



Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA

# Temporal Activity Detection in Untrimmed Videos with Recurrent Neural Networks

Final Degree's Thesis
Telecommunications Science and Technology

**Author:** Alberto Montes Gómez
**Advisors:** Xavier Giró-i-Nieto and Amaia Salvador

**Universitat Politècnica de Catalunya (UPC)**
**2015 - 2016**

# Abstract

This thesis explore different approaches using Convolutional and Recurrent Neural Networks to classify and temporally localize activities on videos, furthermore an implementation to achieve it has been proposed.

As the first step, features have been extracted from video frames using an state of the art 3D Convolutional Neural Network. This features are fed in a recurrent neural network that solves the activity classification and temporally location tasks in a simple and flexible way.

Different architectures and configurations have been tested in order to achieve the best performance and learning of the video dataset provided. In addition it has been studied different kind of post processing over the trained network's output to achieve a better results on the temporally localization of activities on the videos.

The results provided by the neural network developed in this thesis have been submitted to the ActivityNet Challenge 2016 of the CVPR, achieving competitive results using a simple and flexible architecture.

# Resumen

Esta tesis explora diferentes enfoques usando Redes Neuronales Convolucionales y Redes Neuronales Recurrentes para clasificar y localizar temporalmente actividades en videos y propone una implementación propia.

Como primer paso, se han extraido descriptores de videos usando Redes Neuronales Convolucionales 3D del estado del arte. Estos descriptores se introducen en una Red Neuronal Recurrente que resuelve la clasificación de actvidades y su localización temporal de una manera simple y flexible.

Diferentes arquitecturas y configuraciones se han testeado con el objetivo de conseguir el mejor resultado y aprendizaje del conjunto de vídeos subministrado. Además, se han estudiado diferentes tipos de post procesado sobre la salida de la red entrenada para conseguir mejores resultados en la localización de actividades en los vídeos.

Los resultados obtenidos por la red neuronal desarrollada en esta tesis han sido publicados en la ActivityNet Challenge 2016 del CVPR consiguiendo resultados competitivos con una simple y flexible arquitectura.

# Resum

Aquesta tesis explora diferents enfocaments utilitzant Xarxes Neuronals Convolucionals i Xarxes Neuronals Recurrents per classificar i localitzar temporalment activitats en videos i proposa una implementació pròpia.

Com a primer pas, s'han extret descriptors de videos utilitzant Xarxes Neuronals Convolutionals 3D de l'estat de l'art. Aquests descriptors s'han introduït en una Xarxa Neuronal Recurren que resol la classificació d'activitats i la seva localització temporal d'una manera simple i flexible.

Diferents arquitectures i configuracions han estat testejades amb l'objectiu d'aconseguir el millor resultat i aprenentatge del conjunt de videos subministrats. A més, s'ha estudiat diferents tipus de post processat sobre la sortida de la xarxa entrenada per aconseguir els millors resultats en la localització d'activitats en els videos.

Els resultats obtinguts per la xarxa neuronal desenvolupada en aquesta tesis han estat publicats a la ActivityNet Challenge 2016 del CVPR aconseguint resultats competitius amb una simple i flexible arquitectura.

# Acknowledgments

First of all I would like to thank my two advisors of this project, Xavier Giró-i-Nieto and Amaia Salvador for guiding and teaching me during all this project. I am aware that without their help I would not have been able to achieve this thesis.

I would also like to thank Santi Pascual and Professor Ignasi Esquerra for their help on topics they master: Recurrent Neural Networks and audio descriptors respectively. I should give also special thanks to Albert Gil for all the technical support provided along the development of this thesis.

Finally I would like to thank my family and the people closest to me for their encouragment, valuable support and patience during the development of my thesis.

# Revision History and Approval Record

| Revision | Date | Purpose |
|---|---|---|
| 0 | 10/06/2016 | Document creation |
| 1 | 23/06/2016 | Document revision |
| 2 | 26/06/2016 | Document revision |
| 3 | 27/06/2016 | Document approbation |

DOCUMENT DISTRIBUTION LIST

| Name | e-mail |
|---|---|
| Alberto Montes | alberto.montes@estudiant.upc.edu |
| Xavier Giró-i-Nieto | xavier.giro@upc.edu |
| Amaia Salvador | amaia.salvador@upc.edu |

| Written by: | | Reviewed and approved by: | | Reviewed and approved by: | |
|---|---|---|---|---|---|
| **Date** | 24/06/2016 | **Date** | 27/06/2016 | **Date** | 27/06/2016 |
| **Name** | Alberto Montes | **Name** | Xavier Giró-i-Nieto | **Name** | Amaia Salvador |
| **Position** | Project Author | **Position** | Project Supervisor | **Position** | Project Supervisor |

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Statement of Purpose

Recognizing activities in videos has become a hot topic over the last years in the computer vision community [20]. The exponential growth of portable video cameras and online multimedia repositories, as well as recent advances in video coding, storage and computational resources have motivated an intense research in the field towards new and more efficient solutions for organizing, understanding and retrieving video content.

Deep learning techniques have recently become the new state of the art in many computer vision tasks, such as image and object recognition in still images. While successful methodologies have been presented for image understanding, video content still presents additional challenges (e.g. motion, temporal consistency, ...) that often cannot be bridged with still image recognition solutions.

The purpose of this work is to address the challenges of video content analysis taking advantage of state-of-the-art deep learning techniques. The aim of this project is to develop a competitive framework to both classify and temporally localize activities on videos. To achieve this goal, the dataset used to fulfill this task is be the ActivityNet dataset [10], which offers untrimmed videos depicting a diversity of human activities.

In particular, this project's main contributions are:

- The design and training of a deep learning model architecture, which is based on 3D Convolutional features and Recurrent Neural Networks.
- The development and analysis of post processing techniques for classification and temporal localization
- The release of an open sourced package containing all the tools to reproduce the experiments, as well as the conversion of a state-of-the-art C3D model from Caffe to Keras.

This project has been developed at the *Image Processing Group (GPI) of the Universitat Politècnica de Catalunya (UPC)* during the Spring 2016 semester. This group had already work in deep learning techniques for still images, but never before in video analytic.

## 1.2 Requirements and Specifications

This project has been developed with the implicit goal of setting a baseline for video analytic with deep learning in the research group, so that future students and researchers to keep working on it. The requirements of this project are the following:

- Design and train a deep neural network to classify and temporally localize activities on videos using the ActivityNet Dataset.
- Participate in the ActivityNet Challenge 2016, organized as a workshop in the top conference IEEE Conference on Computer Vision and Pattern Recognition (h5-index = 128).

As this project has been developed from scratch, no prior specifications were defined. The specifications were decided taking into account the needs for the project and the available resources. All the development was done on *Python* using a very well-known framework which is *Keras*, also a pioneering use of this library in GPI. This Deep Learning wrapper facilitates the design and training of models over two computational frameworks: *Theano* [29] and *TensorFlow* [1]. Both projects support complex and high demanding computations over both CPU and GPU. For this project *Theano* was used as back-end because, at the time of developing this project, it was the only one that had implemented the convolution 3D and max pooling 3D operations required for its development.

In addition to the software, specific hardware was required. The high demanding computational resources needed to train neural networks required the use of GPUs provided by the *Image Processing Group* at UPC.

## 1.3 Methods and Procedures

This project tries to offer a simple and flexible solution to face the tasks classification and temporal localization on videos. The solution proposed is made by a first step stage where the video features are extracted. To do so, a pre-trained 3D convolutional network has been used to extract features from spatial and temporal dimension from video frames. In parallel, features from audio were extracted to combine them with the ones extracted from the video frames.

As a second stage, a Recurrent Neural Network is proposed to exploit long term relations and predict the sequence of activities available at each video. This network was tested with different architectures and inputs from the previously extracted features. Figure 1.1 shows an schematic of the proposed pipeline.

All the tests were done using a recently publish video dataset, the ActivityNet [10] which offers untrimmed videos from 200 activities. All the videos from the dataset present a single activity on it so its possible to classify them globally. Furthermore the activities along the video are temporally localized. Most of the effort and architecture was performed taking into account resolving the detection task can be easily extrapolated to solve the global classification task for the whole video.

Moreover, multiple post-processing techniques were proposed to improve the classification and detection tasks. Multiple configurations were tested in order to maximize the results.

Finally, the best results were submitted to the ActivityNet Challenge 2016 at the CVPR.

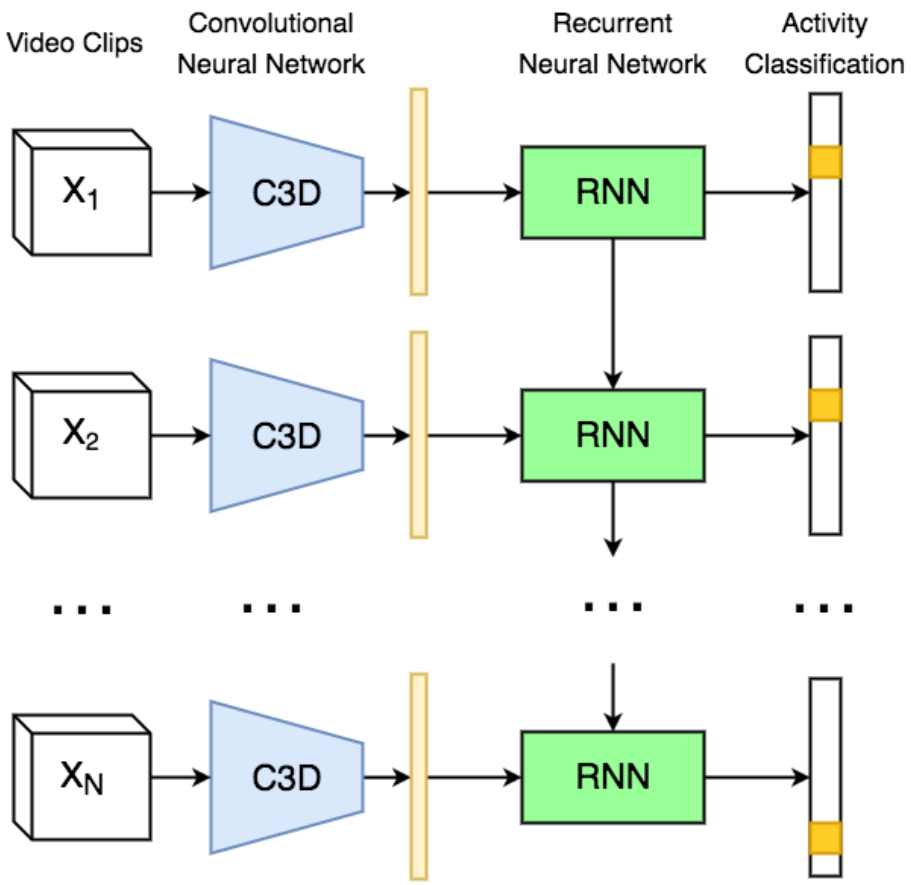



Figure 1.1: Global architecture of the solution proposed on this project.

## 1.4 Work Plan

This project was planned to follow the packages detailed on this section, with the exception of some minor deviations described in Section 1.5.

### 1.4.1 Work Packages

- WP 1: Project Documentation
- WP 2: Research for the State of the Art
- WP 3: Dataset to work with
- WP 4: Software to use
- WP 5: Experimentation and Results Evaluation
- WP 6: ActivityNet Challenge 2016 Participation
- WP 7: Delivery and Exposition of this project

### 1.4.2 Gantt Diagram

The Gantt diagram of this project can be found on the Figure 1.2.

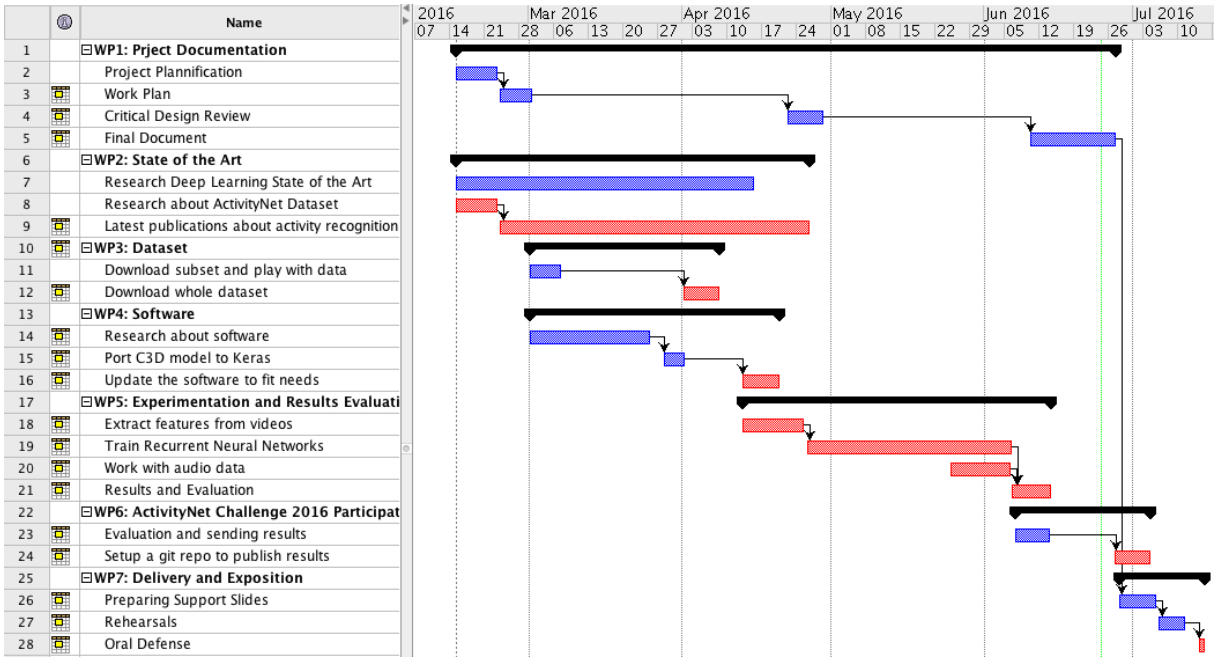



Figure 1.2: Gantt diagram of this thesis.

## 1.5 Work Plan Deviations and Incidents

The initial plan was to adapt the C3D [30] model for the ActivityNet task, which is implemented with the *Caffe* deep learning framework. But in order to use Recurrent Neural Networks, *Keras* was a much better and flexible framework, so it was necessary to export the original C3D model from the Caffe framework to Keras.

Another deviation for the plan was that, due to the huge amount of required computational resources, fine tuning the full C3D network was not possible. Instead, the C3D framework was used to extract features from the videos, which were later used as inputs to train the recurrent neural network.

