semantic neighbor context encoding (as suggested by the dotted line) we get a three pass scheme. We later show that this iterative method improves performance.

information of the neighbor actions is not available. As will be shown in the results section (see Sec. 6), using a different classifier, *p.e.:* our actor centric filtering approach, to generate neighbor semantic context data, and then using that data as input, improves performance. We show this idea in Fig. 4.3. We call this the two pass approach (one pass through the actor centric filtering network, and one pass through the semantic context network approach and fusion of both context sources). We also experiment taking the labels generated from the two-pass approach and using those as input to the semantic neighbor context, and then fuse them again. We call this the three-pass approach, and show it provides (minimal) improvement over the two-pass approach. We also show what results would be obtained if the ground truth labels were used to generate the semantic neighbor context input.

## 4.3 The AVA dataset and the UCF101-24 dataset

The AVA dataset is currently the most challenging dataset in the Action Detection task. It is the dataset used year after year in the CVPR action detection challenge[3] and the per-

formance obtained by the state-of-the-art methods reveals that the video understanding problem is not yet solved when dealing with more complex data (see Section 2). The complexity that is achieved by this dataset will be explained shortly. However, this dataset is prohibitively large, and doing testing of the models proposed in this work on the full dataset was not possible due to computational restraints, therefore, we used miniAVA (See Section 5.1), a subset of the AVA dataset. miniAVA retains important properties of the AVA dataset (class imbalance, sequential actions) while being smaller in size, allowing us to run tests on it. The dataset that used to be the state-of-the-art before AVA was released, the UCF101-24 is also used for testing, since this dataset is more manageable in size. This allows for a fair comparison of the proposed models and the state-of-the-art.

### 4.3.1 AVA - Atomic Visual Actions

The AVA dataset is composed of densely annotated 15-minute videos, sources from movie scenes. The annotations are person-centric with a sampling frequency of 1Hz, and each annotation corresponds to a 3-second video snippet, with the label corresponding to the frame in the center, called the keyframe, of that snippet.

The dataset has a unique multi-label structure as shown in Fig. 4.4. For each person, in each frame, a bounding box is provided with one to seven labels. Each bounding box in a frame must have exactly one pose label like *stand* or *walk* (from mu-


Figure 4.4: Examples of AVA[3] labeling. *HO* stands for human-object actions and *HH* stands for human-human actions.

tually exclusive $C_P$ of them), zero-to-three human-human labels (from non-mutually exclusive $C_H$ of them) like *talk to* or *hit* and zero-to-three human-object labels (from non-mutually exclusive $C_O$ of them) like *hold object* or *watch (e.g TV)*. The total number of classes is $C = C_P + C_H + C_O$.

In AVA, the action class distribution is heavily imbalanced, roughly following Zipf's Law (see[3] for more details.) This imbalance itself is a challenge given that some classes will be severely under-represented (the most prevalent classes have over $10^5$ instances while the least common have less than 20).

Finally, to give the reader an approximate notion of how challenging this dataset is, we highlight the performance that the baseline model provided in[3] achieved, with this dataset. The baseline model is a very deep, 3D-CNN that is a fusion of I3D[2] and Faster RCNN[55]. The performance this model obtained in JHMDB[63] and UCF101-24[15] beats other state-of-the-art methods[3], achieving 78.6% and 59.9% video mAP at 0.5 IoU, respectively. In the AVA dataset, this model scored 12.3% video mAP at 0.5 IoU.

### 4.3.2 UCF101-24

The UCF101-24 dataset is a sports based video dataset for action recognition and detection. For each input video there is exactly one label identifying the action that is represented in it. When there are multiple people in the video, several bounding boxes are provided, but the label is the same for all the bounding boxes. As such, to use the architecture shown in Fig. 4.5 on this dataset we change the output layers to a Softmax layer (since every class will be mutually exclusive with all other classes). Note that since the label is the same for every person in the same video the semantic neighbor context source can't be applied here since the problem would be trivial: the classifier learns to always predict as output the labels it receives as input.

### 4.3.3 Hybrid Sigmoid-Softmax Loss

The baseline method for AVA[3] has an output layer of $C$ (number of classes) independent sigmoid activation functions. Note, however, that this does not truly reflect the label structure of the AVA[3] dataset which is not entirely mutually exclusive but also not entirely multi-label. In the AVA dataset there are three distinct types of classes, each describing a type of action: pose classes (*e.g.* standing), human-human interaction classes (*e.g.* hug), and human-object interaction classes (*e.g.* hold (an object)). While the pose classes are mutually exclusive (*i.e.* each data sample has exactly one pose class in its label), the human-object and human-human interaction classes are not mutually exclusive, and any number of zero to three of each type can be found in the label of each data sample.

As such, in the MVC architecture, we use a custom loss function: Hybrid Sigmoid-Softmax Loss, to address this heterogeneity by having three separate output layers. The layer corresponding to the pose classes has size $C_P$ with a Softmax activation function. The remaining two layers have a sum-of-sigmoids activation function, where the number of sigmoids is $C_H$ for the human-human interaction classes and $C_O$ for the human-object interaction classes.

This implies that the loss function cannot simply be categorical cross-entropy or binary cross-entropy[64]. Therefore the architecture must minimize a global loss which is the sum of the losses of each output layer and which can be expressed as follows:

$$\mathcal{L}_{GB} = \overbrace{-\log\left(\frac{e^{s_p}}{\sum_j^{C_P} e^{s_j}}\right)}^{pose\ classes\ loss} + \overbrace{\sum_j^{C_H}\left(-\sum_{i=0}^{1} t_{j,i}\log(\sigma(s_{j,i}))\right)}^{human-human\ classes\ loss} + \overbrace{\sum_j^{C_O}\left(-\sum_{i=0}^{1} t_{j,i}\log(\sigma(s_{j,i}))\right)}^{human-object\ classes\ loss} \quad (4.1)$$

where the $s$ vectors are the predictions of each layer in equation 4.1 (different for each component of the sum), $s_p$ is the output corresponding to the only positive label in the softmax layer, and $\sigma$ is the sigmoid function, $j$ is iterating over the classes, $i$ is iterating

over the possible labels for each class (1 or 0) and $t$ is the ground truth class. Note that this backpropagation is possible as the encoding of the target labels is done as a binary vector which can then be partitioned into smaller vectors for each output layer. A representation of how this loss is incorporated in the used CNN architecture is shown in Fig. 4.5.

### 4.3.4 Two-Stream CNN

We propose a baseline two-stream CNN architecture motivated by promising results achieved with these networks[40,65], as shown in Fig. 4.5. We use a ResNet50 architecture for each stream because this architecture demonstrated to be successful[66] for these tasks, as well as having available pre-trained weights from[40] trained on UCF101[15]. The input is a single frame for the RGB stream and an optical flow stack for the OF stream. The output layers take into account the unique labelling structure of the AVA[3] dataset where for the pose classes ($C_P$) a Softmax is used, enforcing mutual exclusion between labels. For the human-human interaction ($C_H$) and human-object ($C_O$) interaction classes, a sum of sigmoids loss function is used, which is then thresholded. The top 3 values that are above the threshold are chosen as classes. Each network stream (RGB and OF) is individually fine-tuned for its task. We fuse these networks using a concatenation fusion of their last Fully Connected (FC) layers and then train a FC layer so that spatiotemporal features can be learned. Since retraining both networks together would be too computationally heavy we load their individually fine-tuned weights, and freeze them (*i.e* do not update them in backpropagation) so we only train the desired FC spatiotemporal filters.

