Graduate School Form
30 Updated 12/26/2015

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Konstantinos Raptis

Entitled

THE CLASH BETWEEN TWO WORLDS IN HUMAN ACTION RECOGNITION: SUPERVISED FEATURE TRAINING VS RECURRENT CONVNET

For the degree of Master of Science

Is approved by the final examining committee:

Gavriil Tsechpenakis
Chair

Jiang Yu Zheng

Mihran Tuceryan

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): Gavriil Tsechpenakis

Approved by: Shiaofen Fang 12/2/2016

Head of the Departmental Graduate Program Date

THE CLASH BETWEEN TWO WORLDS IN HUMAN ACTION RECOGNITION: SUPERVISED FEATURE TRAINING VS RECURRENT CONVNET

A Thesis

Submitted to the Faculty

of

Purdue University

by

Konstantinos Raptis

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

December 2016

Purdue University

Indianapolis, Indiana

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

CNN	Convolutional Neural Network
DIT	Dense Improved Trajectories
GMM	Gaussian Mixture Model
HOG	Histogram of Oriented Gradients
HOF	Histograms of Optical Flow
KLT	Kanade-Lucas-Tomasi
LSTM	Long Short-Term Memory
LTC	Long-term Temporal Convolution
NN	Neural Network
MBH	Motion Boundary Histograms
PCA	Principal Component Analysis
RCN	Recurrent Convolutional Network
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SIFT	Scale-Invariant Feature Transform
SURF	Speeded Up Robust Features
SVM	Support Vector Machines
TDD	Trajectory-pooled Deep-Convolutional Descriptors
YOLO	You Only Look Once

ABSTRACT

Raptis, Konstantinos. M.S., Purdue University, December 2016. The Clash between two worlds in Human Action Recognition: Supervised Feature Training vs Recurrent ConvNet. Major Professor: Gavriil Tsechpenakis.

Action recognition has been an active research topic for over three decades. There are various applications of action recognition, such as surveillance, human-computer interaction, and content-based retrieval. Recently, research focuses on movies, web videos, and TV shows datasets. The nature of these datasets make action recognition very challenging due to scene variability and complexity, namely background clutter, occlusions, viewpoint changes, fast irregular motion, and large spatio-temporal search space (articulation configurations and motions). The use of local space and time image features shows promising results, avoiding the cumbersome and often inaccurate frame-by-frame segmentation (boundary estimation). We focus on two state of the art methods for the action classification problem: dense trajectories and recurrent neural networks (RNN). Dense trajectories use typical supervised training (e.g., with Support Vector Machines) of features such as 3D-SIFT, extended SURF, HOG3D, and local trinary patterns; the main idea is to densely sample these features in each frame and track them in the sequence based on optical flow. On the other hand, the deep neural network uses the input frames to detect action and produce part proposals, i.e., estimate information on body parts (shapes and locations). We compare qualitatively and numerically these two approaches, indicative to what is used today, and describe our conclusions with respect to accuracy and efficiency.

1. INTRODUCTION

Humans are capable to detect objects, distinguish different types of motion patterns and analyze complex interactions and temporal events. We can instantaneously determine whether a person is shooting a ball, waving, answering the phone or hugging another person even in the presence of cluttered background, occlusions, and/or illumination changes. However, understanding complex events is hard even for a human. Improving our interaction with machines and take advantage of them would have great benefits in our everyday life.

Applications such as autonomous vehicles or surveillance systems would play a big role in our life in the very near future. The decreasing amount of car accidents and the identification of abnormal events are considered an important problem that has to be solved. In order to attain these goals, smart systems need to be developed for monitoring and understanding our surroundings using images, videos, sensors, or depth cameras. The area of Computer Vision has evolved and is now capable of building such smart systems and solving these kind of problems. Several models have been developed for addressing problems such as object detection and action recognition.

This thesis focuses in the analysis and classification of human activities. The nature of these data make the action recognition problem very challenging due to scene variability and complexity, namely background clutter, occlusions, viewpoint changes, fast irregular motion, and large spatio-temporal search space (articulation, configurations and motions). In addition, recognizing the behavior of a person in a video is challenging due to the variability of ways that different people perform a particular action. For instance, different people run in different pace or style of motion. Problems are also encountered due to cluttered background in images. Moreover, the high dimensionality of the data are also significant challenges for the classification

problem. The goals of this thesis are the development and extraction of low level features, the development of learning algorithms for recognizing human action in videos and a comparison of two of the most sophisticated algorithms used in this problem (dense trajectories and recurrent neural networks).

We focus on two state of the art methods for the action classification problem: dense trajectories and Recurrent Neural Networks (RNN). Dense trajectories use typical supervised training (e.g., Support Vector Machines) of features such as 3D-SIFT, extended SURF, HOG3D, and local trinary patterns; the main idea is to densely sample these features in each frame and track them in the sequence based on optical flow. On the other hand, the deep Neural Network uses the input frames to detect action and produce part proposals, i.e., estimate information on body parts (shapes and locations). We compare qualitatively and numerically these two approaches, indicative to what is used today, and describe our conclusions with respect to accuracy and efficiency.

We apply our methods on two different datasets. The hollywood 2 [1] and the HDMB 51 [2] dataset. In Figures 1.1-1.2 we can see some examples of several actions from both datasets. Hollywood2 has been collected from 69 different hollywood movies including 12 classes of actions. It contains 1707 videos split into a training set of 823 videos and a test set of 884 videos. Train and test videos are split on the movies, such that a movie can only appear on either the train or the test set. HDMB51 is collected from movies and YouTube videos. There are 51 different actions in a total of 6766 video sequences with three train-test splits. For every class and split there are 70 training and 30 testing videos.

1.1 Problem Statement and Challenges

Human action recognition is the problem of identifying an action performed by one or more humans from a collection of observations,in our case image sequences. More specifically, given a video sequence, the goal is to determine which, of a set

Fig. 1.1. Samples of several actions from Hollywood2 dataset. From left to right the corresponding actions are: Stand up, Answer phone, Drive, Run.

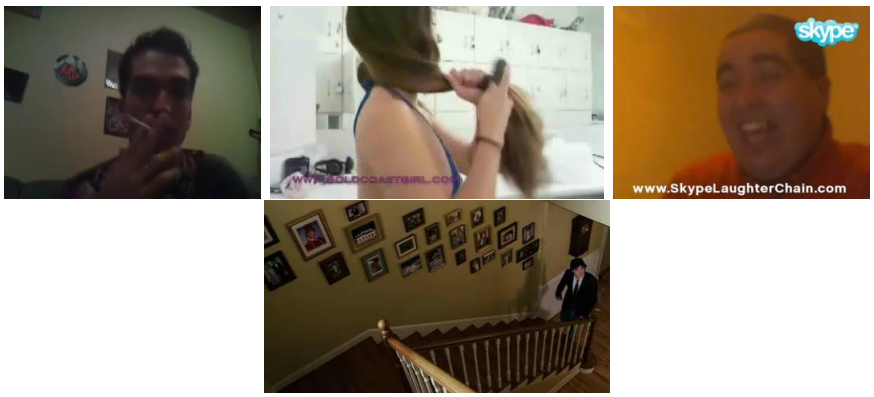

Fig. 1.2. Samples of several actions from HMDB51 dataset. From left to right the corresponding actions are: Smoke, Brush hair, Laugh, Climb stairs.

of predefined action classes, can be more successfully assigned to the sequence. For example, we want to classify a sequence of a new video to action labels, such as answer phone, waving, shooting ball, stand up etc.. As mentioned above, the task of action recognition can be particularly difficult due to scene variability and complexity namely background clutter, occlusions, viewpoint changes, fast irregular motion, and large spatio-temporal search space (articulation, configurations and motions).

