Master of Science in Industrial and Applied Mathematics, Specialization Data Science

Learning action recognition from 3D poses

Erwan Le Roux

February 20^{th} – June 30^{th} 2017

Inria, Rhône Alpes
655 Avenue de l'Europe
38330 Zirst de Montbonnot Saint Martin

Internship supervisor
Gregory Rogez & Cordelia Schmid

Abstract

Human action recognition is a classical problem in computer vision, with applications spanning human-computer interaction, video surveillance, and robotics. Skeleton based action recognition, also called 3D pose based action recognition, has recently attracted increasing attention due to both its succinct view-invariant representation. Following the success of deep learning methods in several computer vision tasks, recent work has focused on using deep recurrent neural networks to analyze skeleton sequences. In this work, after a thorough review of related topics, we examine concurrent deep learning methods. Our model based on recurrent neural networks achieves state-of-the-art performance on the largest dataset for skeleton based action recognition called NTU RGB+D. As a second contribution, we design a dataset with YouTube videos recorded "in the wild" and extract skeleton sequences with a state-of-the-art 3D pose estimation method. Then, we use our model to predict an action label for each skeleton sequence. To the best of our knowledge, we are the first to use skeleton-based action recognition on 3D pose extracted from real videos.

Contents

1 Introduction

Significant progress has been made in computer vision over the past few years (Section 2.1), particularly in image analysis (object classification [41], detection [19] and segmentation [49], image captioning). As examples, Google Photos allows to organize or search photos by people, places or objects, and Facebook automatically describes newly uploaded photos to blind and visually impaired people. Nevertheless, the generalization to videos requires tools to efficiently analyze video content. As explained in [87], automatic video understanding is increasingly relevant as the number and the quality of capturing devices have significantly grown over the past few years with technologies such as smartphones, tablets or action cameras (e.g. GoPro). In 2019, videos are expected to represent 80% of the Internet traffic. Thus, making computers able to understand and interpret massive amount of videos, in particular human actions videos, is a fundamental scientific challenge.

Applications in automation include surrounding environment analysis for autonomous robots (e.g. autonomous cars, drones), video surveillance (e.g. detecting suspicious activities or abnormal behaviors) retrieving and indexing video from both personal cameras and Internet content platforms (e.g. YouTube, Dailymotion). Human-oriented usages include kinesiology, elderly care, human-computer interaction, and entertainment (e.g video games).

Although major progress has been made in human action recognition [28], a key limitation of video based approaches is that they do not fully capture the 3D human motion. These methods are unable to properly handle viewpoint changes and are highly sensitive to illumination variations, partial occlusion and background clutter. On the other hand, 3D pose (i.e. 3D coordinates of human skeleton joints) based action recognition methods are both invariant to viewpoint and insensitive to the context (i.e. background, illumination, clothes).

Several technologies (Section 2.2.1) have renewed interest toward research on skeletal features for human action recognition (see Figure 1). Motion capture system provide location of landmarks placed on the human body with high accuracy, however such systems are expensive and usually of high complexity. In 2010, Microsoft launched Kinect which enables to get a rough human skeleton using infrared lights for 100$. Kinect became the fastest-selling consumer electronics device on record with 8 millions devices sold in 2 months. Recently, advances in deep learning allow to obtain 3D skeleton from monocular images (Section 2.2.2) which would potentially enable to get 3D skeleton from any camera including smartphones.

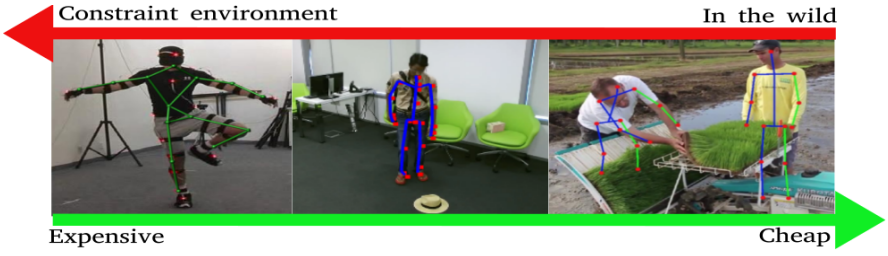


Figure 1: Overview of 3D pose estimation methods (poses only shown in 2D for readability). From left to right: Motion Capture (using infrared red cameras), Kinect (based on a structured light sensor), Monocular images (Figure credit [40] [67] [60])

Skeleton based representation dates back to the seminal work of Johansson [34] which show that humans are capable to recognize actions simply from the motion of the point-light displays attached to the moving subjects. Nevertheless, learning human action recognition from 3D poses is still nowadays far from being solved. The reason for that is the variability of the skeleton sequences both temporally (action speed) and spatially (noisy pose estimation, body shape of the actor).

Above all, 3D pose based action recognition is a challenging task because skeleton sequences consist in a succession of 3D coordinates of human joints, whereas these joints originally lie on a human body subject to various constraints. Indeed, human motion is the result of both physical limitations (e.g. gravity, moment preservation, torque exerted by muscles) and the intentions of subjects [51]. Therefore, due to the high-dimensionality, non-linear dynamics and stochastic nature of human movement, we believe that motion modeling is a complex task that should be ideally learned from observations.

Artificial neural network methods [21] have been successful at learning hierarchical discriminative features efficiently. In our case, human activities can be usually decomposed into three levels of motion complexity [54, 14]. Action primitives are atomic entities out of which actions are built. Then, activities (e.g meet for a drink) are, in turn, decomposed into actions (e.g waving, sitting down, drinking, standing up). Thus, in order to learn action recognition from 3D poses, our goal is to design statistical models, based on artificial neural network, to improve the extraction of discriminative 3D action primitives.

Moreover, thanks to recent breakthrough in 3D pose estimation [60], we believe that the the time is ripe for a video-based action recognition pipeline using a 3D pose sequence as an intermediate representation (see Figure 2). On the comparison to concurrent approaches relying on RGB features [36, 71, 80], such pipeline will not only be invariant to the context (i.e. background, illumination, clothes), but could enable fine grained action recognition. For instance on Figure 2, instead of recognizing solely the action "punching", our method could potentially classify if this punch was a jab, a hook, a cross or an uppercut.