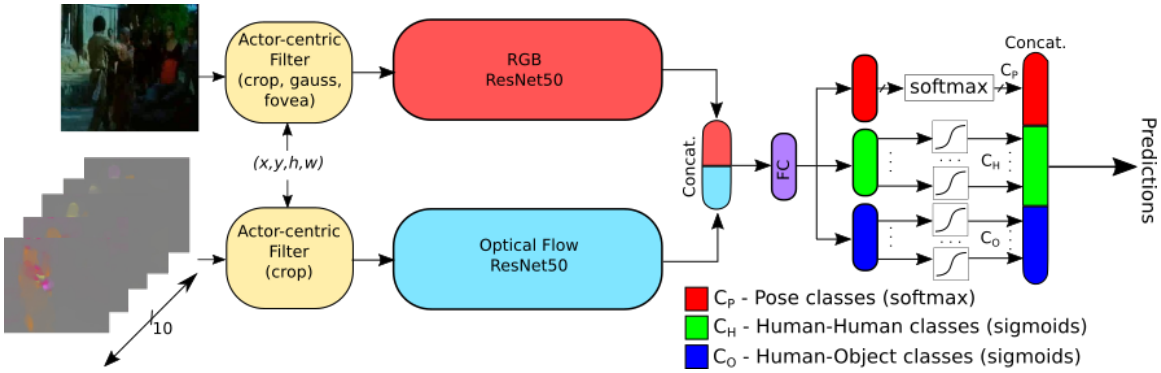



Figure 4.5: The proposed two-stream architecture for the AVA dataset. After training each stream (RGB and OF) we fuse them with concatenation and apply our Hybrid Sigmoid-Softmax Loss on the output.

# Chapter 5

# Experiments

We test our models on two datasets, the UCF101-24 and the AVA dataset. Although we could process the whole UCF101-24 in the computer we had available, due to computational limitations, we were unable to run the tests on the full AVA dataset, so we were forced to create a subset of this dataset. For simplicity, we refer to it as miniAVA.

## 5.1 miniAVA

The AVA dataset was too large to process with the computational resources we had available. In Table 5.1 we show the two datasets used in this work. Note that the AVA dataset is around 17 times larger than the UCF101-24. Furthermore, the multi-label characteristic of AVA makes it more computationally intensive during training.

| Dataset | AVA | UCF101-24 |
|---|---|---|
| #Videos | 430 | 3194 |
| Avg. Video Duration | 15 min | 7 sec |
| #Classes | 80 | 24 |
| Labels per Video | Variable | 1 |
| Total Video Duration | 6450 min | 373 min |

Table 5.1: Comparison of AVA and UCF101-24

When partitioning AVA we had three goals: make it computationally tractable with our available resources, maintain temporal continuity in the samples, and maintain the class distribution of the original dataset (see[3] for details). We ensured temporal continuity by sampling from the first segments from each split of the AVA dataset. We were also able to maintain a distribution that roughly follows Zipf's law, albeit with a smaller number of classes, as shown in Table 5.3. This ensured we would not simplify the dataset by reducing class imbalance inadvertently. Similarly to the testing scheme in the ActivityNet Task B challenge[67] involving AVA, we took particular care to make sure that all classes had at least 20 samples in the test set. we did this by removing the class labels that had less than 20 examples on the test set. The number of samples chosen for each set was 15000. We chose this number so that We could process the dataset in a reasonable time frame. The miniAVA dataset then has 30 classes remaining. A table with all the classes is presented in Table 5.1. The experiments ran on the appearance context alone took around three months of non-stop processing. The computer used to run these tests was equipped with two NVIDIA 1080 TI GPU and 64 GB of RAM.

## 5.2 Subsampling and Voting Scheme for AVA

As discussed in Sec. 4.3, the AVA dataset annotations are person-centric and made at a 1Hz frequency. Each annotation corresponds to a three second video segment. A straightforward implementation, at a rate of 30 FPS for each three second segment, would need a receptive field of 90 frames and a 3D architecture to process a single segment. This would imply a much more computationally intense architecture and, as such, we propose a subsampling scheme as shown in Fig. 5.1.

Since WE have five frames and OF volumes representative of a segment, I use a majority voting scheme to arrive at a consensus for the label to assign to a segment. A similar strategy was employed in[68].

| Label ID | Label | Label Set |
|---|---|---|
| 1 | Bend/Bow (at the waist) | Pose |
| 2 | Crouch/Kneel | Pose |
| 3 | Dance | Pose |
| 4 | Get up | Pose |
| 5 | Lie/Sleep | Pose |
| 6 | Martial art | Pose |
| 7 | Run/jog | Pose |
| 8 | Sit | Pose |
| 9 | Stand | Pose |
| 10 | Walk | Pose |
| 11 | Answer Phone | Human-Object |
| 12 | Carry/Hold (an object) | Human-Object |
| 13 | Drink | Human-Object |
| 14 | Eat | Human-Object |
| 15 | Push (an object) | Human-Object |
| 16 | Read | Human-Object |
| 17 | Ride (e.g., a bike, a car, a horse) | Human-Object |
| 18 | Sail boat | Human-Object |
| 19 | Smoke | Human-Object |
| 20 | Text on/Look at a cellphone | Human-Object |
| 21 | Touch (an object) | Human-Object |
| 22 | Watch (e.g., TV) | Human-Object |
| 23 | Fight/Hit (a person) | Human-Human |
| 24 | Give/Serve (an object) to (a person) | Human-Human |
| 25 | Grab (a person) | Human-Human |
| 26 | Hug (a person) | Human-Human |
| 27 | Listen to (a person) | Human-Human |
| 28 | Sing to (e.g., self, a person, a group) | Human-Human |
| 29 | Talk to (e.g., self, a person, a group) | Human-Human |
| 30 | Watch (a person) | Human-Human |

Table 5.2: Description of the Labels present in miniAVA.

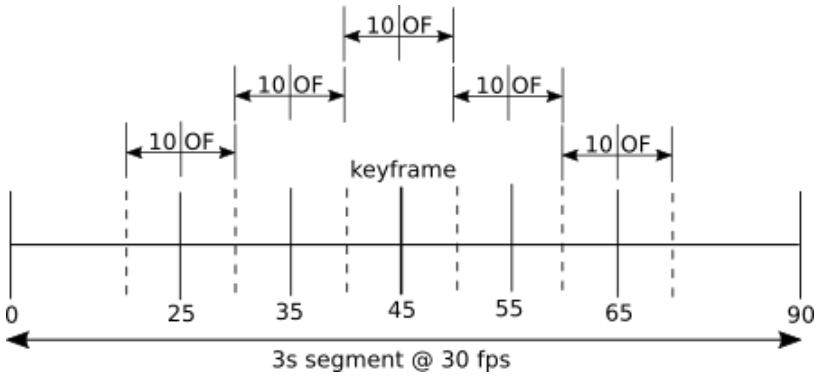



Figure 5.1: Our subsampling scheme. We extract 5 representative RGB frames and 5 OF stacks around a keyframe.