1.2 Thesis Outline

The remaining of the thesis is organised as follows. In Chapter 2, we review existing work in the action recognition area and discuss some of the most popular techniques. Detailed explanation of our methods are discussed in Chapters 3 and 4. We present experiments, results and comparison in Chapter 5 and conclusions as well as future work in Chapter 6.

2. BACKGROUND

In this chapter, we review existing work in the area of action recognition. The existing approaches can be separated into four main categories. We will give a detailed explanation of the two methods that we use along with the related work in each field and we will briefly describe the two remaining methods.

1. Holistic representations. Global information is extracted from every image of a video in order to perform human detection. These features are focused on the person that performs an action.

2. Part based representations. That is, the middle level representation of an action, based on local spatio-temporal characteristics in a sequence of images.

3. Local-feature based methods. That is, the representation of the entire video as a group of descriptors obtained from regions of interest, without pre-processing steps such as tracking and segmentation.

4. Deep Learning. Most of state of the art approaches use Convolutional Neural Networks (CNN) or Recurrent Neural Networks (RNN). They attempt to learn 3D spatio-temporal filters over raw sequence data and frame-to-frame representations. RNN are considered as "deep in time" models while CNN "deep in space" models.

2.1 Holistic Representations

These approaches extract global information from every image of a video in order to perform human detection. Global information is focused on the person that performs an action and is represented by features that are obtained from pixel information of regions of interest, in our case the human body. These features capture

the essence of the human body shape and more importantly the difference between two human silhouettes. The feature extraction is determined by "preprocessing" algorithms such as background subtraction [3, 4], person detection [5–7] or tracking [8, 9] and their ability to extract these global features. A static background or a scene that makes tracking less complicated are required in order for these algorithms to have the anticipated outcome. Other than using the human body shape to represent an action, dense optical flow can be used in the detected area. The advantage of dense optical flow over the silhouette based method is that it avoids the often inaccurate frame-by-frame segmentation.

2.2 Part based representation

Part based representation decomposes an action into several parts capturing local spatio-temporal characteristics in the data. A part-based model represents the human body as a set of rigid parts (e.g., head, leg, etc.) constrained in a way. The used constraints are mainly tree-structured kinematic constraints between body parts. Two major components are of importance. Part appearances specify the appearance of each body part in the image, and configuration priors specify the arrangement of parts with respect to each other. Most of the approaches use sequential data models to represent the temporal variability [10–13]. It is common to use time series of activity codewords, and for each frame it detects "interesting" regions as a part of an action and enforcing the temporal consistency. Each frame is usually segmented with the mean-shift algorithm [14]. As descriptors, local based methods use spatio-temporal HOGs [5, 15].

2.3 Local feature based methods

Local feature based methods represent the video as a group of descriptors, capturing shape and motion characteristics from spatio-temporal regions. These methods can be described in three steps. First, we need to find the points of an image that

correspond to points on another image. These points are usually the local maximum of image functions such as SIFT, SURF, etc. and are tracked over time. The next step is to apply feature descriptors on these points. The most common descriptor is the Histogram of Oriented Gradient (HOG). During the last years, several descriptors were proposed such as Motion Boundary Histogram (MBH), Histogram Of Flow (HOF), which is usually based on a dense optical flow, see [16]. The final step, is the representation of the video. The most common approach is to use Bag of Features or Fisher Vector [16, 17]. Finally, an action is usually classified using Support Vector Machines (SVM) either linear or kernel. There are several approaches that extract spatio-temporal features in videos. In [18], Dollar et al. propose a method for detection of spatial interest points and extensions to the spatio-temporal domain using a a combination of a 2D Gaussian filter in space and a 1D Gabor filter in time. In [19], this previous work has been extended by using a 2D Gabor filter of different orientations. In [20], Scovanner et al. introduce a 3D SIFT descriptor for video. In [15], Klaser et al. introduce the HOG3D descriptor. Willems et al., in [21], present spatio-temporal interest points that are both scale invariant and densely cover the video content.

Several methods related to trajectories have been developed. Trajectories capture motion information given a spatial point. In [22], Messing et al. extract feature trajectories using Birchfield's implementation of the KLT tracker on videos. In [23], a method for motion, based on quantized trajectory snippets of tracked features, was presented which is very computationally efficient. In [24], the spatio temporal context information is modeled in a hierarchical way. They match SIFT descriptors in consecutive frames. A unique match between the frames is required and matches that are far apart are rejected. In [16, 17], Wang et al. propose a method where points are densely sampled and tracked using a dense optic flow field. Dense sampling is used in order to have a good coverage of the video with features and optic flow is used to improve the quality of trajectories. Motion Boundary Histograms (MBH) represent

the gradient of optic flow. In this way camera motion is suppressed but information about the flow field changes are kept.

2.4 Deep Neural Networks

Although Deep Learning methods may fall in one of the previous categories, such as local features based methods, we want to dedicate a separate section for these kind of models. Research on CNN and RNN is either new or recently reintroduced. We estimate in the following years more applications that use CNN or RNN will emerge. We describe the most notable research for both CNN and RNN in computer vision area, especially for action recognition.

In [25], Simonyan and Zisserman used deep CNN for action recognition in videos. The Convolutional Network incorporated spatial and temporal networks and was trained on a dense optical flow. Tran et al. in [26] propose an approach to learn spatial and temporal features using deep 3D ConvNets pointing out the adnantages of 3D ConvNets compared to 2D ConvNets for spatiotemporal feature learning. Donahue et al., in [27], propose a model suitable for large-scale visual understanding tasks using RNNs. Unlike previous models, it makes use of not only spatial but also temporal dimensions. The recurrent sequence models are connected to the convolutional network model and they can be trained jointly in order to learn temporal dynamics. As a result, they have many advantages compared to models that are separately defined or optimized. In [28], Ji et al. developed a 3D CNN for action recognition in image sequences. Taylor et al., in [29], perform convolutional learning of spatio-temporal features using pairs of successive images. In [30], Wang et al. use hand-crafted as well as deep learned features for action recognition in a method called trajectory-pooled deep convolutional descriptor (TDD). In [31] Baccouche et al. propose an action classification method. It is based on the extension of CNNs to 3D in order to learn the features and then a RNN is trained to classify each sequence of the learned features. In [32], Peng et al. propose a Multi-Region two stream R-CNN for action

recognition. Faster R-CNN is used for frame level action detection to combine motion and appearance region proposals. In [33] Grushin et al. use LSTM-RNN for robust action recognition in real world scenarios i.e. poor video quality, small quantities of training data, tighter deadlines to make a decision. These models use unidirectional LSTM-RNNs with one hidden layer. Lefebvre et al., in [34], present a robust method for 3D gesture recognition using a bidirectional LSTM-RNN where a forward hidden layer and a backward hidden layer are used. In most of the cases, motion based CNNs have better performance than CNN representations learned for RGB inputs [25]. Moreover, CNN with 3D convolutions extend the 2D shift invariance to invariance to translation in time [26, 28, 29]. All of the methods mentioned above learn videos with RGB input and the video sequences are relatively small (around 15 frames). In [35], Varol et al. use Long-term Temporal Convolutions (LTC) in order to apply CNN to longer videos. In [36], Karpathy, et al. classify over 1 million videos into 487 classes by "fusing" the representation of frames over time. In [37], Du et al. propose a hierarchical RNN for skeleton based action recognition by dividing the human skeleton into 5 parts.