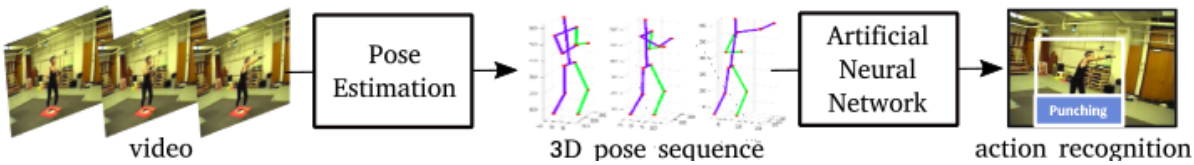


Figure 2: A video-based action recognition pipeline using 3D pose sequence as an intermediate representation. First, an off-the-shelf pose estimation method transforms a video into a sequence of 3D pose. Then, an artificial neural network learns to map 3D pose sequences to their corresponding action label (Figure credit [1]).

After a thorough review of the related topics (Section 2), the contributions of this master thesis are two fold:

1. We exploit the Tensorflow library [2], to experiment with recurrent neural networks (RNN) architectures (Section 3.1), and several regularization methods (Section 3.2) on the largest dataset for skeleton based action recognition called NTU RGB+D [67]. Alongside the key experiments, we present the intuition and the theoretical background to better understand the working mechanisms. Finally, we present two types of preprocessing modules (Section 3.3), which extract static and dynamic features. Thanks to that, **the main objective of the internship, which was to obtain state-of-the-art result for the task of action recognition from skeleton sequence, was reached**.

2. Afterwards, to the best of our knowledge, **we are the first to realize a video-based action recognition pipeline using 3D pose sequence as an intermediate representation**. First, we describe a state-of-the-art 3D pose estimation method [60] and extend it to videos with a temporal consistency mechanism (Section 4.1). Then, in Section 4.2, we design a dataset with YouTube videos recorded "in the wild" (Section 4.2.1). For each video, we extract a skeleton sequence, and estimate its action label by adapting our best model that was trained on skeleton sequences recorded in a constrained environment.

2 Related Work

This related work is organized as follows. After reviewing recent advances in image analysis in Section 2.1, we introduce 3D pose estimation methods in Section 2.2 . Finally, following a presentation of recurrent neural networks in Section 2.3, we describe deep learning based approaches for the task of action recognition from 3D poses in Section 2.4 .

2.1 Image analysis

2.1.1 Revival of neural networks

Artificial neural networks dates back to the 1940s (with the McCulloch-Pitts neuron model [52]) and has endured several downfall in its history. The ILSVRC [13] (ImageNet Large Scale Visual Recognition Challenge) 2012 competition is commonly known as the turning-point in the current resurgence of artificial neural networks. ILSVRC competition's general goal is to estimate the main objects present in images. Training data contains 1000 categories and 1.2 million images collected from Flickr and other search engines, hand labeled with the presence or absence of 1000 object categories.

Before 2012, the ILSVRC challenge was usually won by kernel methods classifying robust handcrafted image features such as SIFT [50]. In 2012, Krizhevsky et al.[41] were the first to use a **CNN** (Convolutional neural network by LeCun et al. [44]) for ILSVRC challenge and they managed to reduce the top 5-error from 25.7% to 15.3% with their architecture called **AlexNet**. Since 2012, only deep learning approaches, i.e. methods using artificial neural network models with many layers, have won this challenge (Table 1). This late success is mainly due to an improvement of GPUs (Graphics Processing Unit) architecture. Indeed, deep learning models' depth is increasing over the years (see Figure 3 for an example) and thus need huge parallel infrastructures to train thoroughly.

Challenge	Architecture Name	Top-5 test error (%)	# of layers
ILSVRC 2011	SIFT + Fisher Vectors (Xerox) [56]	25.7	-
ILSVRC 2012	AlexNet (Toronto University) [41]	15.3	8
ILSVRC 2013	OverFeat (New York University) [66]	11.1	8
ILSVRC 2014	VGG (Oxford University) [72]	6.8	19
	GoogLeNet (Google) [76]	6.7	22
ILSVRC 2015	BN-Inception network (Google) [33]	4.9	31
	Deep Residual network (Microsoft) [27]	3.6	152

Table 1: ILSVRC classification challenge results over the recent years. For deep learning based approaches, we specify the number of layers of the neural network architecture that was used.

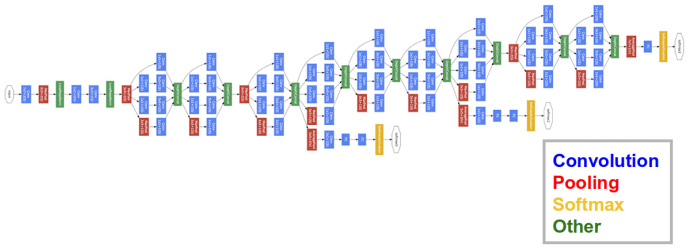


Figure 3: Inception 5 (extension of GoogLeNet [76]) winning architecture for ILSVRC 2014. A classic neural network architecture contains different layers type (Convolution, Pooling, Softmax...) that are combined together in a general computational graph. A layer is a elementary node in this graph and transform the input into an output signal fed to the next layer. The number of layers of a neural network is defined as the depth of the graph. This graph flows from the left to the right: inputs are on the left and outputs are on the right side.

2.1.2 From image classification to image analysis

Instead of using hand-coded images features such as SIFT [50], deep learning models learn adaptive features from the data. Layers' parameters are optimized (by backpropagation [63] combined with a stochastic first order method, e.g. [39]) w.r.t. a task-specific loss function.

To put it simply, CNNs are a non-linear composition of filters learned to extract and combine local patterns from their input. The deeper the filter is, the more abstract is the concept learned. For example, filters applied on the input image (see Figure 4) detect various edges, corners or even some texture and color variations. The first layers of an optimized CNN have been shown to provide a robust feature embedding for images. In other words, instead of learning from scratch an entire neural network, in practice well-optimized neural networks (which are usually ILSVRC challenge best entries) provide a good initialization for the first layers parameters. Afterwards, the following layers learn more abstract concepts from this low-level feature embedding.

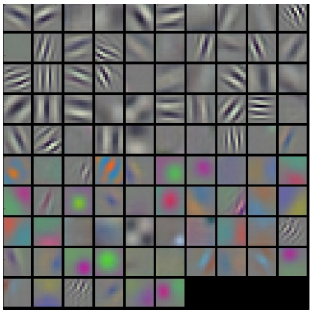


Figure 4: Each element of this grid is a filter from AlexNet's first layer (image credit [90])

This CNN image feature embedding has been both successfully transferred to other classification dataset and even more generally to other tasks. Indeed, computer vision researchers have recently extended CNN to countless other tasks, resulting in state of the art results. Among these successes, Girshick et al. designed R-CNN [19] (a CNN based architecture performing object detection) and Long et al.[49] adapted the CNN to semantic segmentation (task associating a class label to each image pixel). Recently, CNN have also been applied to 2D and 3D pose estimation from monocular images as presented in Section 2.2.2.