Finally, the original work plan was extended by exploring the potential of audio descriptors for activity recognition. For this task, this work was supported by Professor Ignasi Esquerra from the TALP research group at UPC. He took care of extracting the audio features, which were later assessed and fused with the visual ones.

# Chapter 2

# State of the Art

Many works in the literature have explored activity and action recognition problems using deep learning strategies. Most of them used an approach of two stages (encoder and decoder) trained on a dataset of videos. The first stage, the one being consider a decoder, is the stage that tries to encode the visual and temporal information from the input videos into some features vectors or information. The second stage, also known as decoder, casts a prediction of the output, which can be a classification of the video or a temporal localization of an activity.

## 2.1 Convolutional Neural Networks applied to Videos

A popular solution as encoder are Convolutional Neural Networks (or also known as CNNs) as it has been widely demonstrated that applying CNN networks to images and videos produces good results in classifications tasks. A very well known and used CNN network is the VGG [25] network which was top scored on ImageNet Challenge in 2014 on classification task. This network uses 2D convolutional kernels to extract spatial correlations from images.

To understand better how Convolutional Neural Networks works, the Figure 2.1 represent its common architecture. Starting from a 2D input matrix such as an image, the CNN is made up by layers, which each one presents a different number of filter or also called kernels. During the forward pass, it is convolved each filter accross the width and height of the input volume and compute do products between the entries of the filters and the input at any position. As it is slided the filter over the width and height of the input volume, it will be produced a 2-dimensional activation map that gives the responses of the filter at every spatial position. Intuitively, the network will learn filters that activate when they see some type of visual feature. It is also common insert pooling layers in-between successive convolutional layers. Its function is to progressively reduce the amount of parameters and computation of the network using subsampling operations. This subsampling operations take the parameters on a kernel size and perform an operation such as computing the maximum or the mean to reduce the kernel size into a single feature. With this operation a subsampling is performed.

There exist some works which use 2D Convolutional Neural Networks over the frames [12, 38, 3] to extract spatial correlations from video data. As the video also presents temporal correlation, different approaches have been tried to exploit it. The first approach is weighting the frames at the input [37] so information of a chunk of frames is given to the neural network.

Another approach is computing the optical flow to codify temporal correlations and train convolutional neural networks [26, 39] with even two streams [34], one for the frame and the other for the optical flow. These techniques exploit temporal information but they are

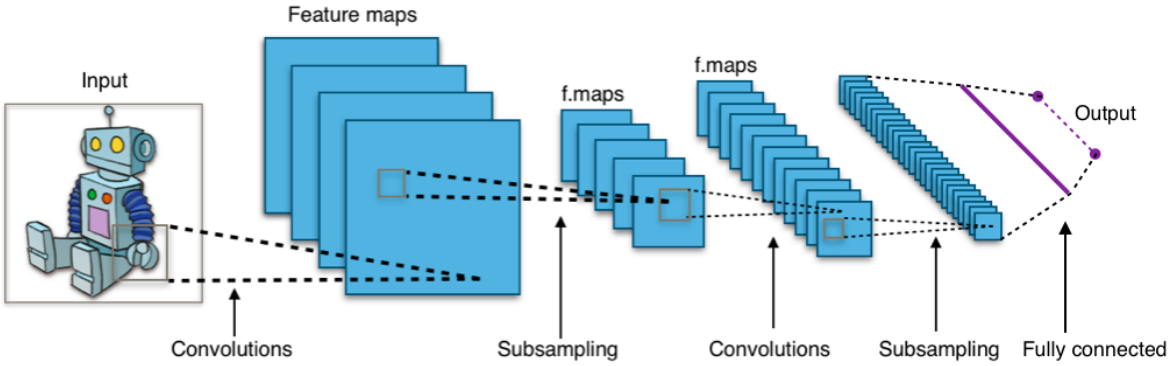



Figure 2.1: Example of the Architecture of a Convolutional Neural Network

limited to the information in a 2D projection, without letting the neural network learning about temporal information.

On the other hand, a recently proposed a CNN tries to exploit both spatial and temporal correlations using exclusively a Convolutional Network. This is the 3D convolutional network, also referenced as C3D [30]. This network uses 3D kernels rather than 2D to extract videos information and try to learn from both the spatial and temporal dimensions. It has been widely used [2, 31, 30, 24] for applications such as video classification. On some other research papers [36, 40] both approaches have been tried using 2D and 3D convolutional networks.

For this C3D network, the input data are clips of videos where all the frames are provided without any previous codification and even it has been proposed a combination of 2D and 3D Convolutional Neural Networks [39, 36]. Also it has been tried to extract motion 3D features from videos to feed as input of the 3D CNN [36].

In addition to spatial and temporal correlations, it has been seen that some approaches extract information from the audio track to explode the information it may give. This technique of combining both video and audio is becoming trendy on the latest activity detection on video datasets [35]. The idea of extracting information from the audio in addition to the video seems really interesting but its success on the results are very related to the coherence between the audio and video tracks.

The recent work presented in [24] is based on Convolutional Neural Networks, more concretely on the C3D network previously explained. It proposes to divide each video in multiple sequences of different lengths and predict for each one whether if they depict an activity or not. In affirmative class, another classification stage determines which of the activities is represented in the video segment. As the first stage, they propose to subsample the segments obtained from the video to put it as input for the C3D. The next stage consist on three parallel networks which are trained separately. All the three networks are trained doing a fine tunning from the original C3D network [30].

## 2.2 Recurrent Neural Networks applied to Videos

A more complex and different approach to address activity classification and temporal localization on videos is using RNNs. This networks are characterized by being trained with sequences of related data and having memory cells. RNNs are mainly used to learn from temporal sequences as text generation or speech recognition, but they have also been used in

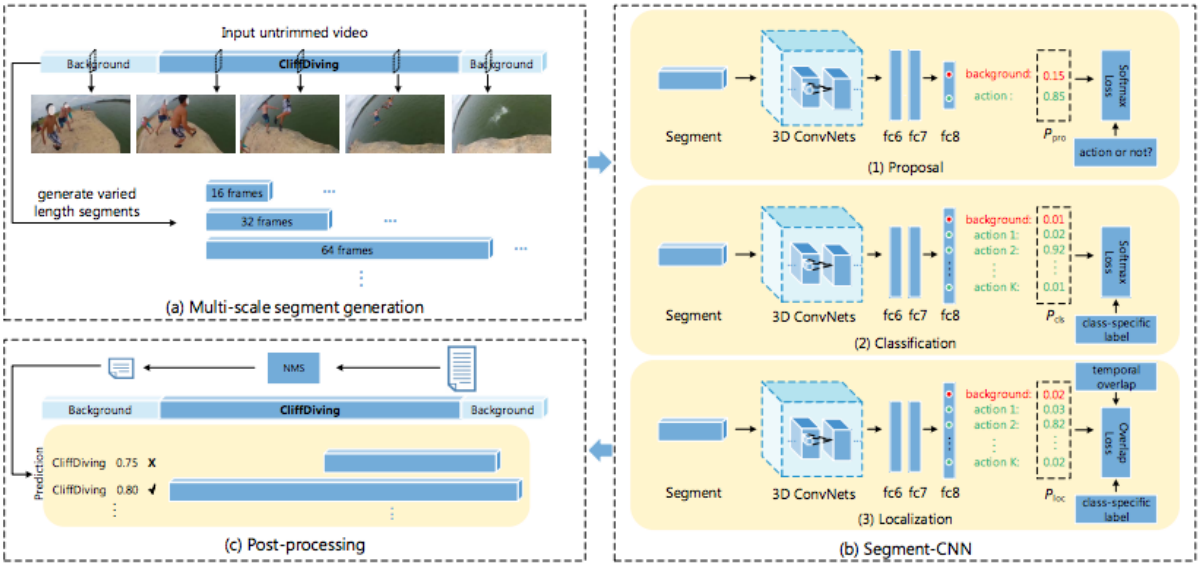



Figure 2.2: Multi-stage architecture to temporal action location using CNNs

tasks related to video.

The most efficient implementation of Recurrent Neural Network found at the literature is the Long Short Term Memory cells [16], or also known as LSTM. This cell, represented in the Figure 2.3, has as input the combination of the previous cell output and the layer's input in addition to a cell state which is propagated at every temporal step and represents the memory of the cell. When forwarding information, the first step is to decide what information is going to be thrown away from the cell state. This decision is made by a *sigmoid* layer called the *forget gate layer*. The next step is to decide what new information is going to be stored in the cell. First a *sigmoid* layer called *input gate layer* decides which values are going to be updated. Next, a tanh layer creates a vector of new candidate values that could be added to the state. As the next step, both will be combined to create an update to the cell state. Finally, the output will be computed based on the cell state, but will be a filtered version. First a sigmoid layer decides what parts of the cell state are going to the output and multiply it by the cell state after a tanh layer. This cells also presents propagation of information along temporal sequences and can be represented forming chain as can also seen on Figure 2.3, where each element represents a single time step.

Most of the literature that uses RNN on video, make their implementation using LSTM cells, but with different approaches. Some works propose to predict activities using LSTMs after features by CNNs [36], while others propose to add an attention model before the recurrent stage to let the network learn where to focus the attention [23, 22]. At [37] for example, a Recurrent Neural Network using LSTMs is used after a 2D CNN to predict the activity on videos returning, from the temporal sequence of input frames, a sequence of activities done at each input frame. Another approach found in the literature is using LSTMs and reinforcement learning to temporally localize activities on videos [38] achieving state of the art performance. This achieves very good results without the requirement to have a prediction after seeing all the video frames because with little iterations, the network predicts where the activity is temporally located.

In addition to previous exposed LSTMs, there is a little variation of it which is a cell called Gated Recurrent Unit (GRU) [7]. This is a slightly variation of the LSTM which combines

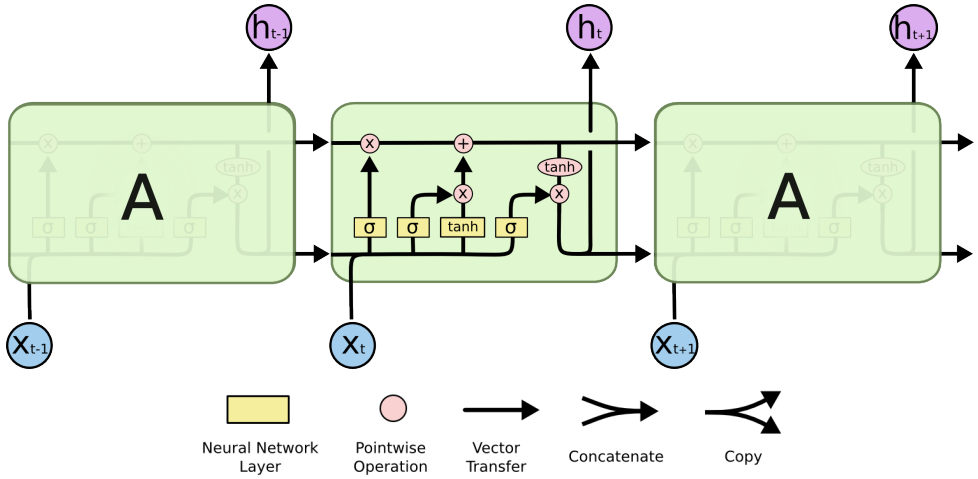



Figure 2.3: Example of the Architecture of a Convolutional Neural Network

the forget and input gate into an update gate and has been applied in the task of video classification [3] in conjunction with CNNs.

# Chapter 3

# Methodology

This chapter describes the required steps to achieve a functional deep neural network capable of detecting and classifying human activities in untrimmed videos.

## 3.1 Objective

The aim of this project is to design and train a deep neural network to classify and temporally localize activities in videos. The network is designed to exploit temporal information by combining Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). Short-term temporal information is captured by the 3D convolutional features computed over short video clips of 16 frames. These features are later fed into a RNN that aims at recognizing longer-term temporal patterns. The RNN is expected to output a label for each clip of 16 video frames, preserving temporal consistency between close predictions in the output sequence. The design of the network enables both activity classification and detection capabilities using simple post-processing techniques on its output.

## 3.2 ActivityNet Dataset

Our model was trained and tested with the ActivityNet Dataset [10], a large-scale video benchmark for human activity understanding. This dataset (on its version 1.3, used in our work), contains 19,994 videos with different 200 activities labeled, representing a wide range of human activities. While this large dataset allows training the millions of parameters that define a neural network, it also poses important computational challenges due to the high performance computing and storage it requires.

ActivityNet 1.3 contains 660 hours of video, which are split in the following way: 50% training set, 25% validation set and 25% testing set. Each video of the dataset is annotated with a single activity, along with all temporal locations in which the activity occurs. Figure 3.1 presents examples of videos of different activities and its temporal annotations.

The dataset is provided as a description text file, with the URL of the video and also the ground truth annotations, as the starting time, ending time and the class of activity in the given interval. In addition, the original video resolution and the duration are also provided.

Because all the videos from this dataset are hosted on *YouTube*, only their links are provided to avoid copyright issues. For this reason, the first and costly task was downloading all


Figure 3.1: Sample of videos from the ActivityNet Dataset

the videos from their URLs. This was done using a library called *youtube-dl*[1], which allows downloading YouTube videos with Python calls. Some issues arose during the acquisition of the dataset, such as videos being removed by their owners, or being blocked due to localization restrictions. In those cases, videos could not be downloaded and therefore were not used in the experiments. The total size of the used dataset was of 19,811 videos.

Once the videos from the dataset were downloaded, the number of frames of each video was also computed to be able to convert from seconds to frames in the temporal domain. In addition, other stats were computed, such as the whole number of frames from all the videos in the dataset, which is 65.6 million frames. Also, the length in minutes of each activity at the dataset was plot on Figure 3.2, to have an estimation about the activities duration. As it can be observed, not all the activities have the same duration along the dataset, varying from 40 minutes as the total activity duration to 3.5 hours for the longest activity. In total, over all the dataset there are 313 hours of activities which needed to be detected and classified in the 660 hours of video.