3. DENSE TRAJECTORIES

Among the local feature based methods, dense trajectories [16,17] seem to have better performance on a variety of datasets. The main idea is to densely sample feature points in each frame and track them through the video using optical flow. During the process, multiple descriptors are computed for capturing shape, appearance and motion information.

3.1 Our approach

Our approach has many similarities with the Dense Improved Trajectories (DIT), [17]. More specifically, we use the same camera motion estimation technique, the same method for trajectory features and also one of the approaches mentioned for feature encoding. The DIT approach can be separated into five parts: camera motion estimation, removing inconsistent matches due to humans, trajectory features, feature encoding and classification.

Camera motion estimation

We assume that two frames in a row are related by a homography in order to estimate the global background motion. We make this assumption based on the fact that usually the global motion between two frames is small, excluding objects like humans and vehicles. 15 frames are used in our case to estimate the camera motion. To estimate the homography, we find the correspondences between two frames. At first, we extract SURF features,see Figure 3.1-3.2, and match them with a nearest neighbor algorithm. SURF features have the advantage of being robust to motion blur. Then, we sample motion vectors from optical flow, see Figure 3.3. This

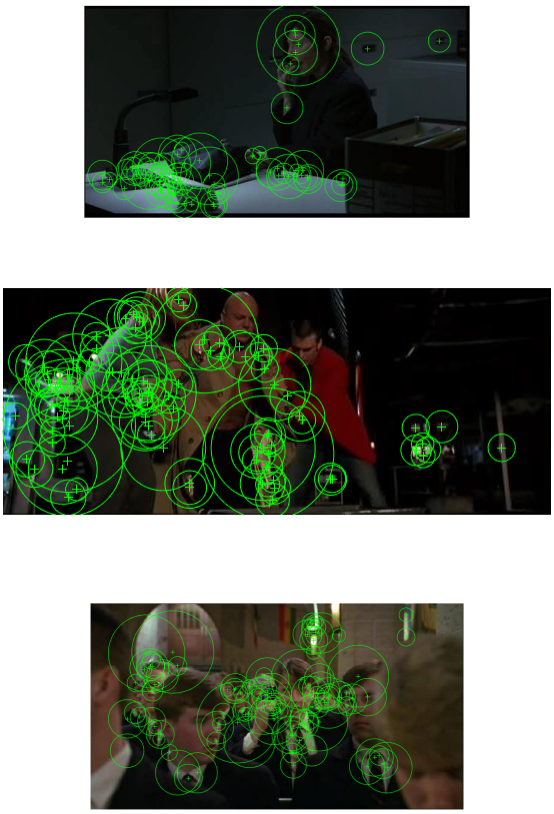

Fig. 3.1. Examples of SURF features extraction on Hollywood2 dataset.

technique provides us with dense matches between frames. The optical flow algorithm we use was implemented by [38]. These two approaches are complementary, SURF focuses on blob structures, whereas optic flow focuses on corners and edges. We estimate the homography using RANSAC [39] as in [16]. This allows us to remove camera motion.

Removing inconsistent matches due to humans

In the majority of datasets, an action is centered around humans . As humans are the main focus point in the image, problems arise for the camera motion estimation since human motion is not consistent with it. Moreover, many difficulties

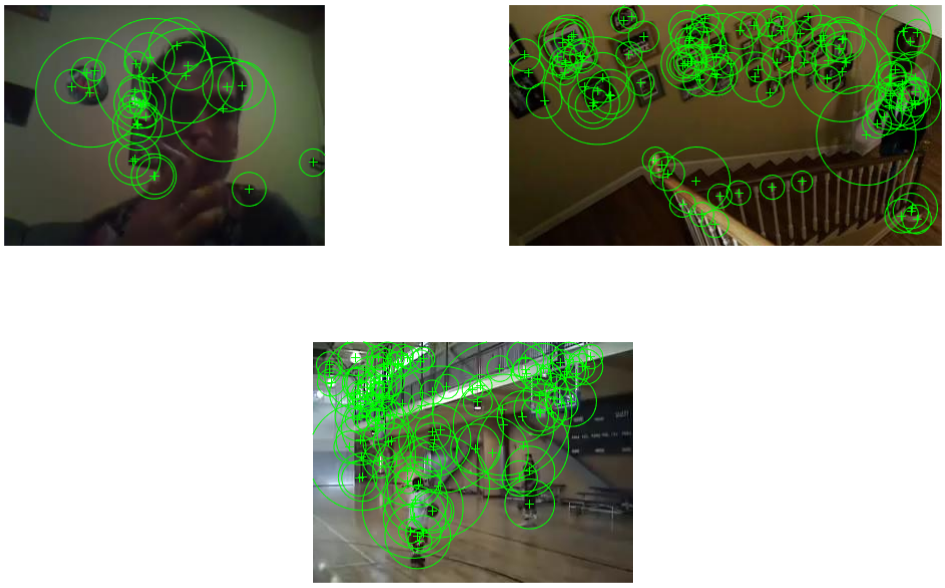

Fig. 3.2. Examples of SURF features extraction on HMDB51 dataset.

Fig. 3.3. Example using dense Optical flow on two stacked frames.

are encountered due to the fact that in action recognition datasets the human body is not easily identified. For this reason, we use a HOG human detector, created by combining part-focused detectors, which is trained with the PASCAL VOC07 data. The bounding boxes are used to remove matches that don't correspond to camera motion. Finally, we are able to extract the trajectory features.

Trajectory features

Different descriptors are computed for each trajectory (Trajectory, HOG, HOF, MBH). The Trajectory descriptor is a concatenation of normalized displacement vectors. The rest of the descriptors are computed in the space-time volume aligned with the trajectory. HOG, as the name implies, is formed of the oriented gradients and captures appearance information. HOF and MBH are based on optical flow, therefore they capture motion information. The feature points are tracked for 15 frames. The dimensionality of the descriptors is as follows: Trajectory 30, HOG 96, HOF 108 and MBH 192.

Feature encoding

In order to encode features, we use Fisher vector with identical settings to [16]. Fisher vector encodes first and second order statistics between the video descriptors and a Gaussian Mixture Model (GMM). As in [16], we start by reducing the descriptors' dimensionality using PCA (Principal components Analysis). The number of Gaussians we use for the GMM is K = 256. A random set of features is taken from the training set to estimate GMM. Each video is represented by a 2*D*K long fisher vector for each descriptor, where D is the descriptor dimension after dimensionality reduction. Finally, power and L2 normalization are applied to Fisher Vector as in [40].

Classification

For classification, a Linear SVM with C=100 is used. We use a one-versus-the-rest approach since the nature of the data require a multi-class classifier.

4. RCN

In our approach we combine several state of the art algorithms, the highlight being the combination of CNN and RNN. We name our model recurrent convolutional network (RCN)and we implement it with TensorFlow 0.10. We use DarkNet (YOLO) for human detection, as CNN we used the Inception v3.

4.1 CNN-RNN

In the next section we discuss in detail the characteristics of both CNN and RNN, Tensorflow, YOLO and Inception v3.