2.2 3D skeleton acquisition

Human representations are intensively investigated to address human centered problems such as human detection, tracking, pose estimation, action recognition. Invariant to the viewpoint and to the context, 3D skeleton representations provide a low dimensional encoding of the entire human body configuration.

2.2.1 Direct acquisition settings

1. Motion capture (Mo-cap for short) started as an analysis tool for biomechanics research in the 70s and expanded into sports, military and robotics (see left image of Figure 1 for an example). Recently, as the technology matured, it has been applied to computer animation for movies, and video games . Such systems capture movements of one or more actors many times per second, by identifying and tracking markers that are attached to the actor's joints or body parts. In general, MoCap technologies are expensive, require calibration and can only be used in well controlled indoor environments. There exists two main categories of Mocap system based on either inertial sensor or visual cameras. Optical systems use reflective markers attached to the human body, and produce data with 3 degrees of freedom for each marker. The 3D position of a subject is triangulated between multiple cameras calibrated to provide overlapping projection. On the other hand, most inertial systems use a combination of gyroscope, magnetometer, and accelerometer, to measure rotational of a body part with respect to a fixed point.

2. A structured-light sensor (e.g. Microsoft Kinect v1) consists of an infrared-light source that emits a known pattern, and a receiver that can detect infrared light patterns (distorted due to the object's shape) and allows for a retrieval of the 3D coordinates of the object's surface. Their capability to work in complete darkness are very important advantages in the context of surveillance. Also, time-of-flight sensors (e.g. Microsoft Kinect v2) belong to a class of LIDAR, in which the entire scene is captured with a light pulse switched on for a very short time and objects' distances are resolved based on the known speed of light (see middle image of Figure 1 for an example). Such sensors are affordable, and several algorithms enable to transform in real time the sensor signal into 3D skeleton information. However, since these methods are based on light reflection, they can only work in indoor environments.

2.2.2 3D pose estimation from a single image

Articulated pose estimation is the task that employs computer vision techniques to estimate the configuration of the human body (see right image of Figure 1 for an example). During the past decade, 3D pose estimation received significant attention from the scientific community [64] due to the widening range of applications (e.g human-robot interaction).

3D pose estimation is a challenging task because different 3D poses can result into a similar 2D image projection. 3D pose estimation can also potentially rely on additional cues such as 2D poses. However, minor errors in the locations of the 2D body joints can have large consequences in the 3D space.

Nevertheless, following the current success of deep learning methods in image analysis (Section2.1), neural networks have been employed to address the 2D pose estimation task with great success [79, 11, 78, 10] and only recently the 3D pose estimation task was tackled using deep learning [46, 47, 77, 95].

Indeed, some recent approaches started employing CNNs for 3D pose estimation in monocular images [46, 47] or in video [77, 95]. For instance, Li and Chan [46] exploit a multi-task learning framework, where their CNN network is first trained for the detection task, and then refined using a 3D pose regression task. They show that the network in its last layers has an internal representation for the positions of the left (or right) side of the person, and thus, has learned the structure of the skeleton and the correlation between output variables Furthermore, Li et al. [47] proposed a framework, which takes as inputs a 3D pose and an image, and produces a similarity score value which varies depending on whether these two inputs depict the same pose or not.

Due to the lack of images with ground-truth 3D annotations, these methods are usually trained (and tested) on 3D MoCap data in constrained environments [46, 47, 95]. However, since deep learning methods are "data-hungry", this lack of training data remains a significant obstacle that prevents training model in the wild.

To overcome this lack of 3D data, some works tackle 3D pose estimation from 2D poses assuming that the 2D joints are available [3, 17, 57, 94] or provided by a 2D pose detector [8, 70, 84]. Most of them reason about geometry. Finally, other methods solve 2D and 3D pose estimation jointly or iteratively [69, 89, 93].

Another direction to overcome the lack of 3D data, are the methods of Chen et al. [86] and Rogez and Schmid [59] which propose techniques to synthesize training images with ground truth pose annotations. Both methods [86, 59] train and compare the performance of 2D/3D pose regressors and classifiers in real images, but require a well-aligned bounding box around the subject.

Finally, without any need of well-aligned bounding box, the recent work of Rogez et al. [60] jointly localizes and estimates 2D and 3D pose. They successfully combine classification and regression in a Faster-RCNN [58] like-manner. In Section 4, we will use this state-of-the-art method to extract 3D poses in YouTube videos.

2.3 Deep sequence analysis

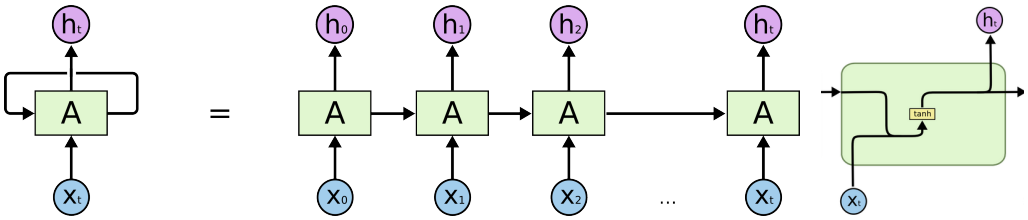
Recurrent neural networks (RNN) [63] are neural networks specialized for processing sequences of variable length. RNN are based on the idea of using past informations for the present task, such as using previous video frames to improve the understanding of the present frame. Essentially any function involving recurrence can be considered as a recurrent neural network.

2.3.1 Standard recurrent neural networks

At time step t, x_t designates the input and h_t the output. Each output is produced using a common update rule. For the standard RNN module we have the following recursion on h_t:

$$h_t = \tanh(W_x x_t + W_h h_{t-1} + b_h), \text{ where } (W_x, W_h, b_h) \text{ are the Standard RNN weights.} \quad (1)$$

This recurrent computation can be visualized with an unfolded computational graph containing a repetitive structure, representing the state of the structure at that point in time (see Figure 5). The unfolded graph now has a size that depends on the sequence length.