## 3.3 Extraction of Video Features Using C3D

The C3D network [30] was chosen as a feature extractor, due to its good performance in previous works [2, 31, 30, 24]. This network is composed of 8 convolutional layers, plus 5 pooling layers, 2 fully-connected layers and a Softmax output at the end. The convolutional layers have $3 \times 3 \times 3$[2] kernels and stride 1 while at the same time the pooling layers compute the maximum at every kernel size of $2 \times 2 \times 2$ (except from the first pooling layer which has

[1] https://rg3.github.io/youtube-dl/

[2] For notation, the dimension ordering is $d \times k \times k$ where $d$ is temporal dimension and $k$ is spatial dimension

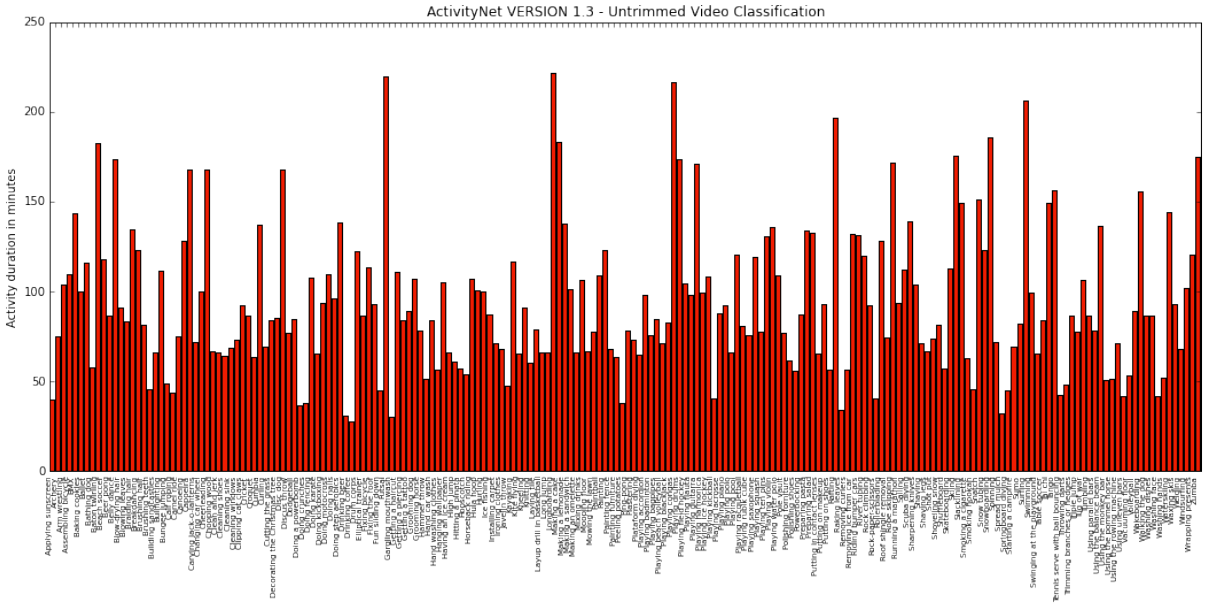



Figure 3.2: Activity duration in minutes for each activity in the dataset

a $1 \times 2 \times 2$ kernel size). The two full-connected layers (*fc6* and *fc7*) have both 4096 neurons while the Softmax output has 200 outputs as the number of classes of the Sports1M Dataset. Figure 3.3 shows a representation of the C3D architecture, including the number of kernels or neurons of each convolutional layer and fully-connected layer respectively.

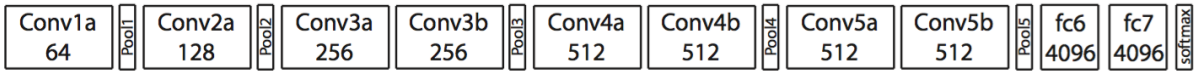



Figure 3.3: The C3D Architecture

The input of the network are clips of videos of 112x112 frames width and height and 16 frames of temporal depth. It was decided to extract the video features at the fully-connected layer *fc6* because it has been reported to work well for both image classification tasks [11] and input feature for training a RNN [9].

The C3D model [30] was developed over a version of *Caffe* [18] dated in 2014. This version did not support the *Python* environment used for this project, nor provides tools for implementing RNNs. Given the relevance of RNNs in this work, it was decided to move to the *Keras* framework, which provided the required tools. As a consequence, the C3D model in Caffe trained with the Sports1M [19] dataset, was ported to *Keras*. This was an unexpected contribution of this thesis, which was warmly received by the community of Keras developers. Its main contributor, François Chollet, tweeted about our ported model. All the process has been open sourced and is available online[3].

At the time to run the C3D model on the model ported to *Keras*, some small changes were applied Keras' source code[4], involving the implementation of the 3D convolution and

[3] https://gist.github.com/albertomontesg/d8b21a179c1e6cca0480ebdf292c34d2

[4] The fork of *Keras* used can be found in: https://github.com/albertomontesg/keras/tree/improved-3d-ops

3D pooling operations. Once this was fixed, all the videos were forwarded through the C3D model, and the features from the first fully connected layer *fc6* were extracted. This process used 2 GPUs in parallel for feature extraction, and multiple CPU cores fetching all the videos from disk. The usage of 2 GPUs in parallel reduced the computational time expected for this task from 1 week to 2 days.

In the process of fetching the videos from disk, the *OpenCV* [6] library was used. With this software, the videos were read in chunks of 16 frames, resized to 112x112 as input frame size, remove the mean for each color channel which the original model was trained with and then forwarded through the C3D network. The values of the model at the full-connected layer fc6, a sequence of 4096-sized vectors was stored in disk for preparing it to train the Recurrent Neural Network.

## 3.4 Extraction of Audio Features

Audio features were also explored as potential sources of information to recognize the activity depicted in the videos. As previously explored for activity detection [35], some audio descriptors were extracted to improve the detection and classification results. In this project, the MFCC and Spectral descriptors were adopted [15].

Both MFCC and Spectral descriptors were extracted using specialized software: SPro [13] and Essentia [5] respectively. For the MFCC descriptors, 40 values were extracted for 10ms temporal windows. Because the misalignment of scale between the features extracted for video and MFCC videos, the second ones were grouped to have the same sequence length of audio features and video features.

In order not to lose too much information when grouping the features, the mean and standard deviations along the temporal scale were computed, so at the end 80 values were available to represent the MFCC features from the audio video's track.

On the other hand, the Spectral features were extracted to have information about the energy distribution on the frequency domain along the video, so only 8 values were extracted for each videos. At the time to give the audio features at the input of the Recurrent Neural Network, for each video clip feature vector, it was concatenated the MFCC features grouped for the temporal window of the clip and then also concatenated the spectral features of the whole video the clip belong.

## 3.5 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) were adopted as the basic module to solve the main task of this project. Among these family of deep learning architectures, the Long-Term Short Memory networks (LSTMs) were adopted thanks to their ability to learn long term correlations.

The basic architecture consist in a single layer of LSTM with 512 cells, which is defined with a total of 9.5 millions parameters. A more advanced architecture would consider two layers of LSTMs, as shown in Figure 3.4.

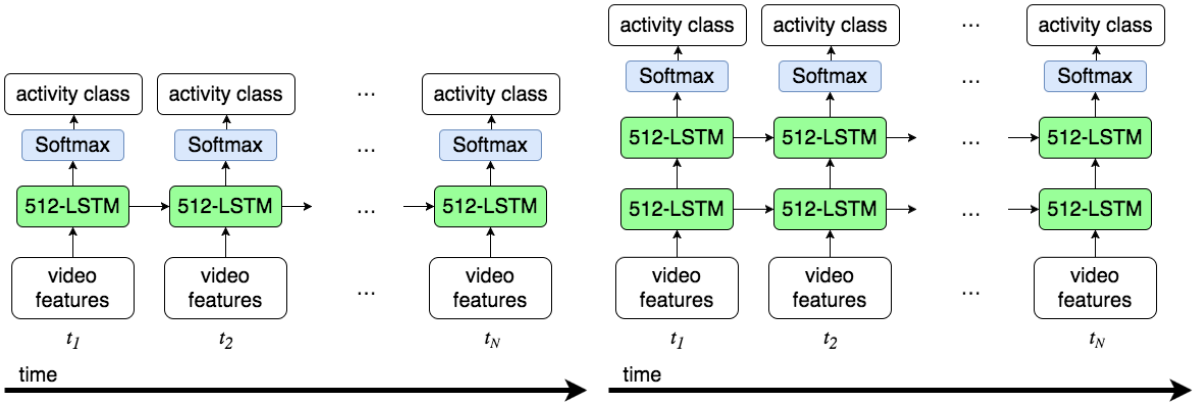



Figure 3.4: Architecture of RNN used with 512-LSTMs with one and two layers respectively.

During the development of this work, it was observed that the single layer architecture provided very different classes in short periods of time, providing this way very inconsistent results. For this reason, a more sophisticated architecture was introduced, which inserted as an additional input the activity predicted for the previous clip. Figure 3.5 presents this advanced architecture. This new input with the information about the previous activity was fuse in a concatenation operation with the already extracted features vector and then used to train the Recurrent Neural Network. This kind of feedback on the Network was expected to give smoother predictions, as the network is aware about the previous activity.

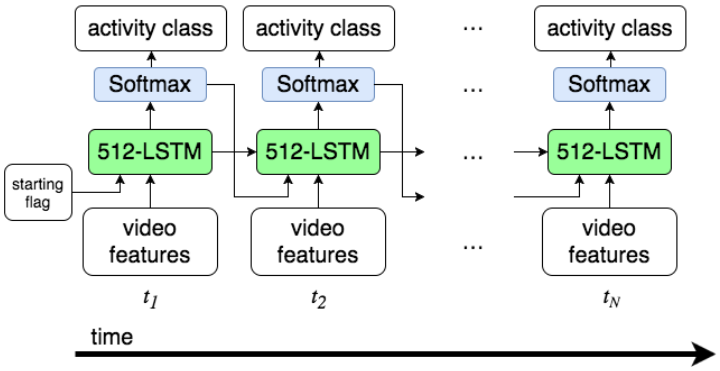



Figure 3.5: Architecture of the RNN network with feedback from the previous output

## 3.6 Data Preparation

While training of neural networks, both the input and output data are grouped in batches to perform parallel and faster computations of the learning algorithm. This batches have all the same number of fixed length sequences of both inputs and outputs vectors of the dataset. The length of all the sequences given at each batch is called *timestep* and must be a low value so that the gradient propagates along all the sequence. Otherwise the gradient might achieve very high values, a problem known as *exploiting gradient*.

Most video datasets include content of different lengths, which forces a pre-processing to build training batches of a fixed length. When training Recurrent Neural Networks, the internal state (which represent the memory stored) of the LSTMs is preserved for every batch

computation but then reset between batches. As the sequences of features vectors extracted from videos in the dataset are longer the than the *timestep* used, during training the memory state of the LSTMs will be reset multiple times (depends on the video length).

There exist two solutions to this problem. The first solution is organizing data in batches of a large *timestep*, but this would require to clip the gradients, as explained in [21]. The second solution is to train the RNN without resetting the memory from batch to batch. With this approach, if a video sequence (Equations 3.1 and 3.2) is longer than the *timestep*, the RNN can be trained by passing fragments of the videos as clips, one after the other in different batches, but at the same batch position. Since the memory is not reset after each batch, the video sequence is processed smoothly.

$$\bar{X} = [x_i, x_2, \ldots, x_{4096}] \tag{3.1}$$

$$V_i = \{\bar{X}_t\}_{t=1}^{T_i} \tag{3.2}$$

For this configuration, the different videos must be carefully set on the same batch index to exploit the *Keras* functionality of *stateful* training for RNNs. Figure 3.6 depicts how the video features are organized to train the Recurrent Neural Network. The notation for the data is as follows: $\bar{X}$ is the feature vector of a 16 frame clip extracted from the C3D network, and $V_i$ the sequence of features for a single video. Note that $T_i$ is the length of the sequence of each video $i$ in number of 16-frame clips.

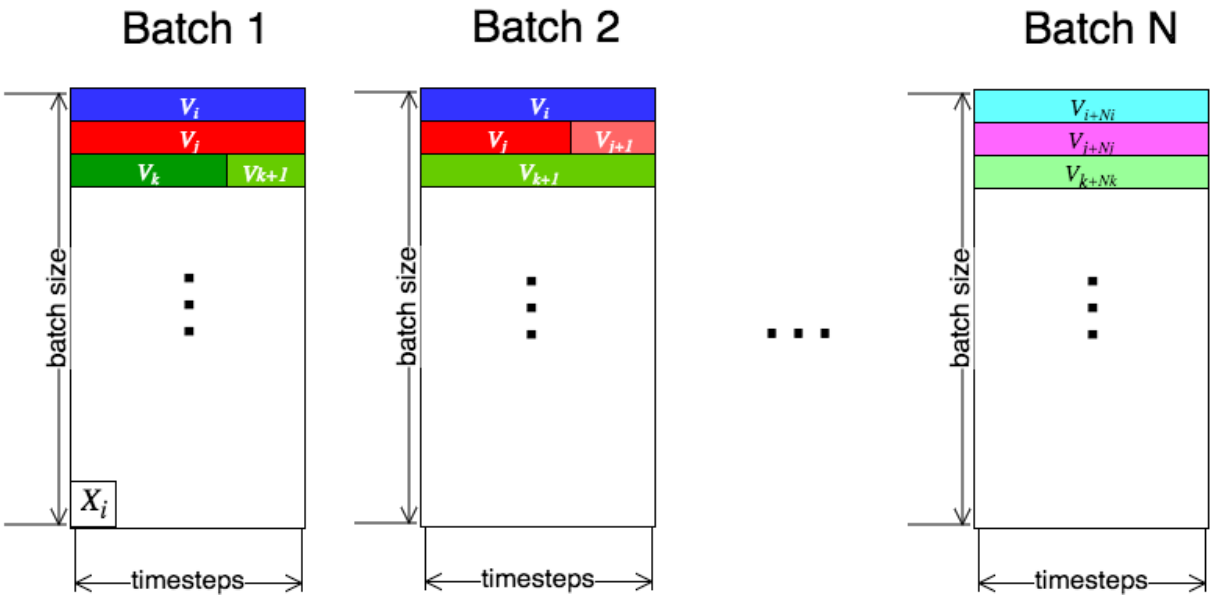



Figure 3.6: Graphical representation of how the data is sorted in batches

As the last step to prepare the data, the misalignment between the blocks of 16 frames and the detailed annotation provided by the ground truth had to be considered. During our training, if the Intersection Over Union (IOU) between the 16-frame clips and the ground truth annotations was higher than 0.5, the clip was considered as *activity*. On the other hand, if the IOU was below 0.5, the clip was considered as *background (no activity)*.

So at the training and validation subsets, the output was encoded with the one-hot encoding, which puts a 1 at the position of the class given and 0s on the rest. This way to encode the

output is the used for Softmax output layers which represent the probability of each class to be predicted as the output is between 0 and 1. The same codification was used on the network which feedback the previous output, but an additional class was added. For the first element of each video, as they do not have a previous output, a $<START>$ class flag was given.

## 3.7 Training Methodology

During training, each batch is forwarded to the Neural Net and with the output predicted and the ground truth given, the loss is computed. The objective of every learning step is to reduce the loss, so using a specific gradient computation, the parameters of the network are updated in proportion of the computed gradient and a value previously set called the learning rate. As the training goes on, it is expected and required that the loss value after every batch decreases for both training and validation subsets. The first subset is the one used to update the model's parameters while the second one is used to test the behavior of the network while training.