4.1.1 CNN

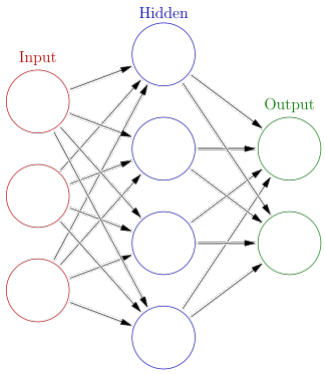

Fig. 4.1. An artificial neural network. Each circular node represents a neuron and each arrow represents a connection from the output of one neuron to the input of another [41].

A Neural Network is "...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dy-

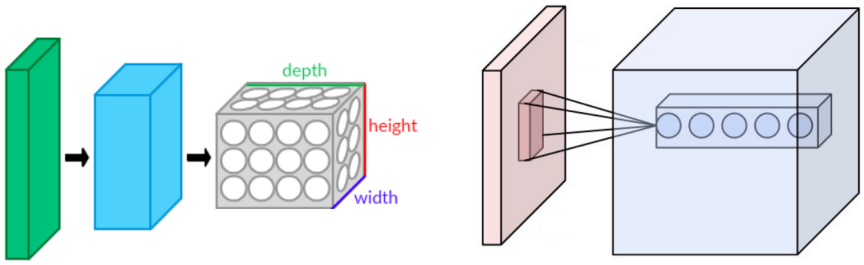

Fig. 4.2. Left: CNN layers arranged in 3 dimensions [42]. Right: Neurons of a convolutional layer, connected to their receptive field [43].

namic state response to external inputs." (In "Neural Network Primer: Part I" by Maureen Caudill, AI Expert, Feb. 1989). Neural Networks use forward feeding, which means that they pass signals along the input-output channel in a single direction, without allowing signals to loop back into the network, Figure 4.1. While Neural Networks are very successful for image recognition, they come with a great cost. This method requires all neurons to be connected and as a result the network becomes very complex. As the size of the datasets becomes larger, more limitations are encountered.

In order to overcome this problem, Convolutional Neural Networks (CNNs) were created. Figure 4.3 shows the typical CNN architecture. Unlike Neural Networks, they are sparsely connected on the input layer. CNNs are inspired by the cat's visual cortex which contains an arrangement of cells that are sensitive to small regions of space. These regions overlap to cover the entire visual field. The cells act as filters that process input images and are later passed to subsequent layers. The most commonly used layers are Convolutional, Pooling, Rectified Linear Units, Fully Connected and Loss layer.

Convolutional layer

The convolutional layer consists of learnable filters. These filter cover small regions of space but they extend through the full depth of the input volume, Figure 4.2. Each filter represents a feature of interest and they are translational invariant.

Pooling layer

The pooling layer is where a subsampling occurs in order to reduce the filter sensitivity to variation and noise. Another reason that subsampling is used, is to reduce the parameters and the computations. Subsampling is usually performed between two convolutional layers.

ReLU layer

Rectified Linear Units (ReLU) layer deals with how the signal travels from one layer to another. There are multiple activation functions, the most commonly used is ReLU f(x)=max{0,x} because of its fast training time. After applying several convolutional and subsampling layers, another layer is applied.

Fully Connected layer

In fully connected layer, neurons of preceding layers are connected to all neurons in the following layer similarly to neural networks.

Loss layer

The Loss layer is usually the last layer of the network. During training it specifies the penalization between the predicted and the true values. There are several loss functions such as Softmax, Sigmoid, cross-entropy and Euclidean loss function.

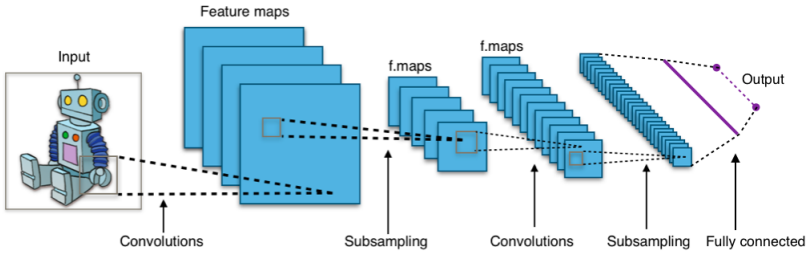

Fig. 4.3. Typical CNN architecture [44].

4.1.2 RNN-LSTM

A recurrent neural network (RNN) is a subset of neural networks where a connection between units form a directed cycle. Given an input sequence $x = \left(x^0, ..., x^{T-1}\right)$, we can derive the hidden states of a recurrent layer $h = \left(h^0, ..., h^{T-1}\right)$ as well as the output of a single hidden layer $y = \left(y^0, ..., y^{T-1}\right)$ as follows:

$$h^t = H\left(W_{xh}x^t + W_{hh}h^{t-1} + b_h\right)$$

$$y^t = O\left(W_{ho}h^t + b_0\right)$$

where W_{xh}, W_{hh} and W_{ho} denote the weights of the connections between the layers x and h, h and h and h and o respectively. H and O are the activation functions at the hidden and output layer and b_h and b_o are two bias vectors. An important limitation of Convolutional Neural Networks is that their input and output is a fixed sized vector, which means that inputs and outputs are independent from each other. The amount of computational steps is also fixed (number of layers in the model). What makes a RNN more interesting is that it operates on sequences of vectors. In other words, RNNs have a memory that keeps information about what has been calculated so far. Although in theory we can use long sequences, practically there are limitations to the number of steps we can look back to. In order to train a RNN, we use a backpropagation algorithm. Since all time steps share the parameters, the gradient depends on calculations of the current time step as well as on preceding time steps. In practice, difficulties are very often encountered when a RNN is trained (especially deep RNN) when activation functions such as sigmoid and tanh are used due to the vanishing gradient problem.

In order to solve this problem, the Long-Short Term Memory (LSTM) architecture was introduced in 1997 by Hochreiter and Schmidhuber [45]. LSTMs are designed to deal with the error blowing up problem. By default, LSTMs remember information for long time periods. The difference between standard RNNs and LSTMs is that the sequence of repeating modules that we encounter in RNNs has a different structure in the case of LSTMs. It contains a memory cell and three multiplicative units where

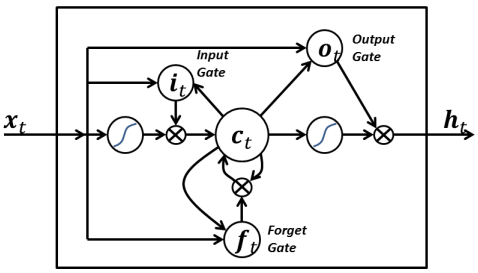

Fig. 4.4. A simple LSTM block with only input, output, and forget gates. LSTM blocks may have more gates [46].

information of the temporal sequence is stored, written to and read from. Figure 4.4 shows the activation memory cell c and the three multiplicative units, where i is the input gate, f is the forget gate and o is the output gate. The activation of c and the gates are given as follows:

$$i^t = \sigma\left(W_{xi}x^t + W_{hi}h^{t-1} + W_{ci}c^{t-1} + b_i\right)$$

$$f^t = \sigma\left(W_{xf}x^t + W_{hf}h^{t-1} + W_{cf}c^{t-1} + b_f\right)$$

$$c^t = f^t c^{t-1} + i^t tanh\left(W_{xc}x^t + W_{hc}h^{t-1} + b_c\right)$$

$$o^t = \sigma\left(W_{xo}x^t + W_{ho}h^{t-1} + W_{co}c^t + b_o\right)$$

$$h^t = o^t tanh\left(c^t\right)$$

where W represents the connection weights between two units and σ is the sigmoid function.