(a) General RNN architecture. *(b) Standard RNN module*

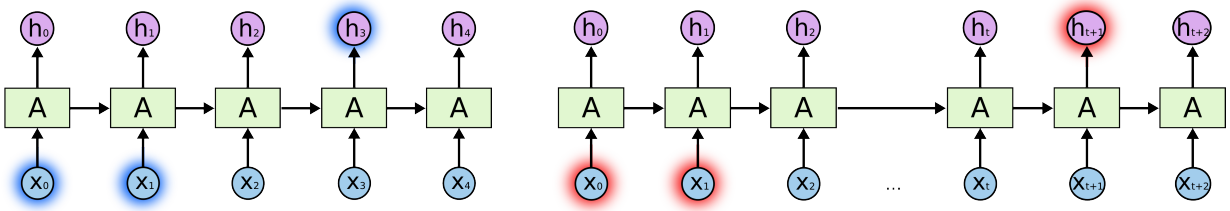
*Figure 5: All RNN can be thought of as multiple copies of the same network, each passing a message to a successor. As shown on Figure 5a, a neural network, **A**, looks at an input x_t and outputs a value h_t. A loop allows information to be passed to the next step. In standard RNN (Figure 5b), this repeating module has a single* tanh *layer, see Equation 1. (image credit [55])*

Using W as the concatenation of all the weights (W_x, W_h, b_h), we rewrite Equation (1) as:

$$h_t = \tanh(W H_t) \text{ with } H_t = \begin{bmatrix} x_t \\ h_{t-1} \\ 1 \end{bmatrix}. \quad (2)$$

Intuitively, a RNN can be imagined as an algorithm where h_t are the algorithm variables values at time t, and the weights W are the instruction modifying the variables between each time step. More precisely, a RNN typically learns to use h_t to selectively keep some aspects of the past sequence of inputs up to t [21]. This selection will depend mainly on the objective function to optimize but could also be influenced by the training algorithm. Backpropagation Through Time (BPTT), which is the adaptation of the backpropagation algorithm [63] to the unrolled computational graph, is currently the standard algorithm for training RNN despite its heavy computational and memory load for very long sequences.

The heuristic solution to this problem is to split the initial sequence into subsequences and only backpropagate on the subsequences. However, such solution, called Truncated BPTT, comes at the cost of losing long term dependencies (see Figure 6). In theory, any RNNs trained with a full BPTT are absolutely capable of handling such long-term dependencies. Indeed, it has been showed [68] that there exists finite RNNs that are Turing complete, and can therefore implement any algorithm. Sadly, in practice, standard RNNs do not seem to be able to learn them [6].



(a) Short term dependencies *(b) Long term dependencies*

Figure 6: For many tasks, RNNs need to capture long term dependencies (image credit [55])

2.3.2 Long short term memory networks

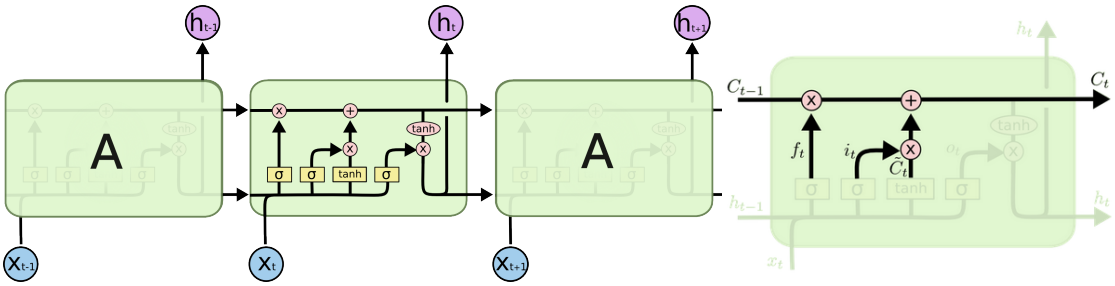
Long Short Term Memory (LSTM) networks [30] are a special kind of RNN, explicitly designed to avoid the long-term dependency problem. Let σ refers to the sigmoid function, $\odot$ designates the Hadamard product, then the recursion equations of the LSTM are the following:

$$\begin{aligned} \begin{bmatrix} f_t \\ i_t \\ o_t \end{bmatrix} &= \sigma\left(\begin{bmatrix} W_f \\ W_i \\ W_o \end{bmatrix} H_t \right) \\ \tilde{C}_t &= \tanh(W_c H_t) \qquad \text{, where } (W_f, W_i, W_o, W_c) \text{ are the LSTM weights.} \\ C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \\ h_t &= o_t \odot \tanh(C_t) \end{aligned} \tag{3}$$

One aspect of LSTM networks is the role of the gates f_t, i_t, o_t. The forget gate f_t can delete parts of the previous memory cell C_{t-1} whereas the modulated input $\tilde{C}_t$ can write new content to the new memory cell C_t modulated by the input gate i_t. The output gate o_t controls what is then read from the new memory cell C_t onto the hidden vector h_t.

LSTM have two important learning properties [35]. Each memory cell vector is obtained by a linear transformation of the previous memory cell vector and the gates (see Figure 7); this ensures that the forward signals from one step to the other are not repeatedly squashed by a non-linearity such as tanh and that the backward error signals do not decay sharply at each step, an issue known as the vanishing gradient problem [29]. The mechanism also acts as a memory and implicit attention system, whereby the signal from some input x_t can be written to the memory vector and attended to in parts across multiple steps by being retrieved one part at a time.

During the last decade, LSTMs have been showed to excel in a wide range of sequence learning domains such as speech recognition [24], machine translation [75] and image-to-caption generation [83].



(a) Long Short Term Memory (LSTM) module *(b) Memory mechanism of the LSTM*

Figure 7: A LSTM module is depicted on Figure 7a. Based on the previous output h_{t-1} and an input x_t, such network modifies its memory cell state c_t, and output a value h_t. As highlighted on Figure 7b, the memory cell state c_t runs straight down the chain, with only some minor linear interactions. Information can easily flow along it unchanged. (image credit [55])

2.3.3 Gated recurrent unit

Recently, a simplification of LSTM, called Gated Recurrent Unit (GRU) [12], has recently shown promising results in various domains. The main difference with the LSTM is that a single gating unit z_t simultaneously controls the forgetting factor and the decision to update the state unit, as shown below:

$$\begin{aligned}
\begin{bmatrix} r_t \\ z_t \end{bmatrix} &= \sigma\left(\begin{bmatrix} W_r \\ W_z \end{bmatrix} H_t\right) \\
\tilde{h}_t &= \tanh\left(W_h \begin{bmatrix} x_t \\ h_{t-1} \odot r_t \\ 1 \end{bmatrix}\right) \text{, where } (W_r, W_z, W_h) \text{ are the GRU weights.} \\
h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t
\end{aligned} \tag{4}$$

The update gate z_t can choose to copy the state vector h_t (at one extreme of the sigmoid) or completely ignore it (at the other extreme) by replacing it by a new target state $\tilde{h}_t$. The reset gate r_t controls which parts of the state get used to compute the next target state $\tilde{h}_t$.