As such, to obtain the predictions for a single segment we pass each frame of the five subsampled representative frames and its corresponding OF through the network and store the predictions. For the mutually exclusive (pose classes) predictions we count the maximum valued prediction as a vote and for all non-mutually exclusive (human-human and human-object classes) predictions I count any prediction values above a certain threshold $v$ as a valid vote. This threshold was initialized at 0.4, following[3]. Then, for the mutually exclusive classes We take the most voted class as the predicted class for the segment, and for each of the non mutually exclusive classes We take their top 3 most voted classes as the predicted classes for the segment. For the non-mutually exclusive classes the number of predicted classes will be between 0 (if no prediction is above the threshold) and 3.

| Class Category | miniAVA | AVA | Mutually Exclusive |
|---|---|---|---|
| Pose | 10 | 14 | Yes |
| Human-Object | 12 | 49 | No |
| Human-Human | 8 | 17 | No |

Table 5.3: Class categories in the AVA[3] dataset vs miniAVA.

## 5.3 Bounding Boxes

Our approach is focused on the action classification (as discussed in Section 3), leaving actor localization in time and space to other works. As such, we use the ground truth bounding boxes provided with the datasets. We argue that using ground truth information for actor localization doesn't reduce the difficulty of the AVA, referring to an ablation study done in[3]. In Table 5 of[3] (Also shown as Table 1.1 in this thesis), the authors report that comparing UCF101-24 and AVA, the performance decrease in actor detection (*i.e.*: locating the person) is quite small (84.8% to 75.3% frame-mAP) when compared to the performance dip in action detection (76.3% to 15.6% frame-mAP).

## 5.4 Metrics

Three metrics are used in the tests that were ran for this thesis: Accuracy, Mean Average Precision (mAP), and F1-score. Accuracy does not merit an explanation, but the mAP and F1-score require some more care when interpreting results.

### 5.4.1 mAP - Mean Average Precision

While there are several definitions of mean Average Precision (mAP), the most prevalent one used in Action Detection tasks is the one defined for the PASCAL challenge[69]. This metric hinges on the concepts of precision, recall and Intersection over Union (IoU). These concepts are defined as:

$$Precision = \frac{TP}{TP + FP} \tag{5.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

$$IoU = \frac{A(overlap)}{A(union)} \tag{5.3}$$

where $TP$ are true positives, $TN$ are true negatives, $FP$ are false positives and $FN$ are false negatives and a true positive only exists if its class matches the ground truth and $IoU \geq 0.5$. IoU measures how much overlap exists between two regions, normally a prediction and the groundtruth (in the Action Detection scenario this relates to both the overlap in the spatial domain - bounding boxes - and over the temporal domain - time segments).

Average Precision can then be defined as the area under the curse of the $Precision(Recall)$ function. If we were to plot $Precision(Recall)$, the idea of AP can be conceptually viewed as finding the area under the curve of the $Precision(Recall)$ curve. While that would imply computing the integral of the curve,[69] introduce the approximation of computing the average of precision at 11 recall levels (from 0.0 to 1.0):

$$AP = \frac{1}{11} \sum_{Recall_i} Precision(Recall_i) \tag{5.4}$$

where the value of the precision at 11 recall levels is averaged (from 0.0 to 1.0 in 0.1 increments). Note that the average precision at recall $i$ is taken to be the maximum precision measure at a recall greater or equal than recall $i$. This becomes a single value summarizing the shape of the precision-recall curve.

Averaging the per class AP over all classes, mAP is defined as:

$$mAP = \frac{1}{C} \sum_{i}^{C} AP_i \tag{5.5}$$

where $AP_i$ is the average precision of class i as computed in 5.4.

### 5.4.2 F1-Score

The F1-score is defined as:

$$F_1 = 2 \frac{Precision \times Recall}{Precision + Recall} \tag{5.6}$$

where Precision and Recall are as defined in Section 5.4.1.

In the case where the problem is multi-class or multi-label, the most balanced approach is to calculate a per-class F1-score and then calculating a weighted average of the per-class F1-scores to generate a global metric.

## 5.5 Implementation Details

Following the recommendations of[70] the training was done for 200 epochs. Batch size is 32 with a learning rate of 0.001 decaying to 0.0001 after 80% of the epochs. All streams were ResNet50[71] networks which were initialized with weights from[66] for the respective RGB and OF networks trained on the UCF101-24 dataset. The input to the networks are 224x224 RGB images and stacks of 10 OF volumes. For the results we use the same benchmarking tools as those provided for the AVA challenge with minor alterations only to obtain useful plots.

We report the results of several experiments on miniAVA and on UCF101-24. This task involves localizing the atomic actions in space and time, achieving the highest video mAP (at 0.5 IoU) possible. As described in Section 5.3, the ground truth bounding boxes were used for spatial localization (see Section 5.3 for the justification why this simplification does not trivialize the problem). For all experiments we round mAP to 2 decimal places.

## 5.6 Tests

### 5.6.1 Appearance Context Source Tests

We ran the following tests on our methods, on both the miniAVA dataset and the UCF101-24 datastet, regarding the appearance context source:

- Baseline RGB, OF, and 2-stream Fusion with no Actor Centric Filtering Filtering

- Hybrid Sigmoid-Softmax Loss vs Sum-of-Sigmoids.
- GBB, Crop and Fovea Actor Centric filtering on RGB streams.
- 2-stream Fusion With all combinations of Actor Centric filtering (including no filtering)

### 5.6.2 Semantic Neighbor Context Source Tests

Regarding the semantic neighbor context source, we ran tests exclusively on the miniAVA dataset. It would not make sense to use semantic neighbor context information on the UCF101-24 dataset because on that dataset, in each video, every person depicted is performing the same action, so the actions of the target and the its neighbors are always the same.

The tests ran were:

- Hyperparameter tuning for the LSTM architectures (tuning number of neighbors, time window, and number of units), deciding with LSTM architecture to use
- LSTM fusion with two passes vs three passes (see Sec. 4.2)
- Noise resilience tests
- Baseline testing: LSTM vs Feed-Forward Neural Network (FFNN) vs Naïve-Bayes

For the noise resilience tests we feed noisy data to the LSTM networks to check how the performance changes with noise. To generate this noisy input we take a clean input sample using ground truth data and switch input labels from the semantic neighbor context encoding randomly, at a fixed rate, varying from 0% noise to 100% noise.

The baseline testing is done to evaluate the impact of two different classifier characteristics: sequencing capability and output structuring (See Table 5.4. When using LSTM architectures to train on our semantic neighbor context encoding we are leveraging not

only the sequencing abilities of the architectures to inform each classification step with the internal representation of past classification steps, but also taking advantage of the structured output cost function described in Section 4.3.3.