4.1.3 TensorFlow and Inception in TensorFlow

TensorFlow is the second generation machine learning tool developed by Google [47]. DistBelief [48], Google's first generation machine learning tool, was targeted to neural networks and was difficult to configure. For this reason, in November 2015 Google released TensorFlow which is easier to use, more general and in many cases much faster than DistBelief. A program using TensorFlow can be used in a variety of

systems, from mobile devices to GPU cards. It has been used for research as well as for production in many different areas of computer science such as computer vision, speech recognition, natural language processing, etc. It is also used by Google's products such as Gmail, Google Photos and Google search. The numerical computations use data flow (directed) graphs, where each node represents mathematical operations and the values that flow along the edges represent multidimensional data arrays called tensors. There also exist edges in the graph, where no data flow along them, called control dependencies.

Inception [49], is a deep convolutional neural network architecture, Figure 4.5. It is based on [50] where micro neural networks with more complex structures were built to compress the data in the receptive field in an attempt to increase the representational power of neural networks. By stacking these micro neural networks, a deep neural network can be implemented. In [49], a 22 layers deep network called GoogLeNet was used for detection and classification in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14) with great results. In this method, the computer resources inside the network are used in an improved way, and as a result, the depth and width of the network are increased while the computation time is kept constant. A Hebbian principle and multi-scale processing were used for the architectural decisions in order to optimize quality. In [51], further improvements were made. The design principles that seem to improve the model include avoiding bottlenecks, increasing the activations per tile in order to train the networks faster, using spatial aggregation over lower dimensional embeddings and having a balance between the width and the depth of the network. Deviations from these principles seem to give worse results, but this does not mean that they straightforwardly improve the quality of the networks when used.

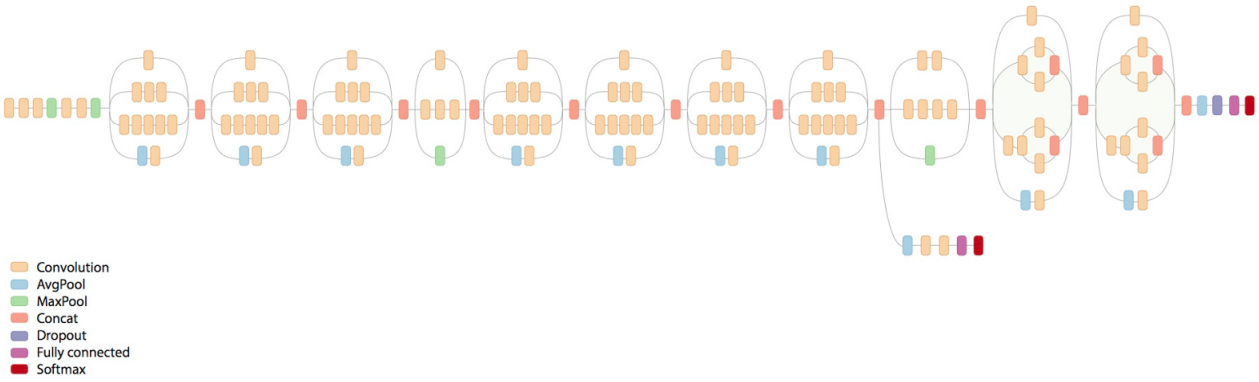

Fig. 4.5. Schematic diagram of Inception-v3 [52]

4.1.4 DarkNet YOLO

Human vision allows us to interpret an image instantaneously. We identify the objects in an image and the interactions between them by looking at the image only once. This is also the idea behind the YOLO (You Only Look Once) system for object detection [53]. The goal is to develop an algorithm as fast and accurate as human vision in order to process real-time scene information. To perform detection, current detection systems alter the use of classifiers, by evaluating a classifier for an object in various locations and scales in an image. R-CNNs for example, propose several bounding boxes and use a classifier on them. Then, the bounding box is refined, the duplicate detections are eliminated and the bounding box is given a new score based on the other objects in the image [54]. These procedures are slow and difficult to optimize since every component is trained separately. In this work, object detection is considered a single regression problem (from pixels to bounding box coordinates to class probabilities). A single neural network is used to predict bounding boxes and the associated class probabilities simultaneously (in one evaluation), Figure 4.6-4.7. This way you only look once to predict the objects and their position in an image, thus, the detection performance is optimized. YOLO can process 45 frames per second while Fast YOLO processes 155 frames per second.

4.2 Our approach

RCN can be divided in three steps. The first step is human detection and extraction of the detected areas. YOLO, a powerful and fast R- CNN implementation is used for that purpose, which is way faster than Faster R-CNN with similar results concerning the accuracy. It is trained with PASCAL VOC12 dataset, see Figure 4.8-4.9. The main problem that human detection algorithms encounter is humans can be partially occluded or out of view.

The next step of our method is feature extraction. For feature extraction, we use Inception v3 with a pre-trained model (on ImageNet dataset). Although we use

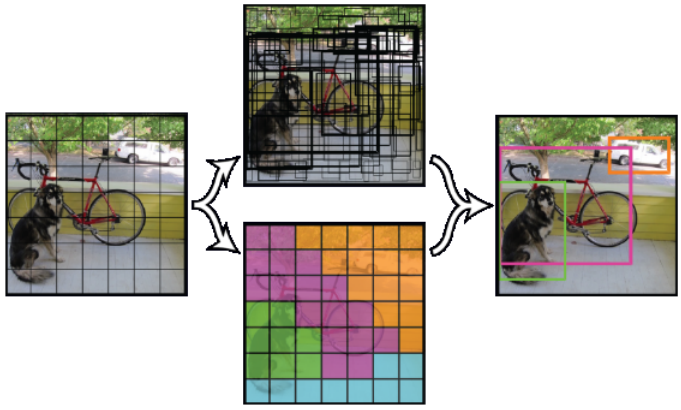

Fig. 4.6. A single NN is applied to the image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities [55].

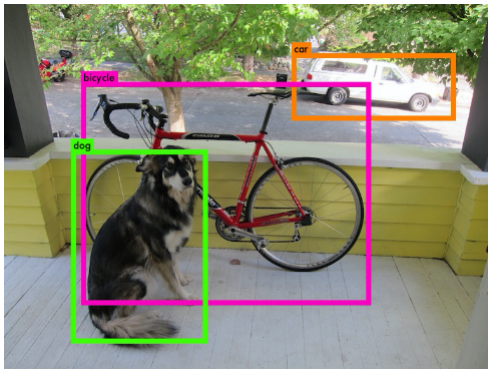

Fig. 4.7. Results after thresholding the detections by some value to only see high scoring detections [55].

Fig. 4.8. Examples of human detection using YOLO algorithm from Hollywood2 dataset. We can observe that YOLO fails to detect the person in the first image. It successfully performs human detection in the other two images.