It is very common that if the learning rate is not correctly set, the loss for the training set (the one the network learns from) goes down, while it increases for the validation set. This will mean that the network has only learned for the subset seen but the prediction over the rest of the dataset are not valid. This effect, which can be explained if the network has very high learning capacities, is called over-fitting and is undesirable. Also while training, the accuracy is tend to be computed to check that the accuracy of the prediction increases.

The training of RNNs allows learning the parameters that govern their behavior. The training process itself is also determined by certain design decisions. Firstly, the loss function, also known as objective function, allows assessing the performance of the trained network by comparing its prediction over the training set. As the last layer used was a Softmax which is non-linear function that gives values between 0 and 1, the output can be understand as the probability of predicting each of the classes at the output. Having at the output a distribution of probability, the common computation of the loss is using the *categorical cross entropy*.

The *categorical cross entropy* function is described in Equation 3.3 and represents the average number of bits needed to identify an event drawn from the set, if a coding scheme is used that is optimized for the predicted probability distribution $q$, rather than the ground truth probability distribution $p$. This function is higher as more different the predicted distribution and the ground truth are.

$$H(p,q) = -\sum_{x} p(x)\log(q(x)) \tag{3.3}$$

Secondly, the learning rate controls how quickly the parameters of the model are updated. Figure 3.7 shows the learning curves of the same network with a different values of learning rate. On this curves the loss (blue) and the accuracy (red) are plot over the epochs, and also separated by subset: training (thin line) and validation (striped line). Note that an epoch is the number of iterations required to have forwarded all the batches of the dataset once. In our case, the best learning rate was $10^{-5}$.

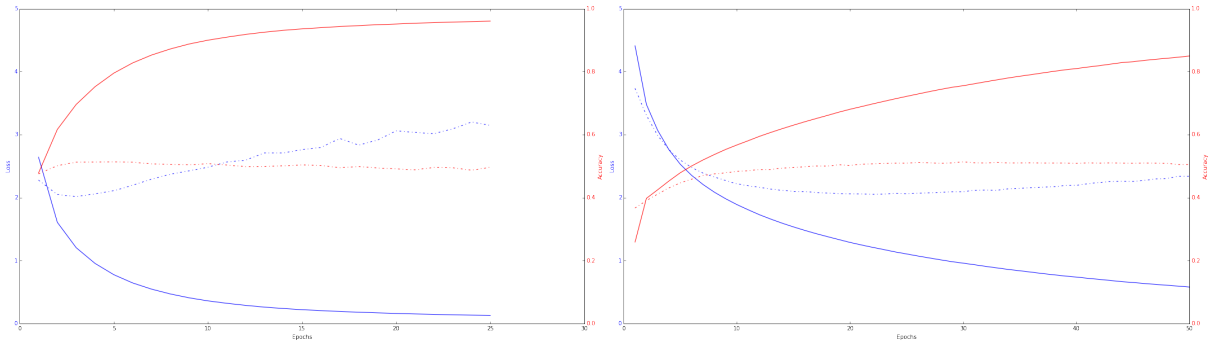


Figure 3.7: Comparison of the learning curves. On the left with a learning of $10^{-4}$ and at the right figure with a learning rate of $10^{-5}$

It was observed that the network presented some over-fitting due mostly to the high learning capacity the network has over the data. This over-fitting problem can be addressed by adding dropout [28] layers before and after the LSTM layers. Dropout consists in randomly setting a fraction $p$ of input units to 0 at each update during training time.

The combination of visual and audio features required a normalization step, as they presented different statistics and range of values. So for all the experiments, a batch normalization [17] was included after the input data in order to have all the inputs with the same statistics, mean 0 and standard deviation equal to 1.

The last problem that was necessary to face, was related to an unbalanced distribution of classes in the dataset. All the activities had approximately the same frequency of appearance, but nearly half of the video clips correspond to a non-activity, or as previously defined, background class. Given the used categorical cross entropy as loss function, the network tended to predict the background class. For this reason, it was decided to weight the loss function in order to slow down the learning of weights when the background class was predicted.

The weighted loss function taking into account the unbalanced classes is defined as

$$H(p,q) = -\sum_{x} \alpha(x) p(x) \log(q(x)), \text{ where } \alpha(x) = \begin{cases} \rho, & x = \text{background instance} \\ 1, & \text{otherwise} \end{cases} \tag{3.4}$$

where $\rho$ is the factor used to weight the loss for background instances at training. This value $\rho$ is usually computed as 1 minus the frequency of appearance of the class (for all the activities class can be simplified to 1).

## 3.8 Post-Processing

The proposed network generates as an output the predicted activity probability for each 16-frames clip. These clip-wise predictions were post-processed to solve the project goals of activity classification and detection.

### 3.8.1 Classification Task

For the classification task, the first step was removing the output probability corresponding to the background class, as it is known that all videos in the ActivityNet Dataset depict one,

and only one, activity. Once the background class was removed, then the mean of each video activity class was computed along the video's output sequence.

$$p_{video}(x) = \frac{1}{T_{video}} \sum_{i}^{video} p_i(x), \text{where } x \in \{\text{Dataset Activities}\} \tag{3.5}$$

With this operation, each video in the dataset is associated to a vector of probabilities, each one giving the probability of each activity happening along the video. The classification task was solved by selecting the activity with the highest probability.

### 3.8.2 Detection Task

For the detection task of the ActivityNet Challenge, also referred as temporal activity localization, more operations were required. We exploited the fact that each video in ActivityNet dataset is annotated with a single activity class, which may occur at multiple temporal segments.

The sequence probabilities predicted by the RNN were filtered with the *mean* operator of size $k$, being $k$ the size of the window backwards and forward. Equation 3.6 defines this filter. This filtering step generated a smoother output prediction.

$$\tilde{p}_i(x) = \frac{1}{2k} \sum_{j=i-k}^{i+k} p_i(x) \tag{3.6}$$

The next step was for every step of the output sequence, compute the activity probability as the sum of all the output classes except the background class.

$$\tilde{p}_i^a = \sum_{x=1}^{200} \tilde{p}_i(x), \text{where } x = \begin{cases} 0, & \text{background class} \\ i, & 1 \leq i \leq 200 \text{ activity classes} \end{cases} \tag{3.7}$$

Finally, only those activity detection with $\tilde{p}_i^a$ over a certain probability threshold $\gamma$ were considered. The rest were discarded as they did not meet a minimum of confidence. Both parameters $k$ and $\gamma$ were learned also during the training process described in Section 4

# Chapter 4

# Results

This section describes the results of the experiments on the ActivityNet dataset obtained with the different variations of our architecture.

## 4.1 Evaluation Metric

In tasks like classification and detection, the research community and also the ActivityNet Challenge uses the same metrics for evaluation to be able to compare results between publications. In our work, we have computed our results using the evaluation scripts provided by the ActivityNet Challenge 2016 organization.

### 4.1.1 Classification Metric

The classification of videos was assessed with the mean Average Precision (mAP). This is computed as the mean of the Average Precision (AP) of all $M$ classes at the Dataset.

$$mAP = \frac{1}{M}\sum_{m=1}^{M} AP(m) \tag{4.1}$$

At the same time, the Average Precision for each class is computed based on a ranking of the classes obtained for each video. A classifier typically predicts a confidence score for each of the considered classes, and these scores allow generating these rankings from the most to the less likely class to be depicted by the video. The precision $P$ is the inverse of the position where the correct class appears, and the AP for class $m$ is obtained by averaging all the precision of the videos associated to the class, according to the ground truth.

$$AP(m) = \frac{1}{K_m}\sum_{n=1}^{k_m} P(n)\cdot rel(n), \text{where } rel(n) = \begin{cases} rel(n) = 1 \Leftrightarrow n \in K_m \\ rel(n) = 0 \Leftrightarrow n \notin K_m \end{cases} \tag{4.2}$$

As an alterative metric, the *Hit*@3 was also considered. In this case the top 3 ranked classes are considered for each video, considering a good prediction if the correct class is among them. The *Hit*@3 metric simply measures the proportion of videos in the dataset that are correctly predicted under this assumption. This metric has become popular as adopted in the ImageNet ILSRVC image classification challenge.

### 4.1.2 Detection Metric

For the other task of the ActivityNet Challenge, the temporal localization or detection, the metric used to verify the results given is the Mean Average Prediction score over all activity classes. To do this, a detection is determined to be a true positive according to the following procedure:

1. It is computed the overlap, measured by the Intersection Over Union (IoU) score, between a predicted temporal segment and a ground truth segment.
2. It is marked the detection as positive if the overlap is grater than a threshold $\alpha$.

For the ActivityNet Challenge $\alpha$ value used is 0.5.

## 4.2 Training

The Recurrent Neural Networks were trained with a 256 batch size and 20 *timesteps* to get a good gradient propagation through the *timesteps* and to fit the most possible data into the GPUs when training.

The optimizer function used was the RMSprop [8] which is known to work very well for training Recurrent Neural Networks. The RMSprop was set with a value of $\rho$ of 0.9 and $\epsilon$ of $10^{-8}$. The value of the learning rate, as it is explained on Section 3.7, was set to $10^{-5}$ for all the experimentation done.

At the weighted loss computation, for the background class was found that its frequency of appearance was 0.4. Different values of $\rho$ were finally tested trying to obtain the best results and finally the $\rho$ was set to 0.3.

Finally the dropout was set to have a probability of 0.5 as it is the most frequent value used at the literature [28].

## 4.3 Classification Task

The first experiments were related to the depth of the network. It was tested from a 3 layers LSTM with 1024 cells each layer, to a one single layer with 512 cells, going through a two layers network with 512 LSTM cells each one. Also all the experiments were done exclusively with the features extracted from the C3D network. Table 4.1 shows how the simplest RNN achieved the best performance. This results indicate that all the networks presented high learning capacity over our data and it was observed a little over-fitting with the deeper architectures.

| Architecture | mAP | Hit@3 |
|---|---|---|
| 3 x 1024-LSTM | 0.5635 | 0.7437 |
| 2 x 512-LSTM | 0.5492 | 0.7364 |
| 1 x 512-LSTM | **0.5938** | **0.7576** |

Table 4.1: Results for classification task comparing different deep architectures. All values with only video features on the validation dataset.

Once the single layer architecture was selected, the network was trained comparing the input features used and also comparing the basic architecture with the one with feedback.

Table 4.2 shows how the basic architecture with only video features obtain the best results. The fail of the feedback architecture may be to the discrepancies of the data use at training and testing. At training, as an input of the previous output it has been use the ground truth, while at testing it has been used the predicted output. The difference between the nature of the data can yield to errors and lower performance [4].

Regarding the audio, doing some manual exploration over the videos and its audio tracks, it was observed that in some cases, the audio was completely unrelated, mostly with music that had no relationship with the activity depicted by the video.

| **Features used** | **mAP** | **Hit@3** |
|---|---|---|
| Only video | **0.5938** | **0.7576** |
| Video w/ audio | 0.5755 | 0.7352 |
| Only video & feedback | 0.5210 | 0.6982 |
| Video w/ audio & feedback | 0.5652 | 0.7319 |

Table 4.2: Results for classification task with the model made by one 512-LSTM. Compare between features and feedback on the validation dataset.

The ActivityNet Dataset is organized in 200 activities, and these activities are also contained in a taxonomy, in particular, at the leaves of a hierarchical structure. On this structure there are 5 top level categories that describe higher levels of abstraction: *Eating and drinking activities*, *Sports, Exercise and Recreation*, *Household Activities*, *Socializing, Relaxing and Leisure*, *Personal Care*. The mean average precision was also computed for these 5 top level activities. As can be seen on Table 4.3 and Figure 4.3 the *Sports, Exercise and Recreation* have an average precision of 94%. This can be explained with the fact that all the features extracted from the videos come from a network which weights were trained to work with a dataset of videos, the Sports1M [19].

| **Global Activities** | **AP** |
|---|---|
| Eating and drinking Activities | 0.56942 |
| Sports, Exercise, and Recreation | 0.93662 |
| Household Activities | 0.74177 |
| Socializing, Relaxing, and Leisure | 0.76494 |
| Personal Care | 0.59931 |
| **Global** (mAP) | 0.72241 |

Table 4.3: Mean Average Precision computed for the top level activities of the ActivityNet Dataset. The results are computed over the validation dataset.

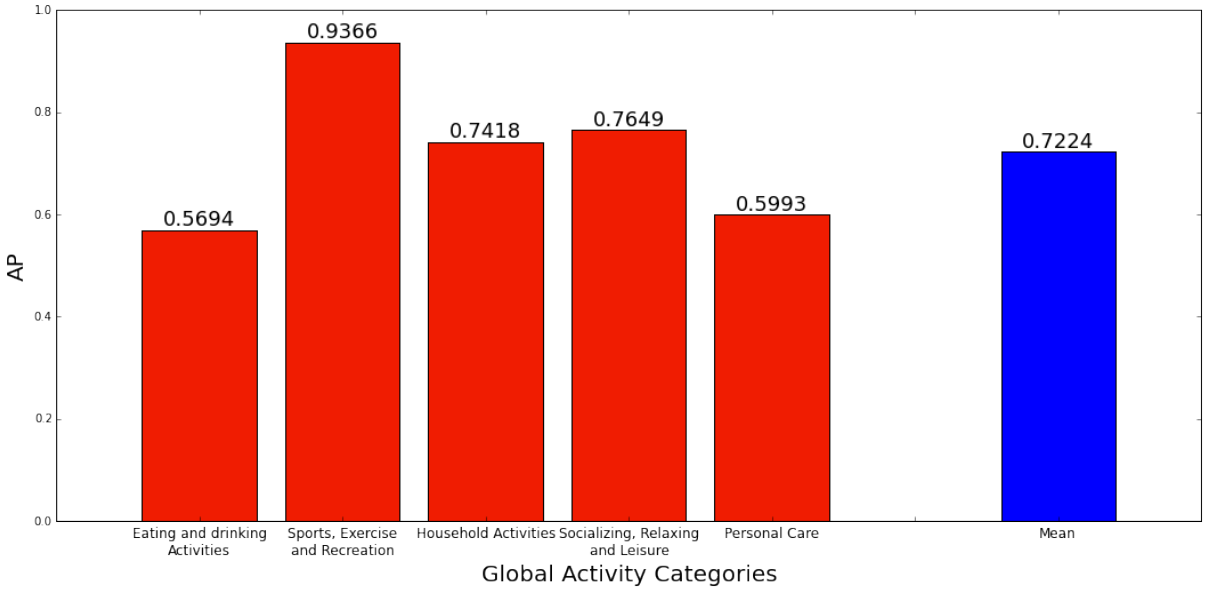



Figure 4.1: Representation of the results over the top-level activities

In addition to the previous figure, and expanding the Average Precision computation to all the activities categories of the Dataset, Figure 4.6 plots the sorted APs of all the activities. This plots shows that there is lot of variability in the precision depending on the activity. Taking a closer look, the activities related to sports and require a lot of movement present higher performance in classification. On the other hand, activities more relaxed, which no action happens, tend to have lowest performance.