All RNNs described up to now only capture information from the past, even though in many applications we may have to look far into the future (and the past) to disambiguate between all possible interpretations [21]. Bidirectional RNN [65], as the name suggests, combine a RNN that moves forward through time beginning from the start of the sequence with another RNN that moves backward through time beginning from the end of the sequence.

Finally, so far we only presented single layered RNNs which have a limited learning capacity. Deep recurrent neural network were introduced by Graves et al. [24] and simply consists of feeding the output of each RNN layer as input to the next RNN layer. This stacking of RNN layers enable to learn more abstract concepts from the sequences, and is a key ingredient of the wide adoption of Recurrent neural network as a major tool to process data sequences.

2.4 Deep action recognition from 3D pose sequences

During the past decade, human action recognition using 3D skeleton information has been widely explored. In this section, we limit our review to methods based on deep learning, which are currently by far the most successful approaches. We refer our readers to the recent arXiv review [26] for more details on earlier works.

Acquisition of 3D action recognition datasets used to be time-consuming due to the complex setting needed by MoCap methods. Early works were usually assess on "small" dataset with a limited number of actor and action classes. Since neural networks easily overfit on small datasets, early deep learning methods [16] had to reduce the capacity of their architecture to avoid overfitting.

To overcome this limitation, in 2016, Shahroudy et al.[67] introduced the NTU RGB+D dataset. This dataset, collected with a Kinect v2, contains 56 thousand video samples and 4 million frames, collected from 40 distinct subjects. Each 3D skeleton representation (see Figure 8) is captured from 3 different cameras. Actions are captured from several camera settings, and involve 60 different action classes including daily and mutual actions (Figure 9). Scores are measured by the accuracy (in %) of 3D pose sequence correctly classified. Two evaluation methods were initially introduced [67], but we choose to only use the cross subject evaluation setting (half of the subject are used for training, the other half for testing). See Table 2 for a benchmarking of deep learning based approaches.

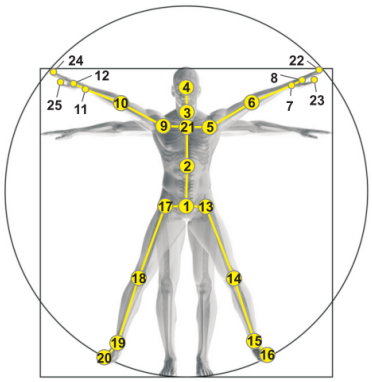
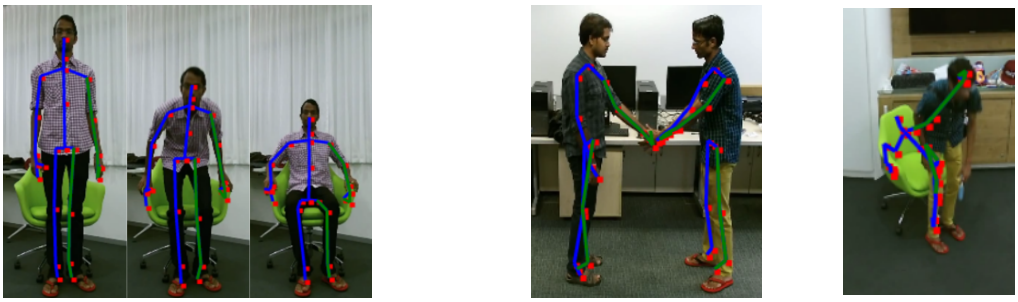


Figure 8: Disposition of the 25 body joints of the NTU RGB+D dataset (image credit [67])

Method		Accuracy (%)	Neural network type	Approach	Year
HBRNN-L	[16]	59.07	Recurrent	Joint-based	2015
LieNet-3Blocks	[32]	61.37	Fully Connected	Embedding-based	2017
Part Aware LSTM	[67]	62.93	Recurrent	Joint-based	2016
Spatio Temporal LSTM	[48]	69.2	Recurrent	Joint-based	2016
Geometric features	[92]	70.26	Recurrent	Joint-based	2017
STA-LSTM	[73]	73.4	Recurrent	Joint-based	2016
Res-TCN	[38]	74.3	Convolutional	Joint-based	2017
Joint Trajectory Maps	[85]	76.32	Convolutional	Embedding-based	2016
Pose-conditioned STA	[5]	77.1	Convolutional	Joint-based	2017
View Adaptive LSTM	[91]	79.2	Recurrent	Joint-based	2017
Clips + CNN + MTLN	[37]	**79.57**	Convolutional	Embedding-based	2017

Table 2: Benchmarking of deep learning approaches for 3D pose based action recognition, evaluated on the NTU RGB+D dataset [67] in the cross subject evaluation setting



(a) Sit down *(b) Handshake* *(c) Pick up*

Figure 9: NTU RGB+D [67] contains 3D pose sequences (shown in 2D for readability) including individual (Figure 9a), and mutual actions (Figure 9b); and also some failure cases (Figure 9c).

Existing deep learning approaches for skeleton-based action recognition can be broadly divided into two main groups depending on how an input sequence is represented. Inspired by the classical moving lights display experiment by Johansson [34], joint-based approaches consist in learning human motion models from 3D joints sequences. On the other hand, embedding-based approaches, first map each skeleton sequence into an embedding space, then feed this novel representation as input to a neural network.

2.4.1 Embedding-based approaches

Similarly to video based action recognition methods, e.g. [7], the first category of embedding-based approaches [15, 37, 85] encodes a video as one or multiple compact RGB images. This simple idea turns out to be quite powerful since available trained CNN architectures [41, 76, 27] can be exploited for fine-tuning. Nevertheless, despite good accuracies, these methods are unsatisfactory, since they require to manually design an encoding that would potentially limit the amount of encoded information.

A second category of embedding, introduced by Vemulapalli et al. [82], represents a skeleton as point on a Lie Group. First, a skeleton can be represented as a graph where the vertex are the N body joints, and the edge are the M oriented rigid body bones connecting the body joints. Therefore, following rigid body kinematics, they represent a skeleton by all the rotation matrix between any two pairs of oriented rigid body bones, i.e. by a point in $SE(3)^d$ with $d = 2\binom{M}{2}$ (where $SE(3)$ is the special Euclidean group). The set $SE(3)^d$ is a Lie Group, and therefore any skeletal sequence can be represented as a curve on a Lie Group.

However, classical machine learning tools require a distance to be defined in the feature space. The main limitation of this seminal work [82] is that, in order to use machine learning tools , they had to map every point of the curved manifold $SE(3)^d$ to an Euclidean space.