Fig. 4.9. Examples of human detection using YOLO algorithm from HMDB51 dataset. We can observe that YOLO fails to detect the person in the first image. It successfully performs human detection in the other two images.

many data, they are still not sufficient to train a CNN from scratch as a result we transfer the learning (weights) from the Inception model to ours. The dimension of the Inception model, and hence for our model, is 2048 features for each frame. In our approach, we use the advantages of RNN in order to handle sequential data. We fit our RNN model with the sequenced features that we extracted using Inception and we perform classification with softmax regression. This method combines the strengths of CNNs in visual recognition problems, and RNN in time-varying inputs and outputs. RCN processes the variable-length visual input with a CNN, whose outputs are fed into a stack of recurrent sequence models. That produce a variable-length prediction with shared weights across time, resulting in a representation that scales to long sequences. The method is complementary, CNN are considered as deep in space while RNN deep in time. The architecture of the RCN model is as follows: we pass each visual input x_t which is a frame through a feature transformation φ_V with aprameters V (Inception), to produce a 2048 long vector representation for each frame, $\varphi_V(x_t)$. Then the outputs of φ_V are passed into a RNN. The RNN model has parameters W. The input x_t and a previous time step hidden state h_{t-1} are mapped to an output z_t and an updated hidden h_t. Thus, we run the inference sequentially, we first compute $h_1 = f_W(x_1, h_0)$, then $h_2 = f_W(x_2, h_1)$, etc., where $h_0 = 0$. In order to predict that a distribution $P(y_t)$over outcomes $y_t \in C$ at time t, where C is the set of outcomes, the outputs $z_t$of the RNN are passed through a linear prediction layer $\hat{y}_t = W_z z_t + b_z$, W_z and b_z are learned parameters. The predicted distribution $P(y_t)$ is computed by taking the softmax of $\hat{y}_t$:

$$P(y_t = c) = softmax(\hat{y}_t) = \frac{exp(\hat{y}_{t,c})}{\sum_{c' \in C} exp(\hat{y}_{t,c'})}.$$

The parameters of softmax are optimize using Adam Optimizer [56]. Adam Optimizer is an algorithm for first-order-gradient based optimization of stochastic functions, based on adaptive estimates of lower-order moments. The algorithm is efficient, does not require a lot of memory and is invariant to diagonal rescaling of the gradient. The parameters have intuitive interpretation, thus, there is no reason for tuning. In

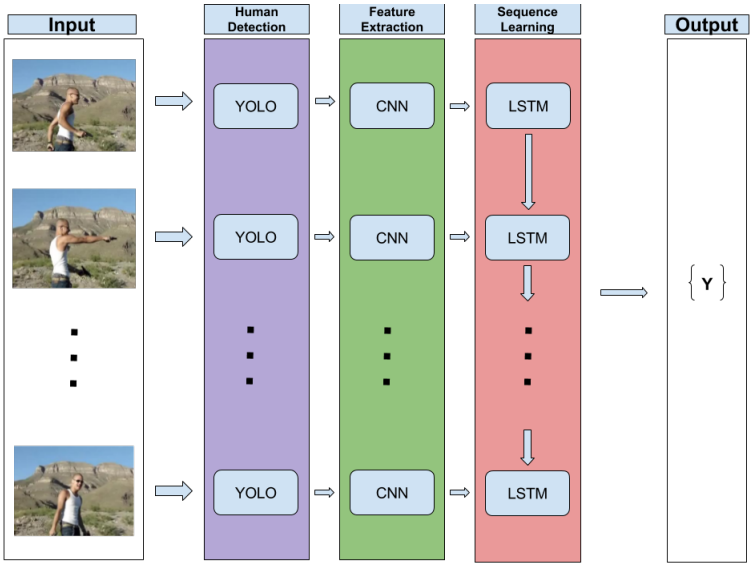

Fig. 4.10. Architecture of our model, RCN. It accepts as input a sequence of raw images, it performs human detection with YOLO and extracts features from bounding boxes using Inception v3. It performs sequence learning using LSTM-RNN. Y is the output, classified images.

our model, we only set the learning rate and the initial forget bias parameter, which is later handled by the algorithm. See Figure 4.10 for the architecture of our model.

5. EXPERIMENTS-RESULTS

5.1 Datasets

We briefly describe the two datasets that we used in our experiments. These datasets are among the most challenging datasets in the literature.

Hollywood2

It has been collected [1] from 69 different hollywood movies. The actions that are included are twelve. It contains 1707 videos split into a training set of 823 videos and a test set of 884 videos. Train and test videos are split on the movies, such that a movie can only appear on either the train or the test set. The dataset includes the actions shown in Table 5.1.

The movies that they used for extracting the clips are the following: Training movies: American Beauty, As Good as It Gets, Being John Malkovich, The Big Lebowski, Bruce Almighty The Butterfly Effect, Capote, Casablanca, Charade, Chasing Amy, The Cider House Rules, Clerks, Crash, Double Indemnity, Forrest Gump, The Godfather, The Graduate, The Hudsucker Proxy, Jackie Brown, Jay and Silent Bob Strike Back, Kids, Legally Blonde, Light Sleeper, Little Miss Sunshine, Living in Oblivion, Lone Star, Men in Black, The Naked City, Pirates of the Caribbean: Dead Man's Chest, Psycho, Quills, Rear Window, Fight Club.

Test movies: Big Fish, Bringing Out The Dead, The Crying Game, Dead Poets Society, Erin Brockovich, Fantastic Four, Fargo, Fear and Loathing in Las Vegas, Five Easy Pieces, Gandhi, Gang Related, Get Shorty, The Grapes of Wrath, The Hustler, I Am Sam, Independence Day, Indiana Jones and The Last Crusade, It Happened One Night, It's a Wonderful Life, LA Confidential, The Lord of the Rings: The Fellowship

Table 5.1.
The 12 classes and the number of train and test samples of the Hollywood 2 dataset.

	Training subset	Test subset
AnswerPhone	66	64
DriveCar	85	102
Eat	40	33
FightPerson	54	70
GetOutCar	51	57
HandShake	32	45
HugPerson	64	66
Kiss	114	103
Run	135	141
SitDown	104	108
SitUp	24	37
StandUp	132	146
All Samples	823	884

of the Ring, Lost Highway, The Lost Weekend, Midnight Run, Misery, Mission to Mars, Moonstruck, Mumford, The Night of the Hunter, Ninotchka, O Brother Where Art Thou, The Pianist, The Princess Bride, Pulp Fiction, Raising Arizona, Reservoir Dogs.

HMDB51

The data set is collected from movies and YouTube videos. There are 51 different actions in total of 6766 video sequences. There are three train-test splits [2]. For every class and split there are 70 videos for training and 30 videos for testing.

The actions categories can be grouped in five types:

General facial actions: smile, laugh, chew, talk. Facial actions with object manipulation: smoke, eat, drink. General body movements: cartwheel, clap hands, climb, climb stairs, dive, fall on the floor, backhand flip, handstand, jump, pull up, push up, run, sit down, sit up, somersault, stand up, turn, walk, wave. Body movements with object interaction: brush hair, catch, draw sword, dribble, golf, hit something, kick ball, pick, pour, push something, ride bike, ride horse, shoot ball, shoot bow, shoot gun, swing baseball bat, sword exercise, throw. Body movements for human interaction: fencing, hug, kick someone, kiss, punch, shake hands, sword fight.

5.2 Results

Both of the algorithms are implemented in Python. We use several python tools: OpenCV, Anaconda, numpy, scikit-learn, etc.. We run YOLO on bash terminal and we use the python wrapper for TensorFlow. We perform the experiments on a OSX with an i7 processor and 8Gb RAM. Although we do not run RCN with the help of a GPU, the time performance is still pretty good. The most time consuming process is YOLO algorithm. It takes about 5 sec to detect a human in one frame. The feature extraction with Inception v3 takes less than an hour and the training of the RNN takes from a couple of hours to a couple of days depending on the configuration. The

Table 5.2.
Accuracy of Dense Trajectories and RCN on Hollywood 2 dataset.