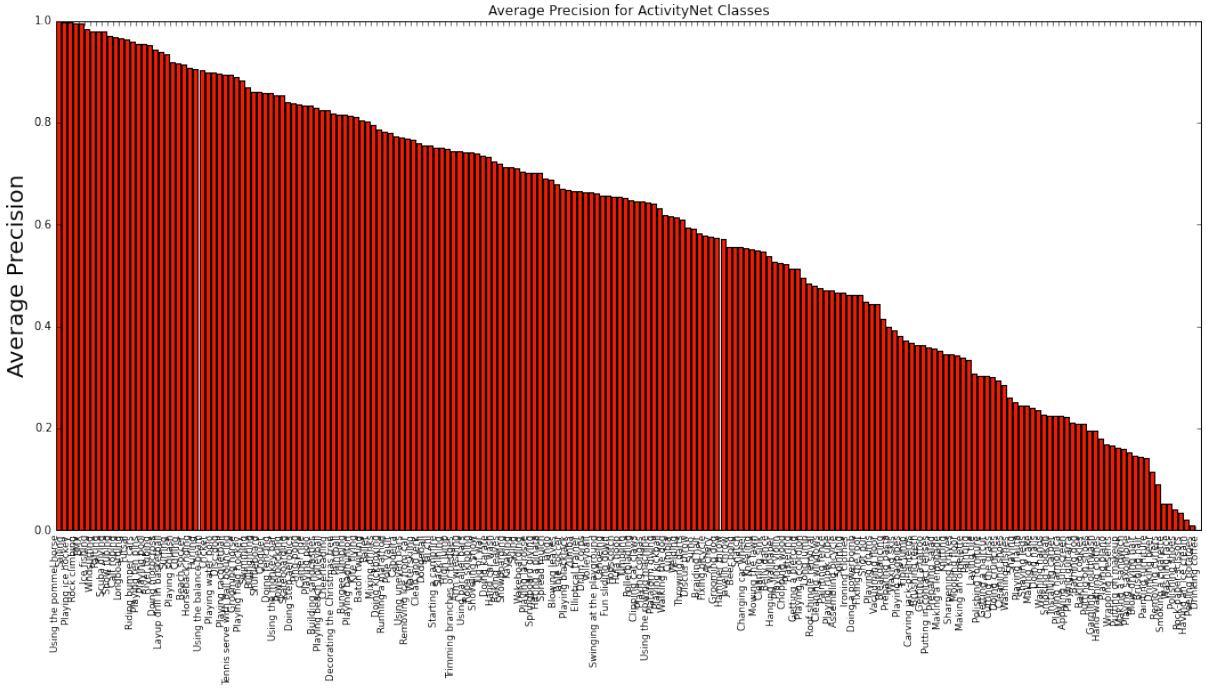



Figure 4.2: Average Precision of all the activities of the dataset for the classification task

This fact can be perfectly explained with feature extraction used from the videos. It was used a CNN specifically trained for sports videos which always present a lot of action and movement, and this fact may have affected the performance in our network on activities that not present many movement over their video frames.

At the time of submission to the ActivityNet Challenge 2016, the basic architecture was chosen by using the video features only. The results in both validation and test subset are shown in Table 4.4.

| **Subset** | **mAP** | **Hit@3** |
|---|---|---|
| Validation (UPC Team) | 0.5938 | 0.7576 |
| Test (UPC Team) | 0.5874 | 0.7554 |
| Baseline [10] | 0.4220 | - |
| Mean Performance from Challenge Participants | 0.6626 | 0.7706 |
| Challenge Winner [33, 32] | 0.9323 | 0.9642 |

Table 4.4: Results and comparison of the UPC team at the ActivityNet Challenge 2016 for the classification task. The third-party results are all referred to the test subset.

As an extra visualization of the results for the classification task, at the Figure A.1 on the Appendix it is attached the confusion matrix of the top-1 activity prediction with the ground truth.

## 4.4 Detection Task

The main goal of this project is to temporally localize activities on videos. The results obtained are presented in Table 4.5, considering two architectures: the basic architecture with video features, and the feedback architecture with video and audio features concatenated (this concatenation helped for the classification task, as shown in Table 4.2). The basic architecture obtains a slightly better results.

| **Architecture** | **mAP** |
|---|---|
| Basic Architecture | **0.22513** |
| Feedback Architecture | 0.20676 |

Table 4.5: Best results obtain for the temporal activity localization task in the two architectures

The detection task included as post-processing stage described in Section 3.8, which aimed at a smoother and better temporal prediction of the activities. The parameters of this post-processing where also jointly estimated, as shown in Table 4.6. These were $\gamma$ as the activity probability threshold, and $k$ as the window size for the mean filter.

| $\gamma$ | $k = 0$ | $k = 5$ | $k = 10$ |
|---|---|---|---|
| **0.2** | 0.20732 | **0.22513** | 0.22136 |
| **0.3** | 0.19854 | 0.22077 | 0.22100 |
| **0.5** | 0.19035 | 0.21937 | 0.21302 |

Table 4.6: mAP with an IOU threshold of 0.5 over validation dataset. Here there is a comparison between values of $k$ and $\gamma$ on post processing.

As it can be seen, the best performance was achieved with an activity threshold $\gamma = 0.2$ and smoothing filter of $k = 5$. The effect of the both of the operations performed after the prediction can be seen in Figures 4.3 and 4.4. On both figures the temporally location of the activity at the ground truth and at the prediction are displayed, before and after of each of the post-processing.

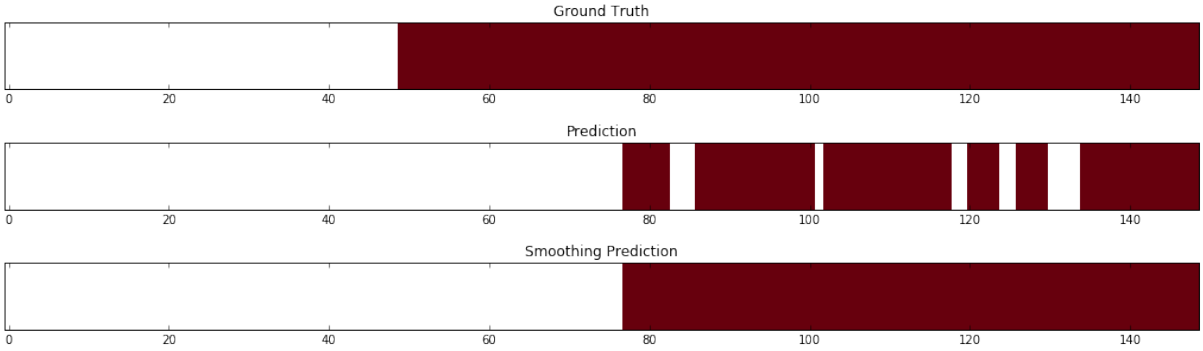



Figure 4.3: Effect of the mean filter with $k = 5$ achieving a smoother activity prediction.

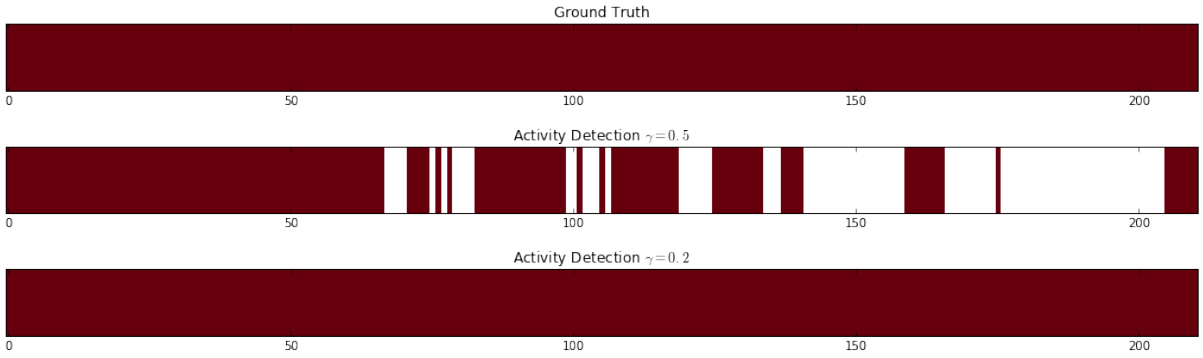



Figure 4.4: Activity Localization using different values of the threshold $\gamma$

In addition, as it has been done for the classification task, the Average Precision has been computed for the top level activities of the ActivityNet Dataset taxonomy. As can be seen on Table 4.7 and Figure 4.5, all the top level activities present a similar precision in activity temporal localization except from the category *Personal Care*, which does not achieve half of the precision of the rest of top level categories. Also notice that the top level category with highest precision is *Sports, Exercise and Recreation*, similarly as in the classification task.

| Global Activities | AP |
|---|---|
| Eating and drinking Activities | 0.25582 |
| Sports, Exercise, and Recreation | 0.30023 |
| Household Activities | 0.26252 |
| Socializing, Relaxing, and Leisure | 0.26060 |
| Personal Care | 0.11234 |
| **Global** (mAP) | 0.23830 |

Table 4.7: Average Precision of activity localization computed for the top level activities of the ActivityNet Dataset. The results are computed over the validation dataset and with a IoU threshold of 0.5.

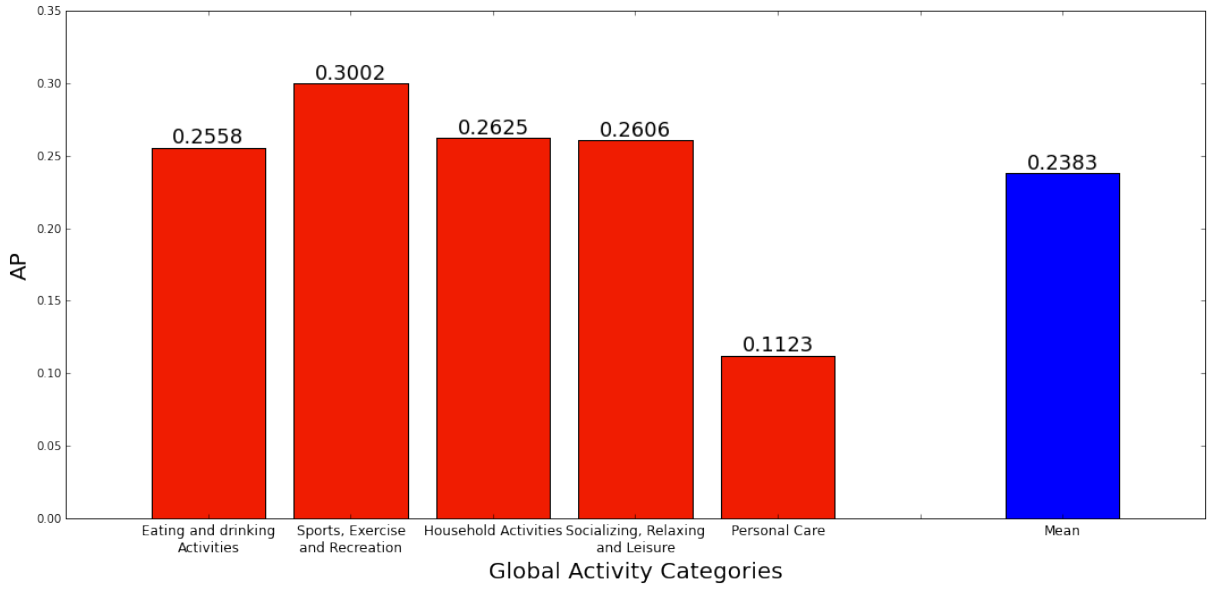



Figure 4.5: Representation of the results over the top-level activities

As it has been done for the classification task, in the Figure 4.6 the Average Precision has been plotted for every activity of the dataset. The same effect in the classification task is observe for detection task, where the average precision not performs equally along the activities on the dataset and the more active activities tent to have better results than the more relaxed ones.

It is very common on tasks where the precision is attached to the Intersection over Union computation, to perform the computation for different values of the $\alpha$ threshold of the IoU. On Figure 4.7 is plotted the mAP for a rank between 0.1 and 0.5 of the IoU threshold $\alpha$. It can be observed also observed how decreasing the IoU threshold, the mAP increases achieving a value of 0.35 when the IoU is set to 0.1.

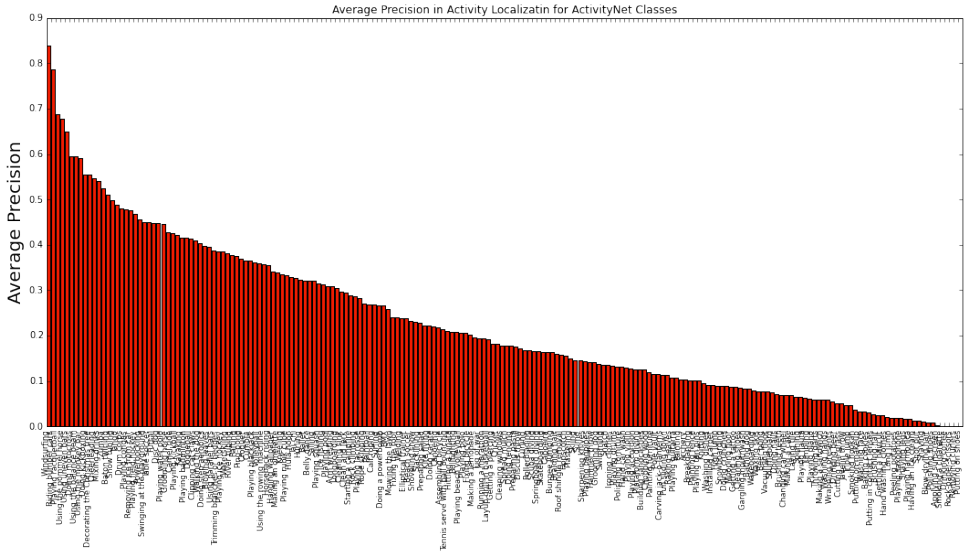



Figure 4.6: Average Precision of all the activities of the dataset for the detection task

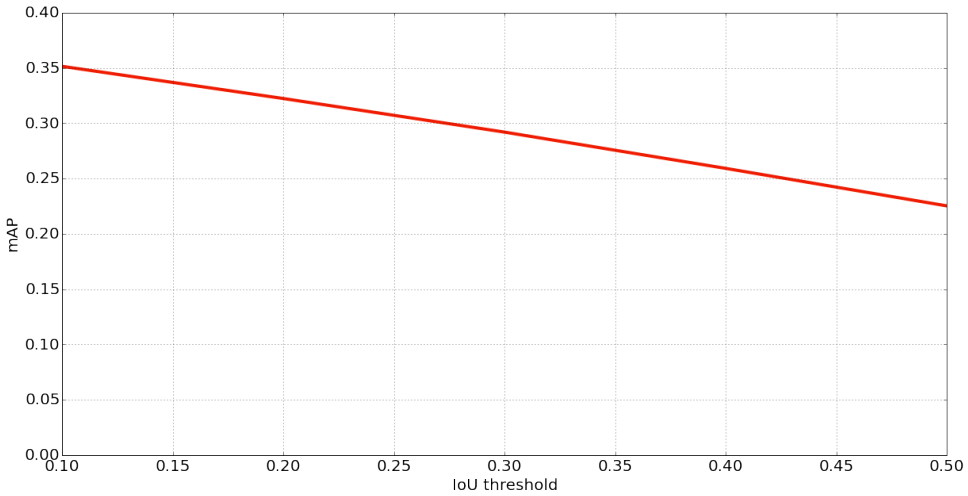



Figure 4.7: mAP computed against different values of the IoU threshold $\alpha$.