The Naïve-Bayes model is designed to tackle binary classification problems. The transformation from binary classification problems to multiclass problems is straight-forward - just develop K (where K is the number of classes) binary models, and then compare the results of all models at the classification step. To enhance it further to deal with multi-label problems some work is required. In our application, the problem follows a unique structure of having some sets of labels being mutually exclusive (i.e. multiclass) and some being truly multi label (See Section 4.3.1). To deal with this, we train 30 independent Naïve-Bayes models for the 30 classes present in the miniAVA dataset, and then use post-processing during the labelling step. For the pose labels (the mutually exclusive set of labels) we simply compare the predictions of the Naïve-Bayes models trained on the pose labels and pick the highest probability occurrence. For the non-mutually exclusive sets (human-human interactions and human-object interactions) we manually enforce the label structure by first selecting the top three most likely labels on each classification instance and selecting from those the ones that are higher than a given threshold (ending up with 0-3 human-human interaction labels and 0-3 human-object labels, as enforced by the miniAVA/AVA dataset). Comparing this procedure with the LSTM approach, we point out that the Naïve-Bayes does not enjoy the properties of having sequencing in the classification procedure (in fact, the Naïve-Bayes model considers every instance to be independent of each other by definition) nor does it take advantage of the dataset's labelling structure during training. For example, in the pose labels, models trained with a softmax-like approach (*i.e.* multiclass models) will enjoy the property that the probabilities of the pose predictions sum to one. This implies that if the likelihood of a single label increases, all others must decrease, enforcing better performance and more accurate training. The Naïve-Bayes does not take

| Model\Properties | Output Structure | Sequencing |
|---|---|---|
| Naïve-Bayes | No | No |
| FFNN | Yes | No |
| LSTM | Yes | Yes |

Table 5.4: Model properties comparison for baseline testing.

advantage of this kind of structure.

In the FFNN models, the labelling structure is taken into account during the training procedure, in exactly the same way that the LSTM models do. However, FFNN models do not have any internal state modelling past classification instances, and, as such, are unable to do sequential classification.

The baseline testing is accurately explained in Table 5.4. This test will expose the contribution of each property to the overall model performance.

we did not run any tests on these datasets using the skeletal data context source. However, we have published preliminary results using skeletal data for action detection and recognition in BMVC 2018[21].

## 5.7 Remarks on Naïve-Bayes

The Naïve-Bayes classifier is a simplistic probabilistic classifier based on applying Bayes Theorem while assuming independence between features. Bayes' theorem yields Equation 5.7:

$$P(C|x) = \frac{P(C)p(x|C)}{P(x)} \tag{5.7}$$

where $C$ is the class having value 1 (in the binary scenario), $x = (x_1, x_2, ..., x_N)$ is the input data vector with N independent features, $P(C|x)$ is the posterior probability, $P(C)$ is the prior, $P(x|C)$ is the likelihood and $P(x)$ is the probability of the data or evidence. Since the denominator has no dependency on C, we can estimate the posterior by estimating the

numerator. Since the numerator is composed of the prior multiplied by the likelihood, it is equivalent to the joint distribution $P(C, x)$.

From the assumption of independence between features, the joint distribution can be estimated as show in Equation 5.8.

$$\begin{aligned} P(C, x) &= p(C, x_1, x_2, ..., x_N) \\ &= p(C) \prod_{i=1}^{N} P(x_i|C) \end{aligned} \tag{5.8}$$

In the semantic neighbor context scenario, $x = (x_1, x_2, ..., x_N)$ is the vector encoding the actions of the neighbors accross time (as described in Section 3.2). The independence assumption means that every action of every neighbor is independent, even actions performed by the same person, which, of course, is an undesirable constraint.

In the Multinomial Naïve-Bayes formulation, the one used for the Semantic Neighbor Context scenario, the feature vectors $x$ are modelled as outputs from K different multinomial distributions, where K is the number of classes (*i.e.* K multinomial distributions must be learned from the data, and K different binary classifiers must be learned for the task.). A feature vector $x = (x_1, x_2, ..., x_N)$ can be interpreted as a histogram where $x_i$ is the number of times feature $i$ was seen in a particular instance. Since this scenario involves K=30 classes, each histogram will have a different binary label $C_k$ (with $k = 1, ..., 30$) with which the joint distribution will be estimated.

Another limitation of Naïve-Bayes libraries is that there is no formal way to deal with the fact that there might be instances in which the exact same input is paired with different outputs. In the semantic neighbor context scenario, this refers to a situation in which all of the neighbors of the target are performing the same actions across time, but the target is performing a different action. Note, that since the Naïve-Bayes classifier has to be trained

as 30 independent classifiers (see Section 5.6) this can happen easily, as it only implies a change in a single label (going from performing one specific action to not performing it, or vice versa). In fact, such as the case, and during training, the input had to be pruned to remove these scenarios. This greatly simplifies the task. In comparison, neural network based approaches tackle this problem by assigning different non-zero probabilities to all possible outputs given the same input. Finally, due to the way that miniAVA was constructed (see Section 5.1) it is possible to have a label with no instances in the training set. Again, neural network based scenarios can learn from this and just attribute a small probability to this label, but the Naïve-Bayes libraries cannot. So in those instances, the model was hard coded to be unable to ever predict that label to occur. Given the highly unbalanced nature of this dataset, this too is an advantage for the Naïve-Bayes model.

# Chapter 6

# Results and Analysis

## 6.1 Appearance Context

In this section, the results obtained on the tests run on the Appearance Context source are shown and explained.

### 6.1.1 Baseline

For the baseline experiment, we need to establish what the mAP of an unaltered, state-of-the-art, Two-Stream CNN is on this task (See Section 4.3.4, as well as the performance of each individual stream. With this baseline we have a result against which we can establish a fair comparison of the performance of our method.

| Model\Dataset | miniAVA | UCF101-24 |
|---|---|---|
| RGB | 5.06% | 53.4% |
| OF | **5.85**% | 66.8% |
| RGB + OF | 5.00% | **73.3**% |

Table 6.1: Baseline individual streams and their fusion. Results are the video mAP at 0.5 IoU.

In Table 6.1 we show the results of this experiment. For the miniAVA dataset we note

how using only Optical Flow performs better than RGB. This has been also found to be the case for some implementations of these types of networks[2] and is often due to the fact that certain actions have very clear motion patterns. Furthermore, we highlight that the baseline of the RGB + OF fusion is lower than both of the fused streams in the miniAVA dataset, which suggests that, in this dataset, the spatiotemporal features being learned are not properly using complementary information from both streams, a result which we improve upon in further experiences.

On UCF101-24 the performance is a lot better for every model. This is another evidence of the high difficulty of the miniAVA dataset classification task. The results obtained in UCF101-24 are as not surprising, with a single OF stream outperforming a single RGB stream. The Two-Stream fusion approach outperforms either single stream. These results are to be expected, considering that the Two-Stream Architecture used had weights pre-trained on UCF101-24.

### 6.1.2 Single Stream RGB Actor Centric Filtering

In this experiment, the goal is to compare the actor centric filtering approaches tested: no-filtering, crop, GBB and fovea actor centric filtering. This experiment is run exclusively on the RGB stream.

| Model\Dataset | miniAVA | Relative Improvement | UCF101-24 | Relative Improvement |
|---|---|---|---|---|
| RGB | 5.06% | – | **53.4%** | – |
| RGB + GBB | **5.63%** | **+11.2%** | 45.3% | -15.1% |
| RGB + Crop | 5.19% | +2.5% | 24.5% | -54.1% |
| RGB + Fovea | 5.12% | +1.1% | 49.2% | -7.8% |

Table 6.2: Actor centric filtering results on individual RGB streams vs baseline (RGB).Results are the video mAP at 0.5 IoU.