	Hollywood2	HMDB51
Dense Trajectories	59.2	50.3
RCN	52.1	48.7

Table 5.3.
Accurancy of each descriptor on Dense Trajectories.

	Hollywood2	HMDB51
HOG	43.1	35.7
HOF	54.8	43.9
MBH	56.5	48.1

dense trajectories implementation is in general faster. For each dataset we follow the instructions of the authors to perform evaluation of our models [1, 2].

The accuracy of the two methods are comparable, none of the two outperforms the other by far. As we can observe in table 1, dense trajectories implementation has the best accuracy in both datasets. RCN outruns the performance of Dense Trajectories in HMD51 dataset when only one descriptor is used. In Hollywood2, RCN outruns Dense Trajectories with HOG descriptors but both HOF and MBH descriptors outperform RCN.

Table 2 compares our models with state of the art methods. The highest accuracy is achieved by Long-term Temporal Convolutions. They perform a 67.2 % accuracy in HMDB51 and this is the best result to our knowledge. Wang et.al combines the dense trajectories with pool trajectories exctracted by CNN and outperform the accuracy of the simple dense trajectories implementation.

As we can see from the results, the performance of RCN in the two datasets has smaller variance than Dense Trajectories (9 points for Dense Trajectories, 3 for

Table 5.4.
Comparison between state of the art methods and our models on both datasets.

	Hollywood2	HMDB51
DIT	64.3	57.2
RCN	52.1	48.7
Dense Trajectories	59.2	50.3
LTC	-	67.2
Pool Trajectories	-	65.9
CNN flow	-	59

RCN). Unfortunately, we do not know a NN method that has been evaluated with the Hollywood 2.Dense trajectories have better performance in Hollywood2 dataset (59.2% over 50.3% for HMDB51).

We also experimented on combinations of the two methods. For example, we use features we extract from Inception for 15 frames for creating feature encoders using the method we described for dense trajectories, fisher vectors. First, we use PCA for reducing the features from 2048 to 512. Then, we fit a GMM with K=256 with our training examples and calculate the fisher vector for each video. After that, we applied again PCA to reduce the dimensionality from more than 100.000 features to 200. We train a linear SVM with C=100 with accuracy around 28%. We observe that feature encoding methods perform very poorly, we strongly believe that is the result of ignoring temporal characteristics.

As we can see in table 3, the performance of DIT is better than ours although some of the same algorithms are implemented. We ascribe this result to the fact that we are running our method for less frames than in the original paper and we also used PCA after the concatenation of the descriptors. The reason we decided to use 15 frames is to make our algorithm faster and also to perform a more accurate comparison for the two methods we implemented.

We run the RCN with several different configurations. For instance, 1-3 LSTM cells, 128-1024 features in hidden layers, learning rate 0.0001-0.001, different optimization algorithms than Adam Optimizer such as gradient descent. The best performance was achieved with 2 LSTM cells, 256 num of features in hidden layers and learning rate 0.001. We would also like to mention the severe overfitting problem. In many configurations, after a few iterations the model seems to be trained well. The loss is small and the accuracy for each batch is near 100%, nevertheless the performance in the test data is poor (30-40% in Hollywood 2).

5.3 Comparison

In the next paragraphs we will show the advantages and disadvantages of each method for action recognition.

Dense Trajectories are easy to implement. There are many libraries such as OpenCV, where the majority of the algorithms are already implemented. It is generally a fast method and it does not require a lot of samples for training. For example, Hollywood 2 dataset has only 800 video clips. It takes into consideration both motion and spatial features. On the other hand, there is not much left to do in order to improve the results of dense trajectories. The best accuracy using this method was achieved in 2013 and no improvements were made since.

RCN still has room for improvement. The methods used in RCN are either new or recently reintroduced (NN, RNN). Thus, further improvements are expected the following years. Furthermore, RCN takes into consideration both motion and spatial features.It is also more suitable (than other NN methods) for videos since it takes into account the image sequences. The disadvantages of RCN include the big amount of data that is required. For instance, the 800 training samples in Hollywood2 are not sufficient. As a result, overfitting problems are often encountered. For object detection problems, models use more than 1.000.000 images. Also, the parameters are hard to estimate. A small change may cause unpredictable behavior. Documentation is also a principal issue. TensorFlow is a new library, therefore documentation is still at early stages. Google recently opened its source and therefore we expect that documentation will be improved. Finally, a disadvantage of major importance is the computational time. A powerful GPU (CUDA) is required, if not training might take days or even weeks.

6. CONCLUSIONS-FUTURE WORK

This thesis presents two approaches for human action recognition. The first approach is a supervised method for training local features such as HOGs, SURF, etc. For the second approach, we introduce a new model, RCN, which combines three state of the art algorithms namely YOLO, Inception v3 and RNN.

We demonstrate the advantages and disadvantages of each model. The main advantage of supervised learning in local features is the simple implementation, the capability of learning with few data and the low computational cost. But few ameliorations can be made. The main advantage of deep learning methods is the room for improvement with the main disadvantage being the high computational cost. The great amount of data needed causes many problems like extreme overfitting.

The literature review indicates that much research has been devoted to recognition of human activities from videos. This claim certainly holds in cases where local features and space-time volume is used. Deep learning algorithms that use appearance regional or time regional representations increased and recently, new approaches, that combine the two representations, have been developed.

The choice of data play a major role in the action recognition problem. Most popular test datasets are still simple, constrained, and created in structured environments, therefore, most action recognition algorithms can achieve high accuracy. More realistic datasets like Hollywood2 and HMDB 51 have proved to be very challenging and the reported accuracy is still low, which leaves room for further improvements. The increment of applications in different fields will emerge new domain specific techniques and cross domain frameworks that will improve these results.

Future work

As we mentioned above, there is plenty of room for improvements for the RCN method and probably enough space to improve dense trajectories. Initially, the human detector can be improved with the help of faster R-CNN method. Although the computational time will dramatically increase, faster R-CNN seem to have better results regarding the accuracy. As a consequence, the Inception model will improve and more representative features will be extracted. Another variation of our model that will, potentially, further improve the results, is to train the detector (faster R-CNN, YOLO) to detect not only humans but objects that are representative of a class. Detection of phones is most likely to improve the prediction in the "answer phone" class and detection of swords will improve the prediction in the "sword" class since it is rather unlikely that a sword will exist in an image of another class. In addition, although we use sequential learning, we could also use Inception for extracting, not only appearance region representations, but also time region representation with the help of a state of the art optical flow. In [35], an algorithm called Epic Flow is used, showing the importance of a good optical flow. Lastly, RNN is sensitive to variations in parameters. As we mentioned above, even the smallest change of a parameter, such as the number of features for each LSTM, might dramatically change the behavior of the model. Sampling more than 15 frames from each video, might further improve the results of our model since the outliers will be a much smaller percentage of the video representation. Dense trajectories could be improved by using a better optical flow as the one mentioned above, a more suitable algorithm for dimensionality reduction and a more sophisticated method for feature encoding. Finally, there is a plethora of classification algorithms that can be used, kernel SVM, Random Forest, Logistic Regression, etc..

REFERENCES

REFERENCES

[1] M. Marszalek, I. Laptev, and C. Schmid, "Actions in context," in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, 2009, pp. 2929–2936.