In a recent work, Huang et al. [32] generalize neural networks to non-Euclidean Lie groups to handle the problem of skeleton-based action recognition. This work follows the recent emergence of deep learning models that deal with data in non-Euclidean domains [9]. Despite their strong theoretical motivations, accuracies scores are limited. One possible reason is that by embedding each skeleton separately in a curved manifold, the curve in a Lie Group becomes a simple succession of skeleton states. Consequently, all the information about the relative displacement of the body into the scene is lost.

2.4.2 Joint-based approaches

The first category of joint-based approaches, represents a sequence of skeleton data as a $T \times N \times C$ tensor, where T is the length of the sequence, N the number of joints and C the number of channels. Usually $C = 3$, since the channels mostly consists of the joints 3D coordinates, but speed and acceleration can also be added as channels, as done in [5]. Each CNN layer scans its input tensor along the temporal and the joint dimension, with filters optimized by backpropagation to extract dicriminative patterns. Such methods take advantage of the Fully Convolutional architecture [49] that enable to process tensors of variable size. Many approaches capitalize on recent findings in video-based action recognition. For instance, Kim and Reiter [38] introduce Res-TCN a Residual [27] version of Temporal Convolution Networks [43] for 3D pose base action recognition.

These methods are powerful, robust to variable duration of the action (thanks to Pooling layers), and enjoy efficient training procedures. Nevertheless, even though pooling layers and convolutional layers guarantees a large temporal receptive field for each time step, CNNs struggle to capture very long range interaction. In fact, they theoretically could capture such interaction by augmenting the size of temporal receptive receptive field of the output units. However, this would be at the computational cost of augmenting the size of the filters, or adding more layers to the network.

The second category of joint-based approaches, considers sequence of skeleton data as a sequence of length T, containing vector of size $N \times C$ (where N is again the number of joints, and C the number of channels). On the contrary to CNNs, even RNNs with a single layer are theoretically capable of capturing very long range interaction. In the context of 3D pose based action recognition, a RNN parses the input sequentially and encodes the frame-level information in their memory. In this review, we separate contributions into three main groups: skeleton preprocessing, selective neural network modules specialized for skeleton sequences, and finally neural network architecture expressly designed for 3D pose sequences.

- **Preprocessing**. Due to the lack of clear successful preprocessing for 3D pose sequence, View Adaptation Network [91] let a RNN network regulates by itself the observation viewpoints to the suitable ones with the optimization target of maximizing recognition performance. In practice, at each time step, they use a second RNN network to predict the rotation and the translation used for normalizing the 3D coordinates.

  Otherwise, Zhang et al. [92] propose to not limit the input of RNNs to 3D coordinates and pre-compute geometric features over the skeleton. They explore the relations (distances, angles) between body joints, bones, and planes defined by two successive bones.

- **Selective modules**. For many actions only a subset of the joints are discriminative, and actions only happen in a few key-frames. Song et al. [73] exploit two RNNs to generate soft-attention weights [4, 25] both spatially and temporally to automatically select dominant joints, and most important frames.

  Finally, we present two approaches [81, 96] introduced before NTU RGB+D dataset's creation. First, Zhu et al. [96] leverage on the intuition that co-occurrence of joints is a

discriminative feature for many human actions (e.g. answer phone, clapping, take off a shoe). They added a mixed-norm regularization term to their RNN loss in order to push the network to learn the mappings between co-occurring joints and human actions.

Then, Differential LSTM [81] adds a gating inside LSTM to explicitly keep track of the memory states's derivatives (which intuitively represent the long short-term change in the information gained over time) to discover patterns within salient motion patterns.

- **Architecture**

The main intuition behind the seminal work of Du et al. [16] is that human actions are composed of the movement of the five main parts of the body. Thus, skeletons are split into anatomically-relevant parts (legs, arms, torso, etc), so that each first layers gets specialized on one part. Then, features are progressively merged as they pass through successive layers (see Figure 10). Also, they exploit a bidirectional recurrent neural network (BRNN) to learn features from past and future movements of the body parts.

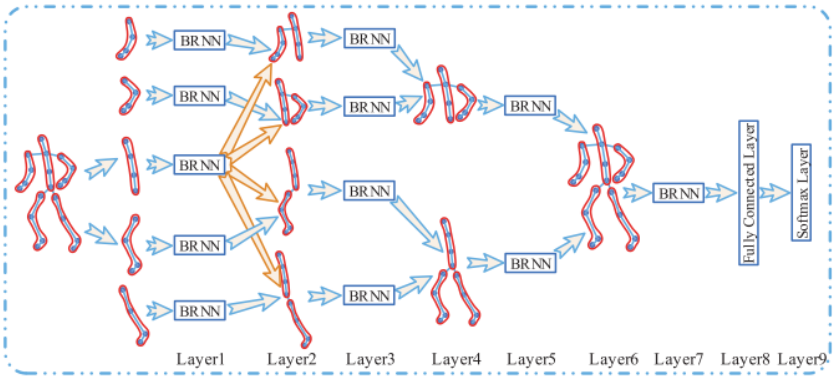


Figure 10: BRNN layers are fused w.r.t the anatomical hierarchy. (Figure credit [16])

In part-aware LSTM [67], body joints are grouped anatomically similarly as [16]. Then, the LSTM memory cell is separated into part-based sub-cells. This motivates the network to learn a long-term representation for each body part.The final prediction is produced by fusing all the part-based memory cells and applying a common output gate.

As a particular case of the Multidimensional RNN introduced by Graves et al [23], Liu et al. [48] model both temporal and spatial dependencies within a single LSTM by applying a recurrent analysis over spatial and temporal domains concurrently. Time has a natural ordering, however they needed to define a spatial ordering for the joint of the skeleton. They model the human skeleton as a tree, and define a spatial chain over the joints. This chain will pass twice through each joint. This enables each joint to potentially get information from both its descendants and ancestors. To sum up, in their model each joint is a unit of the Spatio-Temporal LSTM (ST-LSTM) which receives contextual information from the previous joint and also from previous frame to encode the spatio-temporal context. Also, they introduce a new gating mechanism, called trust gate, to improve the robustness of RNN against noise and occlusion.

3 Improving action recognition from skeleton sequences

The first objective of this internship was to implement and train recurrent neural network (RNN) architectures that match or outperform the state-of-the-art in skeleton-based action recognition and understand the influence of RNNs parameters.

Since previous work did not open source any implementation, we decided to build our own machine learning pipeline with the deep learning library Tensorflow [2]. We constructed a modular organization in Python to preprocess skeleton sequences, train various RNN models, and visualize their performance thanks to Tensorboard [2].