In Table 6.2 we show the results of this experiment. For the miniAVA dataset, two main conclusions can be drawn from these results. One is that the use of all pre-filtering actor

centric mechanisms improve results. The second is that the filtering techniques that create artificial edges (i.e all except fovea filter, see Section 3.1) perform best. We think this is due to the fact that these artificial edges seem to be contributing to more accurate prediction of certain over represented classes, particularly *stand*, which is the most common class. An important observation to make is that the fact that actor centric filters provide improvement on the miniAVA dataset is not surprising. In this dataset, seldom do the labels of all targets in a video coincide, leading to incorrect classifications when no filtering is applied. If the network doesn't know *who* it should classify, how can it be expected to correctly predict the label of two different targets in the same video when their labels are not the same? We believe this to be why our actor centric approach provides the biggest improvement in the miniAVA dataset. We build on this notion in subsequent tests.

On the UCF101-24 dataset the RGB stream with no actor centric filtering achieves the best results. This is not surprising taking into account the characteristics of the UCF101-24 dataset. Although the dataset is multi-person, in a single video, every person will be performing the same activity. As such, allowing the network to look at what a person that is not the target is doing is actually reinforcing the networks performance. Regarding the actor centric filters, we have Fovea as the top performer, GBB in second place and Crop in last.

### 6.1.3 2-Stream Fusion Actor Centric Filtering

In this experiment we compare the results of the actor centric filters when fused in a two-stream CNN architecture.

In Table 6.3 we show the results of this experiment. For the miniAVA dataset, the first result is that cropped flow seems to worsen results when fused with all other streams except the cropped RGB stream, which is an interesting result that suggests some synergy in the learned features. We also note how the fovea filter performs better than all others

| Dataset | miniAVA | | | UCF101-24 |
|---|---|---|---|---|
| Model | OF | Relative Improvement | OF + Crop | OF |
| RGB | 5.00% | – | – | 73.3% |
| RGB + GBB | 3.59% | -28.2% | 4.16% | 70.9% |
| RGB + Crop | 5.01% | +0.2% | 5.06% | 68.7% |
| RGB + Fovea | **5.94%** | **+18.8%** | 4.95% | **74.5%** |

Table 6.3: Testing of several combinations of streams and their respective actor centric filters. Results are the video mAP at 0.5 IoU.

when fused with unfiltered flow and that it is the only two-stream approach that improves on all previous experiments.

For the UCF dataset we again notice that the Crop filtering approach severely underperforming. This is to be expected, considering the poor performance this approach had on this dataset even in single-stream RGB attempts (See Section 6.1.2). However, a surprising result is found: the Two-Stream fusion with Fovea actor centric filtering is outperforming the no-filter approach. Like we stated in Section 6.1.2, we expected no-filter to be the best performer in the UCF101-24 dataset.

The fovea actor centric filter outperforms all other approaches when fused with OF despite not being the best single stream approach (on UCF101-24 the best single stream approach was no-filter, and on miniAVA GBB filtering. See Table 6.2). This seems to indicate that the features learned using actor centric filtering on RGB better complement the features learned by the OF stream.

## 6.2 Semantic Neighbor Context

In this section, the results obtained on the tests run on the Semantic Context source are shown and explained.

We run four tests on semantic neighbor context: 1) hyperparameter tuning, 2) two passes vs three passes (see Sec. 3.2)., 3) Noise Resilience Testing, 4) Baseline Testing.

### 6.2.1 Hyperparameter Tuning

| A | 64 | 128 | 256 | 512 | 1024 | 2048 |
|---|---|---|---|---|---|---|
| T=3 | 4.99% | 5.04% | 4.96% | 4.80% | – | – |
| T=5 | 5.01% | 4.97% | 4.98% | **5.05%** | 5.09% | 5.11% |
| T=10 | 4.85% | 5.00% | 4.68% | 5.04% | – | – |
| B | 64 | 128 | 256 | 512 | 1024 | 2048 |
| T=3 | 5.01% | 5.00% | 4.92% | **5.12%** | 5.08% | 5.09% |
| T=5 | 4.93% | 4.94% | 4.95% | 5.04% | – | – |
| T=10 | 4.81% | 4.90% | 4.92% | 4.97% | – | – |

Table 6.4: Evaluation of the best context generation for model A and model B LSTM architecture (see Figure 3.2, with the columns corresponding to the number of hidden units. All results use $N = 3$ (i.e three closest neighbours)

In Table 6.4 we show the results obtained for the hyperparameter search. We use the ground truth labels to generate the input, as explained in Sec. 3.2. We set the number of neighbors to three by noticing that the average number of people in the miniAVA dataset to be less than two. When we generate the input, if there are less people present in the frame than neighbors being considered by the model, we simply set those to actionless neighbors with no position (*i.e.:* the model actually contemplates *up to* 3 neighbors). While model B is marginally better, model A seems to operate better with a slightly larger time window, regardless of the fact that there is not a significant difference between the best model A architecture or the best model B architecture. Model A seems to need larger layers to model more complex sequences successfully. This is desirable for the purposes of the MVC framework since we plan on having the possibility of fusing the results of this model to the other classification models applied to the other context sources. Having this model have a number of hidden units closer to 1024 (the number of units used by our actor centric filtering approach), the fusion is not unbalanced by the number of units. As such, we use model A in the next experiments.

| Model | mAP@0.5IoU |
|---|---|
| RGB + Fovea + OF | 5.94% |
| Groundtruth Input | **9.11%** |
| Two Pass Input | 6.24% |
| Three Pass Input | **6.28%** |

Table 6.5: Testing context fusion architectures in a groundtruth scenario, *i.e.* using test labels to generate context and our Two Pass scheme, both with class score fusion. A final additional pass is also tested.

### 6.2.2 Two pass vs Three pass

In Table 6.5 the results of fusing our best performing LSTM network for semantic neighbor context with the actor centric filtered CNN are shown. If ground truth data is used to generate the input to the LSTM the result improves greatly to 9.11%. However, this is not a realistic scenario, and not very interesting to discuss. When using the labels obtained from our RGB + Fovea + OF actor centric model to generate the input to the semantic neighbor context LSTM the results are very interesting. First of all, to get an improvement in the performance when fusing, taking into account that the input data is wrong 94% of the cases is, impressive. Furthermore, the relative performance increase is in the order of 5%. When we then take the labels generated from the two-pass scheme as input for a three-pass approach, we observe further improvement, albeit marginal. These results are promising, and convey the potential of our MVC framework for fusing additional context sources.

### 6.2.3 Semantic Neighbor Context Noise Resilience

In Table 6.6 the results of feeding increasingly more corrupted data to our LSTM models is shown. Since this test is just to compare the performance of models trained in exactly the same conditions, the results are presented in accuracy, for ease of interpretation. Unsurprisingly, the trend is to have worse performance with increasing input corruption. There is a surprising result with $50\%$ noise on the input achieving the highest performance of all instances tested. This can be explained by the intrinsic random nature of this test: the input

| NHU\Noise | 0% | 5% | 25% | 50% | 75% | 95% | 100% |
|---|---|---|---|---|---|---|---|
| 32 | 33.06% | 33.04% | 32.47% | 31.73% | 30.06% | 25.71% | 26.74% |
| 64 | 32.93% | 32.87% | 33.16% | 31.59% | 29.29% | 26.06% | 26.93% |
| 128 | 33.19% | 33.25% | 32.82% | 32.06% | 29.60% | 26.39% | 26.77% |
| 256 | 33.53% | 32.80% | 34.21% | **34.68**% | 29.62% | 26.47% | 26.77% |
| 512 | 33.01% | 34.06% | 29.90% | 31.68% | 29.32% | 30.10% | 26.49% |

Table 6.6: Testing noise resilience properties of our semantic neighbor context embedding and LSTM model. Results shown in accuracy.

is being randomized with each experiment by being subjected to random label flipping on the input. However, the trend of having worse performance with higher noise is still easily observable. All models seem to hover around 26% accuracy with 100% noise on the input. This is to be expected, considering the highly unbalanced dataset. Although noise is being applied to the input, noise is not being applied on the output (*i.e.* the labels used on the training process for these models are the correct labels). In that scenario the models are basically learning the most common classes for each task and learning to predict those to minimize their cost functions in a scenario where no learning is possible from the input data.