Another application that this work's proposed Neural Network can do is propose temporal regions where activities might be happening. This application is called temporal proposals. Giving only the region where an activity might be happening on the video can be further combined with other Neural Networks in parallel to improve the results. It has been set very clear that the aim of this project is to achieve the classification and temporally localization of activities in a very simple and flexible approach using one single network.

This regions proposals can be computed after the the post-processing where the activity probability is computed and setting the activity threshold at the same value where detection show the best performance. The performance at this application is plot on Figure 4.8 where the Recall is plot against the IoU threshold.

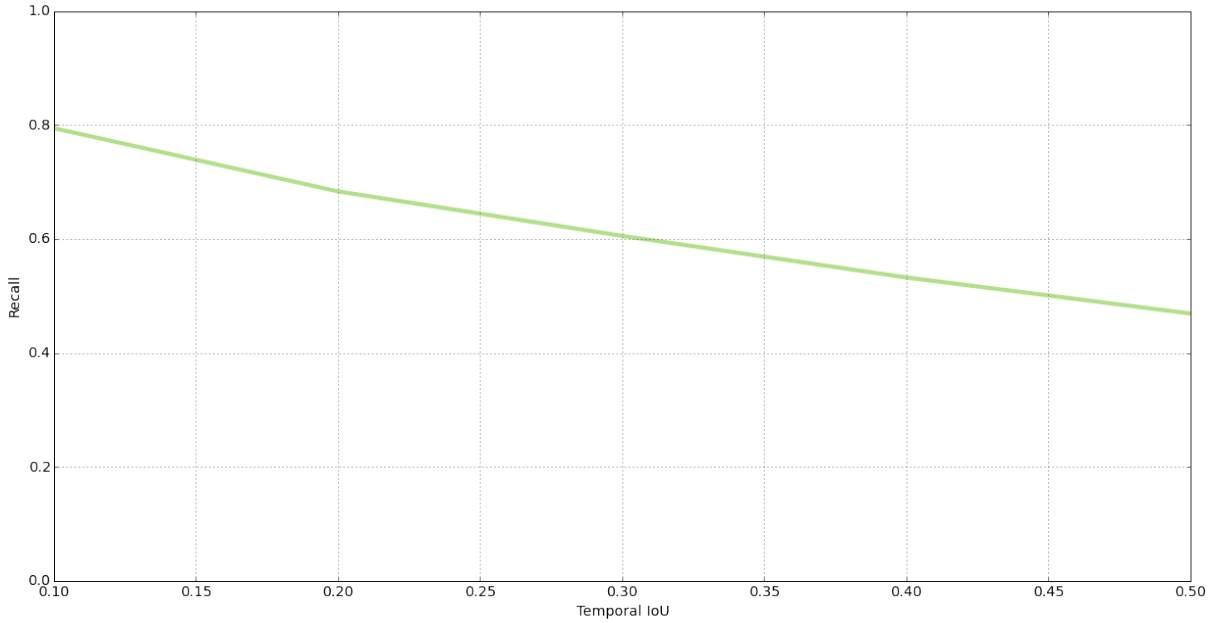



Figure 4.8: Recall of the the temporal regions proposals against different values of the IoU threshold $\alpha$.

Finally, at the time of submission to the ActivityNet Challenge 2016, the same architecture than in the classification task was chosen. The results in both validation and test subset are shown in Table 4.8.

| **Subset** | **mAP** |
|---|---|
| Validation (UPC Team) | 0.2251 |
| Test (UPC Team) | 0.2236 |
| Baseline [10] | 0.0970 |
| Mean Performance from Challenge Participants | 0.2994 |
| Challenge Winner | 0.4247 |

Table 4.8: Results and comparison of the UPC team at the ActivityNet Challenge 2016 for the detection task. The third-party results are all referred to the test subset.

## 4.5 Qualitative evaluation

The behavior of the presented model is further analyzed with some qualitative results shown in Figure 4.9. It presents some examples for classification task, showing the top 3 activities predicted for the each video. Analogously, Figure 4.10 includes a plot the activity detection in time.

Taking a look into the results presented, it can be observed how the classification prediction despite not always being the correct one, it predicts a very similar and related activity which share environment or have very similar action. In addition to this, when the activity require to be temporally localize, sometimes the prediction are not correct due to unsuccessful classifi-

cation. Furthermore the prediction sometimes predicts multiple activity segments for a single ground truth annotation while, on the other hand, the prediction is more conservative when the annotation has multiple temporal segments and the prediction of the proposed network encapsulates them all in a single segment predicted.

Finally, even though the challenge required to temporally localize one single activity per video, the proposed network is capable of predicting multiple classes of activities in a video, just by considering the most probable activity after the smoothing filter. Figure 4.11 shows a case of multiclass activity detection, which proves our claim. Notice that this multiclass scenario was not considered for the ActivityNet challenge, so no videos nor annotations are available for its evaluation.

Video ID: ArzhjEk4j_Y
Activity: Building sandcastles

Prediction:
0.7896 Building sandcastles
0.0073 Doing motocross
0.0049 Beach soccer

Video ID: AimG8xzchfI
Activity: Curling

Prediction:
0.3843 Shoveling snow
0.1181 Ice fishing
0.0633 Waterskiing

Video ID: q8-iXvYyCGg
Activity: Hopscotch

Prediction:
0.8477 Running a marathon
0.0231 Triple jump
0.0218 Javelin throw


Video ID: VrNHEv6aR38
Activity: Playing water polo

Prediction:
0.7645 Playing water polo
0.2018 Swimming
0.0065 Springboard diving

Figure 4.9: Results for the classification task

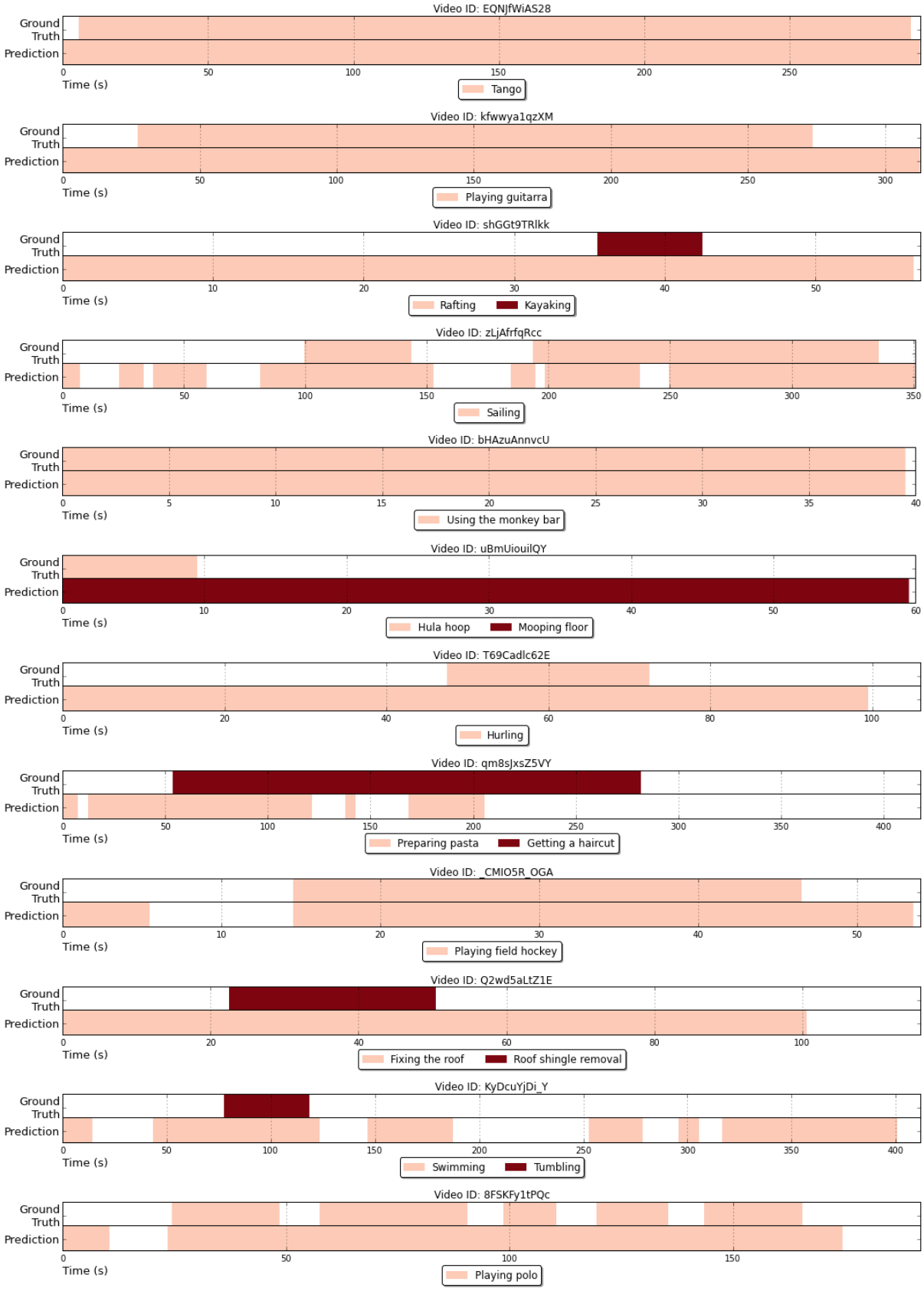



Figure 4.10: Temporal activity localization prediction done by the proposed neural network.

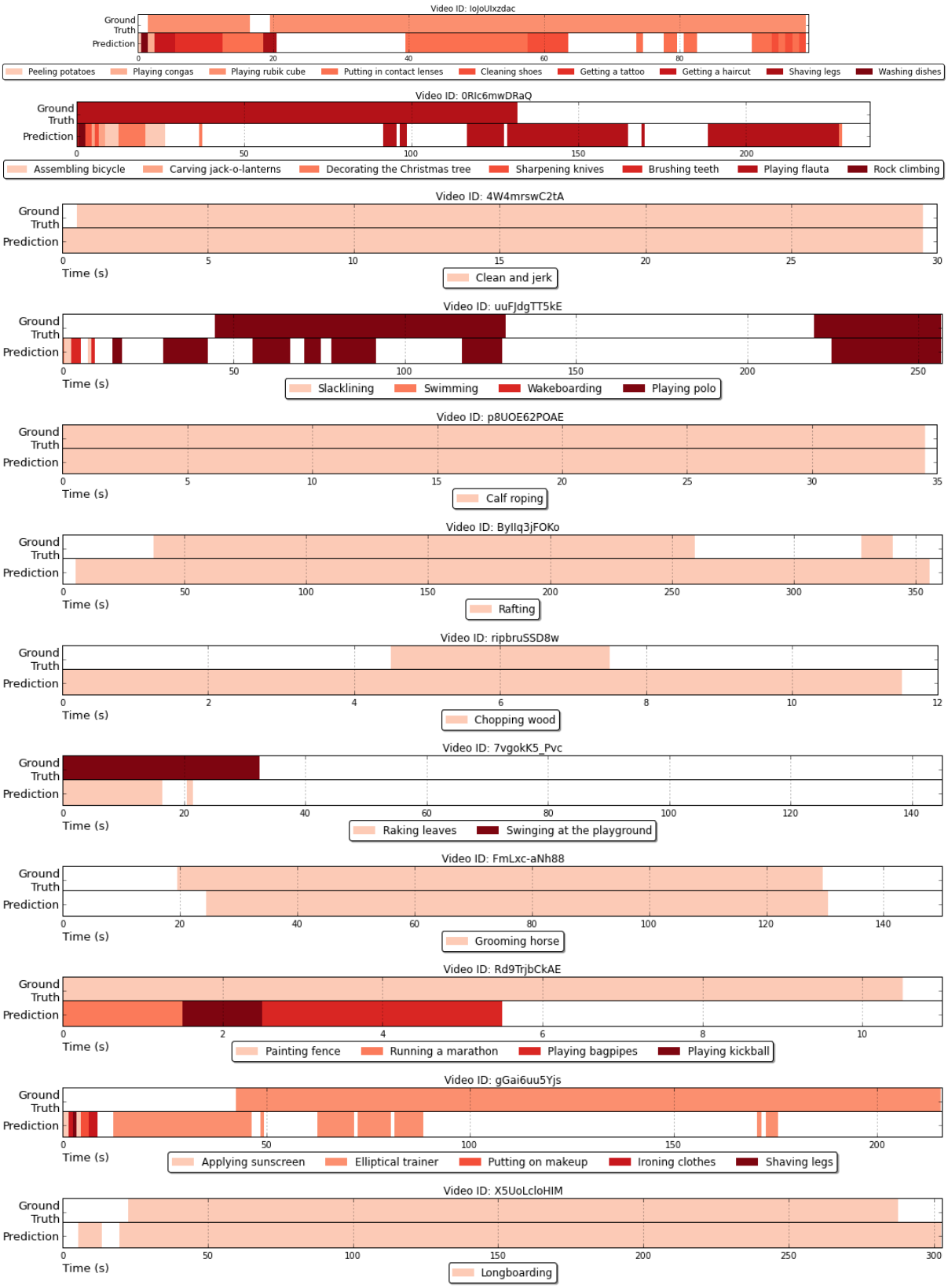



Figure 4.11: Activity predicted from video doing non maximum suppression at each video clip. Different colors represents different activity classes.