As shown in Table 3, we manage to meet the objective of this internship. Also, currently, among methods using recurrent neural network, we currently hold the state-of-the-art performance. However, a paper published on arXiv at the end of April [45], outperforms us, with a more complex architecture largely inspired from the Two Stream Convolutional Network [71] architecture.

Method	Accuracy (%)	Neural network type	Month
STA-LSTM [73]	73.4	Recurrent	Nov 2016
Res-TCN [38]	74.3	Convolutional	April 2017
Joint Trajectory Maps [85]	76.32	Convolutional	Dec 2016
Pose-conditioned STA [5]	77.1	Convolutional	Mars 2017
View Adaptive LSTM [91]	79.2	Recurrent	Mars 2017
Clips + CNN + MTLN [37]	79.57	Convolutional	Mars 2017
Ours	79.58	Recurrent	June 2017
Two Stream CNN [45]	83.2	Convolutional	April 2017

Table 3: Accuracy of our best model compared to concurrent deep learning based approaches for skeleton action recognition. Models are evaluated on NTU RGB+D dataset [67] in the cross subject evaluation setting

In the following sections, rather than an exhaustive listing of the composition of our best RNN, we focus on detailing the main theoretical motivations behind our implementation. Also, we justify experimentally our best RNN architecture (Section 3.1), its regularization (Section 3.2), and the modules added to extract better 3D action primitives (Section 3.3).

3.1 Recurrent neural network for 3D pose action recognition

3.1.1 Baseline settings

As commonly done in machine learning, the original train set of the NTU RBG+D dataset, created by Shahroundy et al. [67], needs to be split into a train and a validation set for training. Shahroudy et al. explain that their validation set contains 5% of the original train set. However, since they created this validation set by sampling randomly, they were not able to specify us exactly their validation set . In machine learning, it is well known that a validation set characteristics should match the test set characteristics in order to provide a reasonable estimate of the generalization error.

For this reason, in the cross subject evaluation setting, we designed ourselves a validation set composed of subjects not present in the train set (the set used to train our model). More precisely, for reproducibility of our setting, the validation set should contain actors with subject id 25, 27, 28, 31, 34, 35 and 38. Therefore, in this setting, the size of the validation set is equal to half the size of the test set.

Also, as explained earlier, NTU RBG+D dataset was acquired with a Kinect v.2 sensor which can occasionally fail to detect the joints of the actor. In order, to train on data as close as possible to the real-time running data of the Kinect v.2 sensor, Shahroudy et al. explain that they do not preprocess the skeleton sequences of their dataset. We found such approach unsatisfactory. Therefore, when there is a skeleton sequence containing Kinect failures, we exploit a cubic spline curve on each joints coordinates to roughly fill out the missing data.

Our baseline model is the same as a baseline model of the state-of-the-art Recurrent neural network on skeleton data [91]. In terms of preprocessing, the skeleton sequence coordinates are translated such that the torso joint of the initial skeleton lies at the origin. This model is a three layered LSTM (100 units for each layer). Then, a Fully Connected (FC) layer with rectified linear unit (ReLU) is followed by a softmax layer (see Figure 11 for more details).

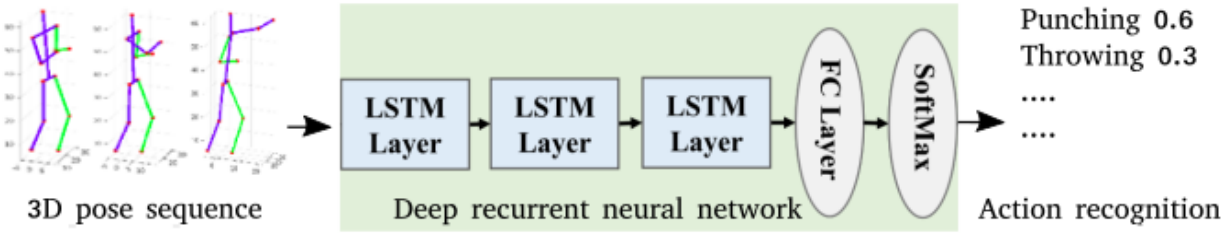


Figure 11: Our baseline model comes from Zhang et al. [91] and starts with 3 LSTM layers. Then, the output vector of the final LSTM layer at the final time step is multiplied by a weight matrix and passed element-wise through a rectified linear unit (ReLU) in a Fully Connected (FC) layer. This produces a score for each action label. Finally, a Softmax layer enables to transform these scores into probabilities for each action label.

Let x_i refers to a skeleton sequence and y_i its corresponding action class. Following the usual supervised learning setting for classification, at each training iteration we sample a mini batch of size $B = 256$. Then, to train our model, we exploit the cross entropy loss:

$$L(y_{1:B}, x_{1:B}) = \frac{1}{B}\sum_{i=1}^{B} -\log p_i^y \text{ ,with } p_i = f_\theta(x_i). \tag{5}$$

where p_i^c is the predicted probability of the model f parametrized by θ for the action class c with respect to input x_i. At inference, for an input x_i the predicted action class is $\text{argmax}_c\, p_i^c$.

Following, common deep learning practice, during training this cross entropy stochastic objective function is minimized with a first-order gradient-based optimization algorithm. At iteration t, let θ_t refers to the parameters values, g_t to the stochastic gradient and η to the initial learning rate. Then, for example $\theta_{t+1} = \theta_t - \eta \times g_t$ is the most simple first-order method known as gradient descent. In our case, we exploit a slightly more complicated approach, called Adam [39], which belongs to the family of first-order optimization methods that computes an adaptive learning rates for each parameter (see [62] for more details).

Adam keeps an exponentially decaying average of past gradient m_t and of past squared gradients v_t which are respectively estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients. However, in order to counteract the bias of these moments toward zero, Kingma and Ba [39] compute bias-corrected first and second moment estimates $\widehat{m_t}$, $\widehat{v_t}$ with the following formula:

$$\begin{aligned} \widehat{m_t} &= \frac{m_t}{1-\beta_1} \text{ where } m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \\ \widehat{v_t} &= \frac{v_t}{1-\beta_2} \text{ where } v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2. \end{aligned} \tag{6}$$

Then we use these estimates $\widehat{m_t}$ and $\widehat{v_t}$ to update the parameters as follows:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\widehat{v_t}} + \epsilon}\widehat{m_t}. \tag{7}$$

In our implementation, for the Adam optimizer we use $\epsilon = 10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and an initial learning $\eta = 5 \times 10^{-3}$

Also, **for this baseline model, we use several regularization methods**. In section 3.2, we describe and experiment on these powerful regularization methods.