[2] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "Hmdb: a large video database for human motion recognition," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[3] A. M. Elgammal, D. Harwood, and L. S. Davis, "Non-parametric model for background subtraction," in *Proceedings of the 6th European Conference on Computer Vision-Part II*, ser. ECCV '00. Springer-Verlag, 2000, pp. 751–767.

[4] T. Ko, S. Soatto, and D. Estrin, "Background subtraction with distributions," in *Proceedings of the European Conference on Computer Vision*, 2008.

[5] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, ser. CVPR '05. IEEE Computer Society, 2005, pp. 886–893.

[6] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, sep 2010.

[7] P. Dollar, S. Belongie, and P. Perona, "The fastest pedestrian detector in the west," in *Proc. BMVC*, 2010, pp. 68.1–11.

[8] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *INTERNATIONAL JOURNAL OF COMPUTER VISION*, vol. 1, no. 4, pp. 321–331, 1988.

[9] G. Sundaramoorthi, J. D. Jackson, A. J. Yezzi, and A. Mennucci, "Tracking with sobolev active contours," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006, pp. 674–680.

[10] A. D. Wilson and A. F. Bobick, "Parametric hidden markov models for gesture recognition." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 9, pp. 884–900, 1999.

[11] P. P. Yang Song, Xiaolin Feng, "Towards detection of human motion," *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 810–817vol.1, 2000.

[12] N. Ikizler and D. Forsyth, "Searching video for complex activities with finite state models," *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 1–8, 2007.

[13] K. Prabhakar, S. M. Oh, P. Wang, G. D. Abowd, and J. M. Rehg, "Temporal causality for the analysis of visual events." in *CVPR*. IEEE Computer Society, 2010, pp. 1967–1974.

[14] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 603–619, 2002.

[15] A. Klaser, M. Marszalek, and C. Schmid, "A spatio-temporal descriptor based on 3d-gradients," in *In BMVC '08*, 2008.

[16] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *ICCV 2013 IEEE International Conference on Computer Vision*. Sydney, Australia: IEEE, Dec 2013, pp. 3551–3558.

[17] H. Wang, A. Klaser, C. Schmid, and C.-L. Liu, "Dense trajectories and motion boundary descriptors for action recognition," *International Journal of Computer Vision*, vol. 103, no. 1, pp. 60–79, May 2013.

[18] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *Proceedings of the 14th International Conference on Computer Communications and Networks*, ser. ICCCN '05. IEEE Computer Society, 2005, pp. 65–72.

[19] M. Bregonzio, S. Gong, and T. Xiang, "Recognising action as clouds of space-time interest points," in *CVPR*, 2009.

[20] P. Scovanner, S. Ali, and M. Shah, "A 3-dimensional sift descriptor and its application to action recognition," in *Proceedings of the 15th ACM International Conference on Multimedia*, ser. MM '07. New York, NY, USA: ACM, 2007, pp. 357–360.

[21] G. Willems, T. Tuytelaars, and L. Gool, "An efficient dense and scale-invariant spatio-temporal interest point detector," in *Proceedings of the 10th European Conference on Computer Vision: Part II*, ser. ECCV '08. Springer-Verlag, 2008, pp. 650–663.

[22] R. Messing, C. Pal, and H. Kautz, "Activity recognition using the velocity histories of tracked keypoints," in *ICCV '09: Proceedings of the Twelfth IEEE International Conference on Computer Vision*. Washington, DC, USA: IEEE Computer Society, 2009.

[23] P. K. Matikainen, M. Hebert, and R. Sukthankar, "Trajectons: Action recognition through the motion analysis of tracked features," in *Workshop on Video-Oriented Object and Event Classification, ICCV 2009*, September 2009.

[24] J. Sun, X. Wu, S. Yan, L. F. Cheong, T.-S. Chua, and J. Li, "Hierarchical spatio-temporal context modeling for action recognition." in *CVPR*. IEEE Computer Society, 2009, pp. 2004–2011.

[25] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 568–576.

[26] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Generic features for video analysis," *CoRR*, 2014.

[27] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," *CoRR*, 2014.

[28] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, pp. 221–231, jan 2013.

[29] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler, "Convolutional learning of spatio-temporal features," in *Proceedings of the 11th European Conference on Computer Vision: Part VI*, ser. ECCV'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 140–153.

[30] L. Wang, Y. Qiao, and X. Tang, "Action recognition with trajectory-pooled deep-convolutional descriptors," in *CVPR*, 2015, pp. 4305–4314.

[31] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, "Sequential deep learning for human action recognition," in *Proceedings of the Second International Conference on Human Behavior Unterstanding*, ser. HBU'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 29–39.

[32] X. Peng and C. Schmid, "Multi-region two-stream r-cnn for action detection," in *ECCV 2016 - European Conference on Computer Vision*, Amsterdam, Netherlands, Oct 2016.

[33] A. Grushin, D. Monner, J. A. Reggia, and A. Mishra, "Robust human action recognition via long short-term memory." in *IJCNN*. IEEE, 2013, pp. 1–8.

[34] G. Lefebvre, S. Berlemont, F. Mamalet, and C. Garcia, "Blstm-rnn based 3d gesture classification," in *International Conference on Artificial Neural Networks (ICANN 2013)*, sep 2013, pp. 381–388.

[35] G. Varol, I. Laptev, and C. Schmid, "Long-term temporal convolutions for action recognition," *CoRR*, 2016.

[36] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR '14. IEEE Computer Society, 2014, pp. 1725–1732.

[37] Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[38] G. Farneback, "Two-frame motion estimation based on polynomial expansion," in *Proceedings of the 13th Scandinavian Conference on Image Analysis*, ser. SCIA'03. Springer-Verlag, 2003, pp. 363–370.

[39] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, jun 1981.

[40] F. Perronnin, J. Sanchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ser. ECCV'10. Springer-Verlag, 2010, pp. 143–156.

[41] Glosser.ca. (2013) Colored neural network. [Online]. Available: https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg

[42] Aphex34. (2015) Conv layers. [Online]. Available: https://commons.wikimedia.org/wiki/File:Conv_layers.png

[43] Aphex34b. (2015) Conv layer. [Online]. Available: https://commons.wikimedia.org/wiki/File:Conv_layer.png

[44] Aphex34c. (2015) Typical cnn. [Online]. Available: https://commons.wikimedia.org/wiki/File:Typical_cnn.png

[45] S. Hochreiter and J. Schmidhuber, "Long short term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, nov 1997.

[46] BiObserver. (2015) Lstm. [Online]. Available: https://commons.wikimedia.org/wiki/File:Long_Short_Term_Memory.png

[47] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *CoRR*, 2016.

[48] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks," in *NIPS*, 2012.

[49] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[50] M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR*, 2013.

[51] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, 2015.

[52] Google. (2015) Inception. [Online]. Available: https://research.googleblog.com/2016/03/train-your-own-image-classifier-with.html

[53] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, 2015.

[54] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, 2013.

[55] J. Redmon. (2016) Yolo. [Online]. Available: http://pjreddie.com/darknet/yolo/

[56] D. P. Kingma and J. Ba, "Adam: Method for stochastic optimization," *CoRR*, 2014.

[57] A. D. Wilson and A. F. Bobick, "Hidden markov models." World Scientific Publishing Co., Inc., 2002, ch. Hidden Markov Models for Modeling and Recognizing Gesture Under Variation, pp. 123–160.