# Chapter 5

# Budget

This project has been developed using the resources provided by the Image Processing Group at UPC. For the development, computational resources and equipment was required such as a laptop to develop and a high performance GPUs to train the Neural Networks.

In addition, the main costs of this project comes from the salary of the researchers and the time spent on it. It will be considered that my position has been as Junior Engineer, while my two tutors of the project who where advising me have the salary of a Senior Engineer. It will be considered that the total duration of the project has been 22 weeks, as depicted in the Gantt diagram at Figure 1.2. The project last 22 weeks.

| | **Amount** | **Wage/hour** | **Dedication** | **Total** |
|---|---|---|---|---|
| AWS Instance: *g2.8xlarge* | 1 | 2.6 €/h | 800 h | 2,080 € |
| Junior engineer | 1 | 12 €/h | 30 h/week | 7,920 € |
| Senior engineer | 2 | 20 €/h | 4 h/week | 3,520 € |
| | | | **Total** | 13,520 € |

Table 5.1: Project's budget

# Chapter 6

# Conclusions and Future Work

The main objective of this project was to propose a framework which could be able to face the classification and temporally localization of activities on videos and submit the results to the ActivityNet Challenge 2016. The results obtained by the Neural Network proposed on this thesis to fulfill the goal are very good taking into account that was achieved from scratch. It can be consider that the main goal of this thesis has been accomplished.

Reaching this point though has not been a straight line but full of turns and dead ends. At first it was required to change the framework to use and port a whole model. This additional step gave the opportunity to share the process to the deep learning community which was warmly received. In addition to this, using information extracted from the audio tracks did not improve the performance as expected. To understand the reason, a qualitative exploration over the Dataset was done, concluding that in many videos the audio does not match the activity happening as there is music added after the recording.

Furthermore, a small bias has been observed at the results for both classification and temporally localization of activities. This bias affect the activities that are related to sports and movement activities which achieve better performance than the rest. This can be explained with the network used to extract the features from the frames, which was trained exclusively with sports videos. As this network was not fine-tuned for our data distribution, this bias appeared. As future work it can be possible to study the possibility to train the whole pipeline end to end to improve and get more uniformly distributed results.

It is also important to remark the good results achieved by a simple pipeline which from sequence of video frames predicts a sequence of depicted activities. Moreover, the flexibility of this sequence to sequence prediction allows the proposed network to face tasks where more than a single activity is present on a video. Recently it has been possible to compare the results with the other participants of the ActivityNet Challenge 2016 and it has been observed that the proposed network on this thesis obtain similar results than much more complicated approaches [27] and submissions of much more experienced researchers.

Finally, as future work, it may be interesting to use the optical flow as input as many state of the art publications propose [26, 39, 36]. Another possible modification would be using a multi-stage architecture, predicting first whether there is an action or not, and if so, predict the activity class, as it is proposed in [24]. As a last idea for future work would be to pre-trained the Recurrent Neural Network to predict the next output from the current output. This technique is called semi-supervised and help to initialize a network when training with all your dataset and achieve improved results [14]. All this ideas could also be tested with the Thumos dataset which, as ActivityNet, is made of untrimmed videos.

At last, all the code and models from this thesis to check and reproduce the results is publicly available on: https://github.com/imatge-upc/activitynet-2016-cvprw.

# Appendix A

# Results Figures

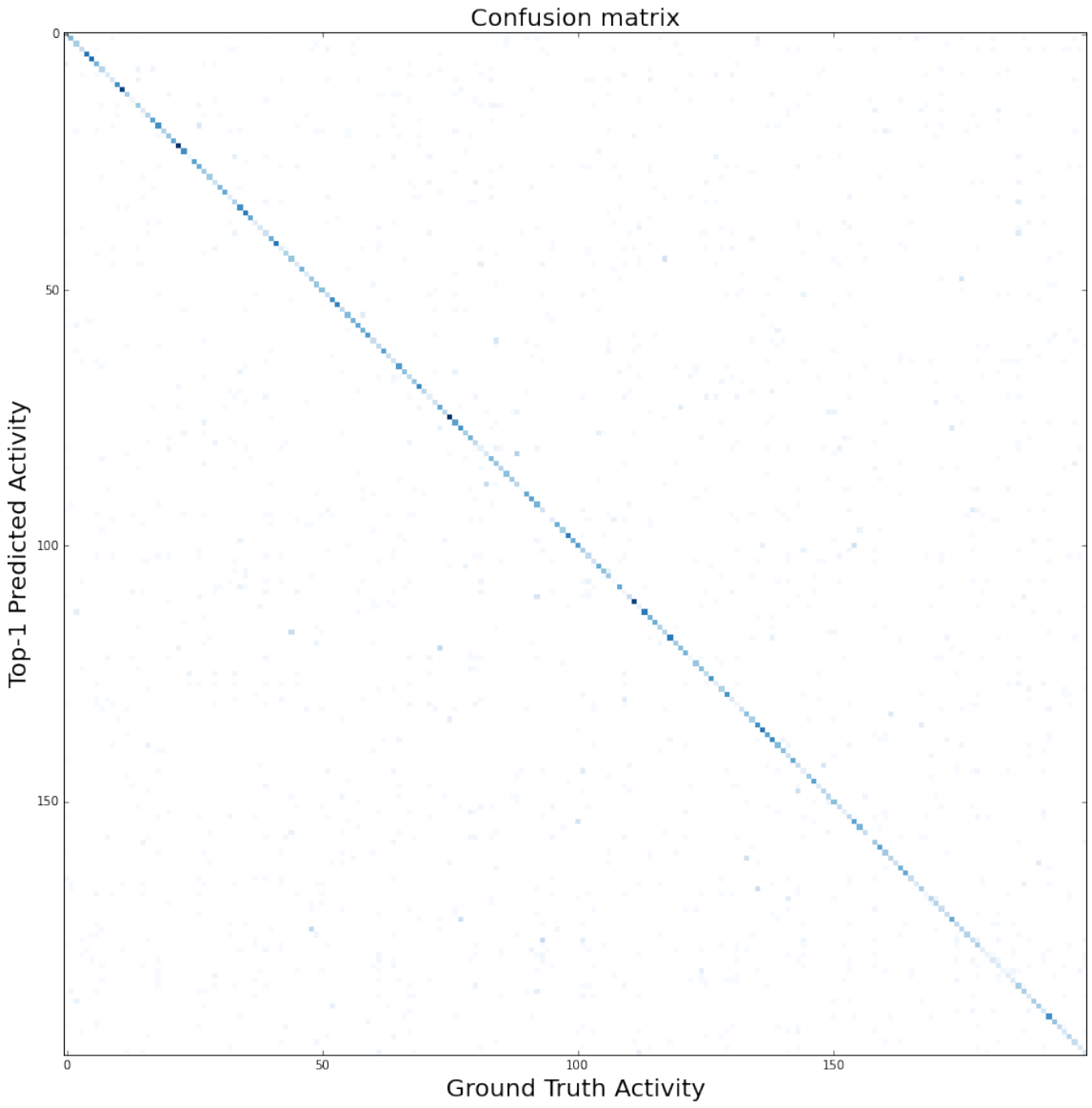
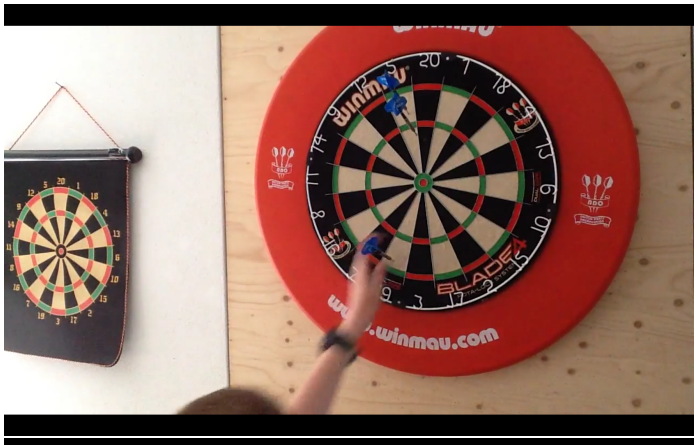



Figure A.1: Confusion matrix of the top-1 activity predicted with the ground truth

Video ID: c-zbA4zixfE
Activity: Throwing darts

Prediction:
0.7325 Throwing darts
0.1364 Baking cookies
0.0254 Playing blackjack




Video ID: p8MvTi8hJdE
Activity: Snowboarding

Prediction:
0.5043 Skiing
0.2683 Snowboarding
0.1637 Snow tubing

Video ID: ATk8OkvNHHQ
Activity: BMX

Prediction:
0.9099 BMX
0.0413 Doing motocross
0.0103 Paintball

Video ID: -uR5-jYeOAg
Activity: Putting on makeup

Prediction:
0.1035 Smoking a cigarette
0.1002 Getting a haircut
0.0941 Putting on makeup

Figure A.2: Results for the classification task

Video ID: vc820BteGzY
Activity: Making a cake

Prediction:
0.3165 Making a lemonade
0.1073 Putting on makeup
0.0768 Making a cake

Video ID: tBNOJJx4Z9k
Activity: Hand washing clothes

Prediction:
0.4742 Cleaning sink
0.0728 Washing hands
0.0497 Baking cookies

Video ID: crbkEVcbF2M
Activity: Windsurfing

Prediction:
0.5009 Windsurfing
0.1129 Hand car wash
0.0738 Sailing

Video ID: j4iaeT5xIdw
Activity: Futsal

Prediction:
0.9889 Futsal
0.0008 Playing field hockey
0.0008 Longboarding


Figure A.3: Results for the classification task

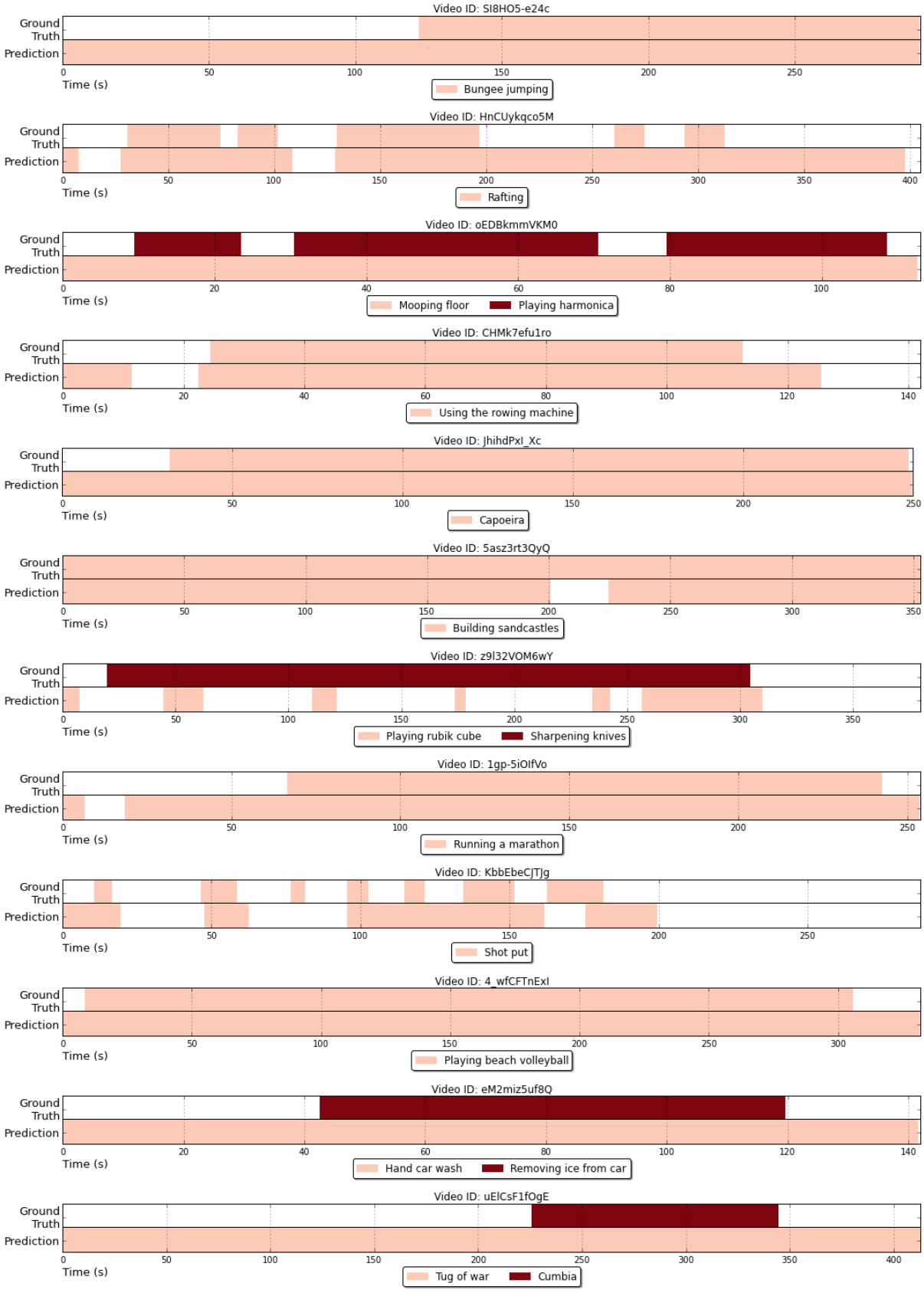



Figure A.4: Temporal activity localization prediction done by the proposed neural network.