### 6.2.4 Baseline Testing: LSTM vs FFNN vs NB

In Tables 6.7, 6.8 and 6.9 we show the results of comparing the performance of Naïve-Bayes, feed-forward neural networks and LSTM networks on increasingly noisier input data. Since the training and preprocessing procedure for the Naïve-Bayes model was different, we show the results in F-1 score. This metric has the advantage of averaging performance across labels taking dataset imbalance into account.

| Noise | 0% | 5% | 25% | 50% | 75% | 95% | 100% |
|---|---|---|---|---|---|---|---|
| F1-Score | 42.31% | **43.17**% | 42.39% | 39.47% | 35.71% | 31.78% | 31.80% |

Table 6.7: F-1 Scores for different noise values for the Naïve-Bayes models.

| NHU/Noise | 0% | 5% | 25% | 50% | 75% | 95% | 100% |
|---|---|---|---|---|---|---|---|
| 32 | 41.05% | 39.88% | 39.70% | 35.25% | 35.61% | 25.67% | 31.06% |
| 64 | 45.01% | 40.55% | 47.19% | 40.67% | 38.79% | 27.57% | 28.76% |
| 128 | 43.85% | 35.18% | 41.77% | 38.31% | 38.12% | 30.01% | 36.49% |
| 256 | 42.68% | 47.01% | 38.21% | 37.15% | 34.36% | 24.28% | 35.87% |
| 512 | 39.27% | **47.47**% | 40.00% | 38.33% | 27.70% | 32.02% | 28.45% |

Table 6.8: F-1 Scores for different noise values for the FFNN architecture

| NHU/Noise | 0% | 5% | 25% | 50% | 75% | 95% | 100% |
|---|---|---|---|---|---|---|---|
| 32 | 36.06% | 36.91% | 37.44% | 31.46% | 32.59% | 19.55% | 23.62% |
| 64 | 38.24% | 38.91% | 31.78% | 37.30% | 31.65% | 28.21% | 25.93% |
| 128 | **43.41**% | 33.91% | 42.55% | 43.16% | 32.86% | 27.87% | 24.35% |
| 256 | 40.83% | 39.41% | 32.25% | 31.28% | 32.60% | 21.99% | 34.42% |
| 512 | 39.45% | 43.36% | 41.42% | 27.77% | **44.38**% | 29.62% | 23.40% |

Table 6.9: F-1 Scores for different noise values for the LSTM architecture

Looking first at the comparison between the results obtained by the LSTM and FFNN models, an interesting result appears. There isn't much to be gained by incorporating sequential classification into this problem, as shown by the fact that the FFNN models seem to outperform the LSTM architectures. The only difference in those two types of models is exclusively the usage of LSTM layers or fully connected layers in the first layers of the model. Both the training parameters (*i.e.* number of epochs, learning rate, cost functions...) and the output layers are exactly the same, which points to the conclusion that the sequential processing of this data is not beneficial to the system's performance.

Comparing the LSTM and FFNN results with the ones obtained in Naïve-Bayes models, shown in Table 6.7, is a more delicate subject. At first glance, the performance of the Naïve-Bayes models seems slightly worse, but there are many factors at play here. Not only is the training procedure different (necessarily so, as Naïve-Bayes is not a neural network based approach) but there are also pre-processing steps that had to be applied to this model that were not required to the FFNN and LSTM models. As described in Section 5.7, the input vectors had to be pruned not to have duplicates with different output labels, and actions that never have a positive sample in the training set are effectively excluded from

the output space. Both of these changes make the problem a bit simpler for Naïve-Bayes. However, there are also handicaps in this model: this approach can only be trained as 30 independent binary classifiers that are then joined *a posteriori*. The advantages that this model has are hard to measure independently. But the effect of its handicap can be easily verified with confusion matrices.

In Tables 6.10, 6.11 and 6.12 we show the confusion matrices of the pose labels (the set of mutually exclusive labels) obtained from the predictions on the test set of the best performing models in a zero-noise scenario. The models that obtained these results were the FFNN with 64/32 hidden units and the LSTM with 128/64 hidden units. We show the results on the 0% noise scenario because that is the scenario is which we are sure that random input switching is not affecting the results.

The most common labels in the dataset are labels 8, 9 and 10, respectively *sit*, *stand* and *walk*. In the case of the FFNN and LSTM models, during training, these labels were learned using a softmax classifier. This implies that whenever a label's score goes up (*i.e.* when it approaches 1) the scores of all other labels must go down. The effect this had on training is that very rarely will the models predict a class of low occurrence as long as the model has evidence to believe it can me mistaken for a different class. This can be seen on Tables 6.11 and 6.12, noting that labels 7 and below are very rarely (if ever) predicted, although they do have some presence in the training data. Comparing this with the results found in Table 6.10, for Naïve-Bayes, the impact of independent binary training is apparent. The confusion matrix is much more spread out, with values spread out on every label. In particular, this has a strong impact on the correct classification of label 9, with every label being mistakenly assigned to it. The takeaway from these tests is that, although on the surface, the performance of the Naïve-Bayes may not seem very far away from the performance of the other two models, the impact of this methods handicaps are easily apparent, and the limitations it imposes on the training procedure are not justified

by performance. Furthermore, this model cannot be generalized,to situations in which the input cannot be pruned.

| True Class\Predicted Class | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 14 | 2 | 0 | 0 | 4 | 6 | 0 | 20 | 76 | 15 |
| 2 | 5 | 0 | 0 | 0 | 0 | 3 | 0 | 1 | 8 | 6 |
| 3 | 4 | 5 | 7 | 9 | 5 | 4 | 1 | 0 | 6 | 5 |
| 4 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 5 | 14 | 0 |
| 5 | 0 | 0 | 0 | 0 | 9 | 2 | 0 | 8 | 51 | 10 |
| 6 | 0 | 0 | 0 | 0 | 0 | 89 | 0 | 2 | 43 | 0 |
| 7 | 5 | 1 | 0 | 4 | 1 | 4 | 1 | 11 | 29 | 23 |
| 8 | 61 | 4 | 1 | 0 | 16 | 5 | 0 | 893 | 208 | 56 |
| 9 | 241 | 11 | 3 | 6 | 22 | 110 | 2 | 235 | 3,230 | 249 |
| 10 | 17 | 3 | 3 | 1 | 5 | 6 | 0 | 37 | 291 | 452 |

Table 6.10: Confusion matrix for the pose labels of the Naïve-Bayes model with 0% noise on the input.

| True Class\Predicted Class | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 17 | 110 | 10 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 17 | 4 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 41 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 17 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 76 | 4 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 134 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8 | 68 | 3 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 791 | 421 | 32 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 145 | 3,874 | 90 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 41 | 468 | 306 |

Table 6.11: Confusion matrix for the pose labels of the FFNN model with 64/32 hidden units and 0% noise on the input.

## 6.3 Hybrid Sigmoid-Softmax Loss

In this experiment, we validate the choice of using the custom loss by showing the results of the best individual RGB stream when trained with a single output layer consisting only of sigmoid activation functions. This sum-of-sigmoids loss function is employed in the

| True Class\Predicted Class | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 19 | 104 | 14 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 18 | 4 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 35 | 6 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 17 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8 | 67 | 5 |
| 6 | 4 | 0 | 0 | 0 | 11 | 0 | 0 | 1 | 116 | 2 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 62 | 7 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 815 | 385 | 44 |
| 9 | 2 | 0 | 0 | 0 | 1 | 0 | 0 | 162 | 3,791 | 153 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 45 | 404 | 366 |

Table 6.12: Confusion matrix for the pose labels of the LSTM model with 128/64 hidden units and 0% noise on the input.

miniAVA baseline[3].

| Model\Dataset | miniAVA |
|---|---|
| RGB + GBB - Hybrid Sigmoid-Softmax Loss | **5.63%** |
| RGB + GBB - Sum-of-Sigmoids | 5.01% |

Table 6.13: Results that validate the choice of a custom loss against a standard binary cross entropy loss.

In Table 6.13 we show the results of this experiment. Our loss achieves an video mAP of $5.63\%$, representing a relative mAP improvement of $12.6\%$ over using a standard sum-of-sigmoids loss. We believe this is due to the fact that our Hybrid Sigmoid-Softmax Loss takes into account the mutual exclusion of the pose classes, while the Sum-of-Sigmoids approach does not.

# Chapter 7

# Conclusion and Future Work

## 7.1 Conclusion

In this work, a flexible, extensible and modular framework for context source testing and fusion - the Multisource Video Classification (MVC) framework is proposed. This framework enriches the CNN-based approaches by allowing extra sources of information to be leverage into performance gains. The context sources tested, actor centric filtering and semantic neighbor context both improve the performance of the tested models when used. Further tests reveal the noise resilience properties of the semantic neighbor context models. A custom loss function the Hybrid Sigmoid-Softmax loss, that leverages the structure present when the label sets of the dataset being classified are only partly mutually exclusive is also tested, also showing performance improvements when compared with the sum of sigmoids standard. Finally, ablation tests determining the importance of sequential classification and structured output functions in the semantic neighbor context approach are also presented. These tests show that although sequential classification does not warrant significant performance gains, the structure output has verifiable impact in reducing class confusion, particularly for the mutually exclusive classes.

From this work, we have one paper accepted at BMVC 2018[21], describing preliminary results on action detection and recognition, one paper submitted to ECCV 2020 describing the proposed actor centric filtering approaches, and a paper on the results obtained with our LSTM architectures for semantic neighbor context is currently under preparation.

## 7.2 Future Work

In the future, we wish to expand the testing on the MVC framework to include all four types of context sources discussed. This will improve the innovation achieved with the framework, while still improving its flexibility, without restricting its modularity and extensibility.

Furthermore, it would be relevant to test the Actor Centric Filtering approach using state-of-the-art models like I3D[2] or the Slow-Fast Networks[45] and filtering the feature maps that come from the region proposal network part of those architectures. This testing is motivated by the fact that performance gains were obtained by using the foveation filters instead of cropping (cropping is currently the approach used by the state-of-the-art models.

Finally, for the semantic neighbor context approach, devising a way to insert attention based modules in the FFNN models seems a promising way to boost its performance, following the trend shown by the NLP community.

# Bibliography


[1] Varsamopoulos, S., Bertels, K. & Almudever, C. G. Designing neural network based decoders for surface codes. *arXiv preprint arXiv:1811.12456* (2018).

[2] Carreira, J. & Zisserman, A. Quo vadis, action recognition? a new model and the kinetics dataset. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 4724–4733 (2017).

[3] Gu, C. *et al.* Ava: A video dataset of spatio-temporally localized atomic visual actions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[4] Tripathi, R. K., Jalal, A. S. & Agrawal, S. C. Suspicious human activity recognition: a review. *Artificial Intelligence Review* **50**, 283–339 (2018).

[5] Abu-El-Haija, S. *et al.* Youtube-8m: A large-scale video classification benchmark. In *arXiv:1609.08675* (2016).

[6] Kang, S. & Wildes, R. P. Review of action recognition and detection methods. *CoRR* **abs/1610.06906** (2016). 1610.06906.

[7] Stratonovich, R. L. Conditional markov processes. In *Non-linear transformations of stochastic processes*, 427–453 (Elsevier, 1965).

[8] Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *nature* **323**, 533–536 (1986).

[9] Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–1780 (1997).

[10] Sun, J. *et al.* Hierarchical spatio-temporal context modeling for action recognition. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2004–2011 (IEEE, 2009).

[11] Zhang, Z., Hu, Y., Chan, S. & Chia, L.-T. Motion context: A new representation for human action recognition. In *European Conference on Computer Vision*, 817–829 (Springer, 2008).

[12] Liu, J., Wang, G., Hu, P., Duan, L.-Y. & Kot, A. C. Global context-aware attention lstm networks for 3d action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1647–1656 (2017).

[13] Wu, X., Xu, D., Duan, L. & Luo, J. Action recognition using context and appearance distribution features. In *CVPR 2011*, 489–496 (IEEE, 2011).

[14] Han, D., Bo, L. & Sminchisescu, C. Selection and context for action recognition. In *2009 IEEE 12th International Conference on Computer Vision*, 1933–1940 (IEEE, 2009).

[15] Soomro, K., Zamir, A. R. & Shah, M. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402* (2012).

[16] Russakovsky, O. *et al.* Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* **115**, 211–252 (2015).

[17] Tran, D. *et al.* A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6450–6459 (2018).

[18] Tran, D., Bourdev, L. D., Fergus, R., Torresani, L. & Paluri, M. C3D: generic features for video analysis. *CoRR* **abs/1412.0767** (2014). 1412.0767.

[19] Xu, H., Das, A. & Saenko, K. R-C3D: region convolutional 3d network for temporal activity detection. *CoRR* **abs/1703.07814** (2017). 1703.07814.

[20] Simonyan, K. & Zisserman, A. Two-stream convolutional networks for action recognition in videos. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, 568–576 (MIT Press, Cambridge, MA, USA, 2014).

[21] Antunes, J., Bernardino, A., Smailagic, A. & Siewiorek, D. AHA-3D: A labelled dataset for senior fitness exercise recognition and segmentation from 3d skeletal data. *Vision for Interaction and Behaviour Understanding (VIBE) Workshop, British Machine Vision Conference (BMVC)* (2018).

[22] Hara, K., Kataoka, H. & Satoh, Y. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? *CoRR* **abs/1711.09577** (2017). 1711.09577.

[23] Ji, S., Xu, W., Yang, M. & Yu, K. 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **35**, 221–231 (2013).

[24] Karpathy, A. *et al.* Large-scale video classification with convolutional neural networks. In *CVPR* (2014).