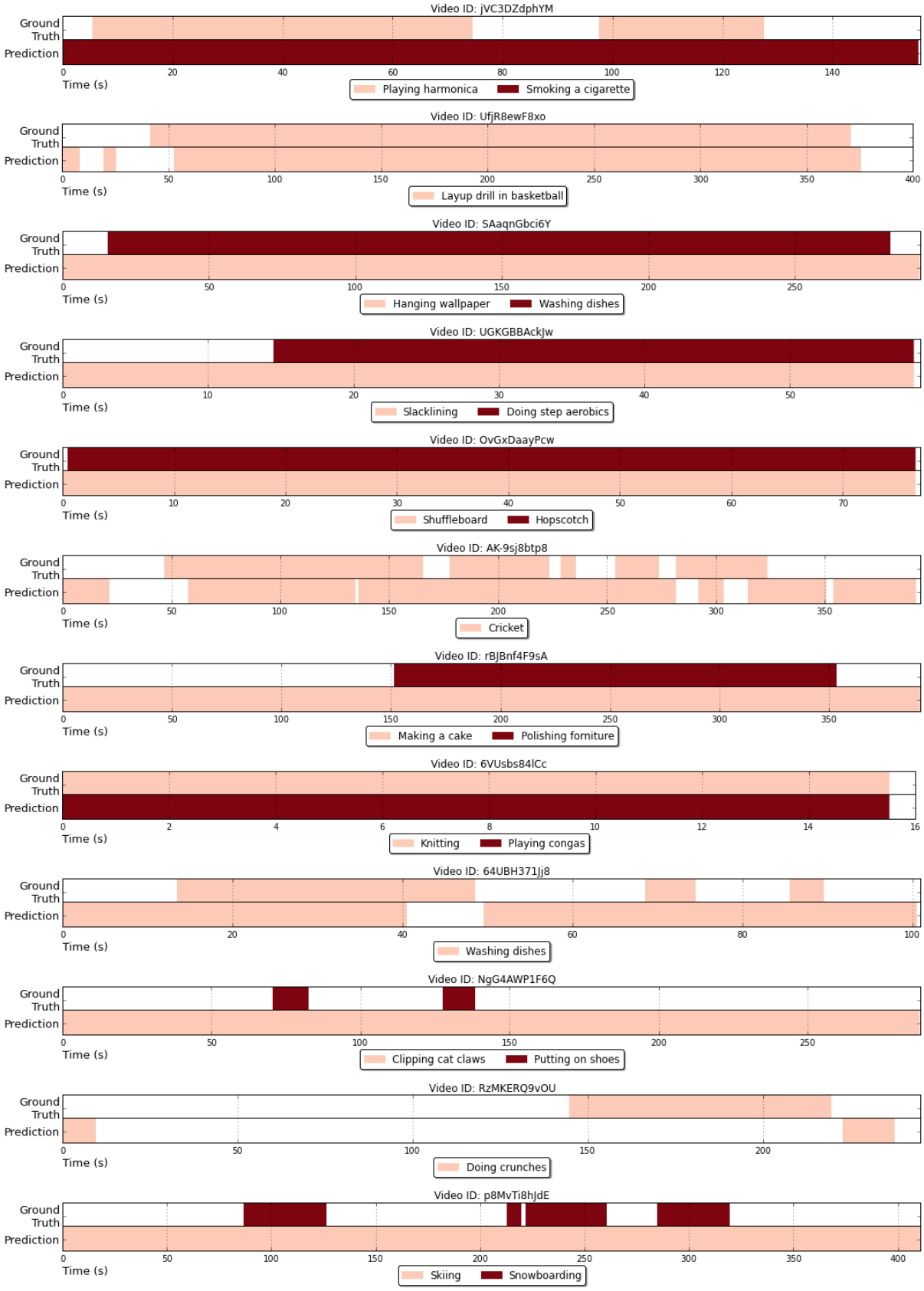



Figure A.5: Temporal activity localization prediction done by the proposed neural network.

# Appendix B

# Notebook Paper

At the appendix there is the notebook submitted to the ActivityNet Challenge 2016 explaining the work done for to face the challenge.

# UPC at ActivityNet Challenge 2016

Alberto Montes, Santiago Pascual de la Puente, Amaia Salvador, Ignasi Esquerra and Xavier Giró-i-Nieto,
Universitat Politècnica de Catalunya
{al.montes.gomez, santi.pdp}@gmail.com, {amaia.salvador, ignasi.esquerra, xavier.giro}@upc.edu

## Abstract

*This notebook describes our proposed solution for both the classification and detection tasks of the ActivityNet Challenge 2016. We propose a system consisting of two different stages. First, the videos are organized in 16-frame clips, for which we individually extract both audio and visual features. Visual features were extracted from a pre-trained 3D convolutional network (C3D), while MFCC coefficients were extracted for audio. On top of these features, we train a recurrent neural network to predict the activity sequence of each video at the granularity of the 16-frames clip.*

## 1. Introduction

Recognizing activities in videos has become a hot topic over the last years due to the continuous increase of video cameras devices and online repositories. This large amount of data requires an automatic indexing to be accessed after capture. The recent advances in video coding, storage and computational resources have boosted research in the field towards new and more efficient solutions for organizing and retrieving video content.

The techniques described in this document have been tested on the video dataset defined by the ActivityNet Challenge 2016. This dataset contains 640 hours of video and 64 million frames. The ActivityNet dataset offers untrimmed videos, which means that has temporal annotations for the given ground truth class labels. Nearly half of the video hours (311 hours of video) contain a label among the 200 activity classes defined by the dataset. This dataset also give the temporal regions where activities occurs. For the details of the ActivityNet dataset please refer to the dataset description[1].

The architecture proposed is composed of two stages. First, we extract spatio-temporal features with a 3D convolutional neural network, which exploits temporal correlations in short video clips. The second stage of our proposed architecture is a Recurrent Neural Network (RNN), which exploits long term dependencies in the feature sequence. The recurrent neural network generates a sequence of predictions that naturally allows the temporal localization of the activities within a video shot.

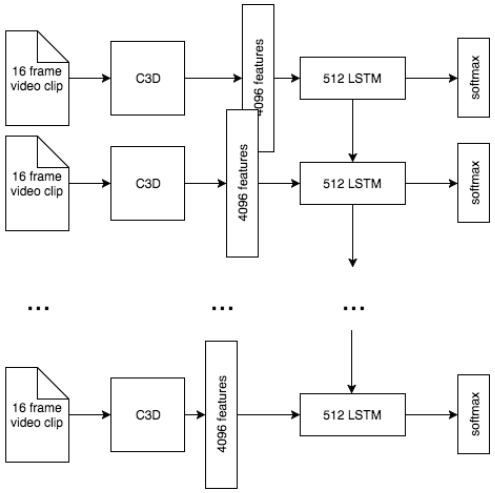



Figure 1. The proposed architecture. The network receives as input the features from the 3DS network, and trains an LSTM to output the class probability for each video clip.

## 2. Architecture

This section explains in detail the two stages of our proposed architecture, which is depicted in Figure 1. This architecture allows solving both the classification and detection tasks formulated in the ActivityNet challenge.

### 2.1. Audiovisual Feature Extraction

In order to extract spatio-temporal correlations on short clips of 16 frames, we adopted the C3D features proposed in [5], which have been proven to be well suited for video classification tasks [4] [6]. We use the network proposed in the original C3D network which was trained with the Sports1M dataset[3] and extract the features from the first fully connected layer (fc6), which was chosen based on the previous results reported in [5]. For visual feature extraction, the videos were split in clips of 16 frames each without overlap,

ending up with a total of 4 million clips. Videos clips were resized to 112x112 pixels for feature extraction, in order to match the original input size for which the C3D network was originally trained. This way, for each 16-frame clip, we obtain a visual feature of dimension 4096.

In addition to the video features, audio features were also explored as additional information for activity recognition. The audio features chosen were 40 MFCC coefficients (20 MFCC + 20 Delta-MFCC coefficients) for 20ms window length and 10ms window shift. In addition 8 Spectral coefficients for global audio track where added. The MFCC coefficients were grouped together to match video features in length and duration. The grouping of the MFCC coefficients was made computing the mean and the standard deviation. In total 88 audio features were computed. They where used in addition to the visual features in order to test if this could improve results. When used it, the audio features were concatenated to the video features out of the C3D before training the recurrent neural network.

### 2.2. Recurrent Neural Network

As a second stage, a Recurrent Neural Network aims at exploiting the long term dependencies in time of the extracted audiovisual features. Our RNN is based on LSTM cells, which control the flow of information that goes through them with gating mechanisms, retaining the necessary information for long periods of time, making them exploit the long-term dependencies better than classic RNNs[2]. We also proposed a sequence to sequence approach, where the model is fit with the video features as a sequence and returns a sequence of the activity class for each clip.

In addition, during our tests we explored an architecture with *feedback*, where the output predicted at the previous time step is added as an input to the LSTM. This approach aimed at smoothing the output sequence of predictions.

## 3. Experiments

The presented model was trained with the training partition provided by the ActivityNet challenge, and the results reported were obtained based on the predictions over the validation set.

### 3.1. Classification Task

For the classification task and knowing that each video has a single activity on it, we obtain the activity probabilities for the whole video as the mean of each activity output through the whole video sequence. Then, we get the maximum among all classes (excluding the background) and sort them by probability. Testing different architectures, we obtained the results given on Table 1 and Table 2. The best configuration was obtained with a single layer of LSTM with 512 neurons using only video features as input, without audio features nor feedback from the previous timestep.

| Architecture | mAP | Hit@3 |
|---|---|---|
| 3 x 1024-LSTM | 0.5635 | 0.7437 |
| 2 x 512-LSTM | 0.5492 | 0.7364 |
| 1 x 512-LSTM | **0.5938** | **0.7576** |

Table 1. Results for classification task comparing different deep architectures. All values with only video features on the validation dataset.

| Features used | mAP | Hit@3 |
|---|---|---|
| Only video | **0.5938** | **0.7576** |
| Video w/ audio | 0.5755 | 0.7352 |
| Only video & feedback | 0.5210 | 0.6982 |
| Video w/ audio & feedback | 0.5652 | 0.7319 |

Table 2. Results for classification task with the model made by one 512-LSTM. Compare between features and feedback on the validation dataset.

| $\alpha$ | $k = 0$ | $k = 5$ | $k = 10$ |
|---|---|---|---|
| 0.2 | 0.207324 | **0.225138** | 0.221362 |
| 0.3 | 0.198542 | 0.220776 | 0.221001 |
| 0.5 | 0.190353 | 0.219376 | 0.213029 |

Table 3. mAP with an IOU threshold of $0.5$ over validation dataset. Here there is a comparison between values on post processing.

### 3.2. Detection Task

In order to solve the detection task, we post-process the output of the network with the assumption that videos only contain a single activity. This way, in this task we only focus on detecting the class with the highest probability throughout the video. To achieve this, we compute the activity probability as the sum of probabilities from all the activities except the background. A threshod $\alpha$ was learned and then applied along the sequence of predictions over the 16-frames clips, so that only the predictions with a probability over the threshold were considered. The best results were obtained with $\alpha = 0.2$.

Finally, a post-processing was required to improve the temporal localization of the activities. A mean filter with a window of $k = 10$ at the output of our recurrent network provided the best results, as seen in Table 3. Figure 2 shows an example of the output of our model.

## References

[1] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–970, 2015.

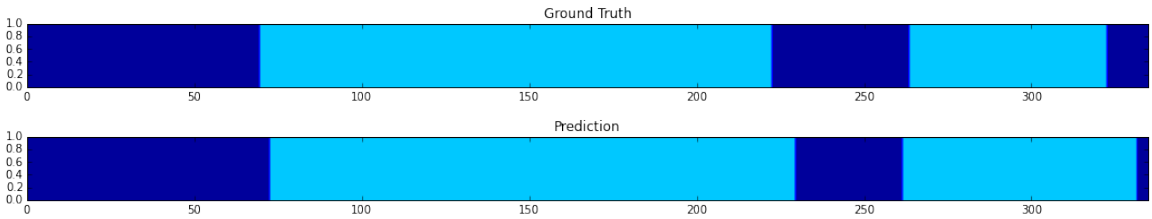



Figure 2. Example of prediction. The dark blue represents background and the light blue represents the *Rafting* activity on video `K3sJnHGHQHM`.


[2] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[3] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[4] Z. Shou, D. Wang, and S.-F. Chang. Temporal action localization in untrimmed videos via multi-stage cnns.

[5] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. *arXiv preprint arXiv:1412.0767*, 2014.

[6] H. Zhang, M. Xua, C. Xu, and R. Jain. Modelling temporal information using discrete fourier transform for video classification. *arXiv preprint arXiv:1603.06182*, 2016.

# Bibliography

[1] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Sequential deep learning for human action recognition. In *Human Behavior Understanding*, pages 29–39. Springer, 2011.

[3] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*, 2015.

[4] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179, 2015.

[5] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez, Sankalp Gulati, Perfecto Herrera, Oscar Mayor, Gerard Roma, Justin Salamon, José R Zapata, and Xavier Serra. Essentia: An audio analysis library for music information retrieval. In *ISMIR*, pages 493–498. Citeseer, 2013.

[6] G. Bradski. Opencv. *Dr. Dobb's Journal of Software Tools*, 2000.

[7] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[8] Yann N Dauphin, Harm de Vries, Junyoung Chung, and Yoshua Bengio. Rmsprop and equilibrated adaptive learning rates for non-convex optimization. *arXiv preprint arXiv:1502.04390*, 2015.

[9] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2625–2634, 2015.

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

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[12] Georgia Gkioxari, Ross Girshick, and Jitendra Malik. Contextual action recognition with r* cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1080–1088, 2015.

[13] G Gravier. Spro: a free speech signal processing toolkit, 2010.

[14] Félix G Harvey and Christopher Pal. Semi-supervised learning with encoder-decoder recurrent neural networks: Experiments with motion capture sequences. *arXiv preprint arXiv:1511.06653*, 2015.

[15] Toni Heittola, Annamaria Mesaros, Antti Eronen, and Tuomas Virtanen. Context-dependent sound event detection. *EURASIP Journal on Audio, Speech, and Music Processing*, 2013(1):1–13, 2013.

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[18] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[19] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[20] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696, 2011.

[21] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*, 2012.

[22] AJ Piergiovanni, Chenyou Fan, and Michael S Ryoo. Temporal attention filters for human activity recognition in videos. *arXiv preprint arXiv:1605.08140*, 2016.

[23] Shikhar Sharma, Ryan Kiros, and Ruslan Salakhutdinov. Action recognition using visual attention. *arXiv preprint arXiv:1511.04119*, 2015.

[24] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal action localization in untrimmed videos via multi-stage cnns.

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[26] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014.

[27] Bharat Singh and Ming Shao. A multi-stream bi-directional recurrent neural network for fine-grained action detection.

[28] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[29] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, 2016.

[30] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. *arXiv preprint arXiv:1412.0767*, 2014.

[31] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Deep end2end voxel2voxel prediction. *arXiv preprint arXiv:1511.06681*, 2015.

[32] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards Good Practices for Very Deep Two-Stream ConvNets. *ArXiv e-prints*, July 2015.

[33] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, pages 4305–4314, 2015.

[34] Limin Wang, Yuanjun Xiong, Zhe Wang, and Yu Qiao. Towards good practices for very deep two-stream convnets. *arXiv preprint arXiv:1507.02159*, 2015.

[35] Zhongwen Xu, Linchao Zhu, Yi Yang, and Alexander G Hauptmann. Uts-cmu at thumos 2015. *THUMOS challenge*, 2015.

[36] Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. Describing videos by exploiting temporal structure. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4507–4515, 2015.

[37] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *arXiv preprint arXiv:1507.05738*, 2015.

[38] Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. *arXiv preprint arXiv:1511.06984*, 2015.

[39] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[40] Haimin Zhang, Min Xua, Changsheng Xu, and Ramesh Jain. Modelling temporal information using discrete fourier transform for video classification. *arXiv preprint arXiv:1603.06182*, 2016.