[25] Mikolov, T., Chen, K., Corrado, G. & Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[26] Pennington, J., Socher, R. & Manning, C. D. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543 (2014).

[27] Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J. & Khudanpur, S. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association* (2010).

[28] Chelba, C. *et al.* One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005* (2013).

[29] Vaswani, A. *et al.* Attention is all you need. In *Advances in neural information processing systems*, 5998–6008 (2017).

[30] Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[31] Csurka, G., Dance, C. R., Fan, L., Willamowski, J. & Bray, C. Visual categorization with bags of keypoints. In *In Workshop on Statistical Learning in Computer Vision, ECCV*, 1–22 (2004).

[32] Wang, H. & Schmid, C. Action recognition with improved trajectories. In *Proceedings of the 2013 IEEE International Conference on Computer Vision*, ICCV '13, 3551–3558 (IEEE Computer Society, Washington, DC, USA, 2013).

[33] Sánchez, J., Perronnin, F., Mensink, T. & Verbeek, J. Image classification with the fisher vector: Theory and practice. *International journal of computer vision* **105**, 222–245 (2013).

[34] Perronnin, F., Sánchez, J. & Mensink, T. Improving the fisher kernel for large-scale image classification. In Daniilidis, K., Maragos, P. & Paragios, N. (eds.) *Computer Vision – ECCV 2010*, 143–156 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2010).

[35] Jégou, H., Douze, M., Schmid, C. & Pérez, P. Aggregating local descriptors into a

compact image representation. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 3304–3311 (2010).

[36] Ng, J. Y. *et al.* Beyond short snippets: Deep networks for video classification. *CoRR* **abs/1503.08909** (2015). 1503.08909.

[37] Donahue, J. *et al.* Long-term recurrent convolutional networks for visual recognition and description. *IEEE Trans. Pattern Anal. Mach. Intell.* **39**, 677–691 (2017).

[38] Xie, S., Sun, C., Huang, J., Tu, Z. & Murphy, K. Rethinking spatiotemporal feature learning for video understanding. *CoRR* **abs/1712.04851** (2017). 1712.04851.

[39] Bai, S., Kolter, J. Z. & Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *CoRR* **abs/1803.01271** (2018). 1803.01271.

[40] Feichtenhofer, C., Pinz, A. & Zisserman, A. Convolutional two-stream network fusion for video action recognition. *CoRR* **abs/1604.06573** (2016). 1604.06573.

[41] Wang, L. *et al.* Temporal segment networks for action recognition in videos. *IEEE transactions on pattern analysis and machine intelligence* (2018).

[42] Girdhar, R., Ramanan, D., Gupta, A., Sivic, J. & Russell, B. ActionVLAD: Learning spatio-temporal aggregation for action classification. In *IEEE Conference on Computer Vision and Pattern Recognition* (Honolulu, United States, 2017). Project page: https://rohitgirdhar.github.io/ActionVLAD/.

[43] Zhu, Y., Lan, Z., Newsam, S. D. & Hauptmann, A. G. Hidden two-stream convolutional networks for action recognition. *CoRR* **abs/1704.00389** (2017). 1704.00389.

[44] Tran, D. *et al.* A closer look at spatiotemporal convolutions for action recognition. *CoRR* **abs/1711.11248** (2017). 1711.11248.

[45] Feichtenhofer, C., Fan, H., Malik, J. & He, K. Slowfast networks for video recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, 6202–6211 (2019).

[46] Xia, J., Tang, J. & Lu, C. Three branches: Detecting actions with richer features. *arXiv preprint arXiv:1908.04519* (2019).

[47] Li, W., Yuan, Z., Zhao, A., Shao, J. & Wang, C. Bytedance ai lab ava challenge 2019 technical report .

[48] Jiang, J. *et al.* Human centric spatio-temporal action localization. In *ActivityNet Workshop on CVPR* (2018).

[49] Girdhar, R., Carreira, J., Doersch, C. & Zisserman, A. A better baseline for ava. *arXiv preprint arXiv:1807.10066* (2018).

[50] Yao, T. & Li, X. Yh technologies at activitynet challenge 2018. *arXiv preprint arXiv:1807.00686* (2018).

[51] Kneser, R. & Ney, H. Improved backing-off for m-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, 181–184 (IEEE, 1995).

[52] Bengio, Y. & Senécal, J.-S. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks* **19**, 713–722 (2008).

[53] Wang, T. & Cho, K. Larger-context language modelling. *arXiv preprint arXiv:1511.03729* (2015).

[54] Ji, Y., Cohn, T., Kong, L., Dyer, C. & Eisenstein, J. Document context language models. *arXiv preprint arXiv:1511.03962* (2015).

[55] Ren, S., He, K., Girshick, R. & Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M. & Garnett, R. (eds.) *Advances in Neural Information Processing Systems 28*, 91–99 (Curran Associates, Inc., 2015).

[56] Almeida, A. F., Figueiredo, R., Bernardino, A. & Santos-Victor, J. Deep networks for human visual attention: A hybrid model using foveal vision. In Ollero, A., Sanfeliu, A., Montano, L., Lau, N. & Cardeira, C. (eds.) *ROBOT 2017: Third Iberian Robotics Conference*, 117–128 (Springer International Publishing, Cham, 2018).

[57] Traver, V. J. & Bernardino, A. A review of log-polar imaging for visual perception in robotics. *Robotics and Autonomous Systems* **58**, 378–398 (2010).

[58] Burt, P. & Adelson, E. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications* **31**, 532–540 (1983).

[59] Cao, Z., Simon, T., Wei, S.-E. & Sheikh, Y. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR* (2017).

[60] Simon, T., Joo, H., Matthews, I. & Sheikh, Y. Hand keypoint detection in single images using multiview bootstrapping. In *CVPR* (2017).

[61] Wei, S.-E., Ramakrishna, V., Kanade, T. & Sheikh, Y. Convolutional pose machines. In *CVPR* (2016).

[62] Cao, Z., Hidalgo, G., Simon, T., Wei, S.-E. & Sheikh, Y. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In *arXiv preprint arXiv:1812.08008* (2018).

[63] Jhuang, H., Gall, J., Zuffi, S., Schmid, C. & Black, M. J. Towards understanding action recognition. In *Proceedings of the IEEE international conference on computer vision*, 3192–3199 (2013).

[64] Goodfellow, I. J., Bengio, Y. & Courville, A. *Deep Learning* (MIT Press, Cambridge, MA, USA, 2016).

[65] Feichtenhofer, C., Pinz, A. & Wildes, R. P. Spatiotemporal multiplier networks for video action recognition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7445–7454 (2017).

[66] Feichtenhofer, C., Pinz, A. & Wildes, R. P. Spatiotemporal residual networks for video action recognition. In *NIPS* (2016).

[67] Ghanem, B. *et al.* The activitynet large-scale activity recognition challenge 2018 summary (2018). 1808.03766.

[68] Zhao, Y. *et al.* Temporal action detection with structured segment networks. *ICCV, Oct* **2** (2017).

[69] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. & Zisserman, A. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision* **88**, 303–338 (2010).

[70] Wang, L., Xiong, Y., Wang, Z. & Qiao, Y. Towards good practices for very deep two-stream convnets. *CoRR* **abs/1507.02159** (2015). 1507.02159.

[71] He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 770–778 (2016).