Finally, when RNN are trained on long sequences, the gradient becomes unstable and can sometimes either explode or vanish [29]. This instability is a fundamental problem for gradient-based learning in deep neural networks. To avoid exploding gradients, we use gradient-clipping which normalizes the gradients of a parameter vector when its L^2 norm exceeds 1.

3.1.2 Recurrent neural network architectures

A major challenge lies in the initialization of neural networks weights. If the weights start too small (resp. too large), then the signal shrinks (resp. grows) as it passes through each layer, and becomes useless. To tackle this problem, we exploit Xavier initialization [20] which consists of drawing the weights from a distribution with zero mean and a specific variance. This specific variance depends of the number of neural network units in the previous and the next layer and helps signals reach deep into the network. Also, GRU units reset and update gates are initialized with a bias of 1 since it has been shown to improve memorization.

As an initial experiment, we modify the baseline (Figure 11) by changing the type and the number of RNN units. We use either Unidirectional or Bidirectional RNN. To make a fair comparison in Table 4, in both cases the model prediction only takes into account the last time step of the last RNN layer, and the number of units is equal (in the bidirectional case, each directional RNN gets half the number of units that was in the Unidirectional RNN). Then, with our best result we study the influence of the number of RNN layers in Table 5.

	RNN Units / RNN type	100	200	400	800
Unidirectional RNN (URNN)	Standard RNN	43.58	48.98	59.5	60.3
	LSTM	76.05	76.82	76.22	75.98
	GRU	**79.2**	79.03	78.83	79.14
	RNN Units per Direction / RNN type	50	100	200	400
Bidirectional RNN (BRNN)	Standard RNN	40.91	54.82	54.9	66.54
	LSTM	73.82	76.67	77.13	77.35
	GRU	73.9	78.37	**78.71**	78.1

Table 4: Ablation study of RNN architectures. In general, URNN performs better than BRNN. In all cases, GRU beats LSTM which in turns outperforms Standard RNN. Also, as we augment the number of units, the performance of Standard RNN improves, and stays roughly the same for GRU and LSTM (except for BRNN with 50 units where the performance is diminished).

Layers	1	2	3	4	5	6	7	8
Accuracy (%)	72.35	77.09	**79.2**	77.94	77.83	77.2	76.72	74.62

Table 5: Study of RNNs' number of layers. We use a URNN with 100 GRU units. An architecture with 3 layers brings the best score. If we either decrease or increase the number of layers, the accuracy will progressively diminish. However, this statement should be moderated, since that all baseline hyperparameters (e.g initial learning rate) were specially chosen for an architecture with 3 layers.

From now on, in all our experiments, we use the best model, i.e. a Unidirectional RNN with 3 layers each containing 100 GRU units. We will refer to it as "improved baseline".

3.2 Recurrent neural network regularization

Since deep models have many parameters, they have a high degree of expressive power, and are capable of significantly overfitting. Regularization refers to methods that discourage complex models since they are unlikely to generalize well.

Dropout [74] provides a computationally inexpensive but powerful method of regularization. Dropout involves removing randomly a unit from a network by multiplying its output value by zero. Therefore, dropout is usually parametrized with a "dropout keep probability". We can notice that if this probability is equal to 1, none of the units are dropped, and this tantamounts to not using any dropout. In Table 6, we study the influence of dropout keep probability on our improved baseline performance.

Intuitively, dropout is powerful because by training a single base network, it implicitly trains an ensemble (i.e. a combination) consisting of the exponential number of sub-networks that can be formed by removing units from this base network. Also, when evaluating the neural network's performance, it has been shown in [18] that dropout can be used as a practical tool to obtain uncertainty estimates of its accuracy.

Dropout keep probability	1	0.9	0.7	0.5	0.3	0.1
Accuracy (%)	75.17	76.93	78.11	**79.2**	72.91	57.72

Table 6: Influence of dropout keep probability on our improved baseline. We notice that this parameter is quite sensitive. Using, either not enough or too much dropout can dramatically diminish the accuracy.

Tikhonov regularization, also known as L^2 regularization, consists in adding to the objective function a term $\Omega(\theta) \propto \|\theta\|_2$, where θ designates the model's parameters. First, L^2 regularization can be interpreted as specifying that the prior distribution over the model parameters $p(\theta)$ should follow a normal distribution $\mathcal{N}(0, I)$ [21]. A second possibility is to view L^2 regularization as a strategy that drives the parameters θ closer to the origin (see Figure 12).

Also, when training deep learning models with a lot of capacity, the training error decreases steadily over time, but after some time validation error begins to rise. This means we can potentially obtain a model with better generalization error by stopping the training when the error on the validation set has not improved for some amount of time. This strategy is known as early stopping. In fact, as shown in Figure 12, similarly to L^2 regularization, early stopping can be viewed as an implicit spatial constraint on the model's parameters [21].

In our implementation, we use a learning rate decay scheme in combination with early stopping. When the validation error does not improve for some amount of time, we decrease the learning rate by a factor of 10 until reaching a minimal learning rate where we stop training. Such learning rate scheduling generally enables to reduce the validation error further.

In Table 7, we study jointly the influence of early stopping and L^2 regularization.

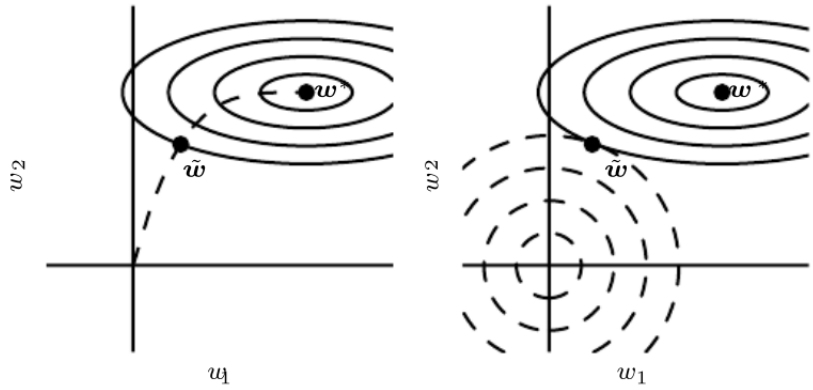


Figure 12: ***In this figure,*** w_1 ***(resp.*** w_2***) represents the model's parameters*** θ_1 ***(resp.*** θ_2***).*** *Solid ellipses represent the level set of the unregularized objective. On the right, the dotted circles represent level sets of the L^2 regularizer. At the point $\tilde{w}$, these competing objectives reach an equilibrium. On the left, the dashed line indicates the trajectory taken by w with Stochastic Gradient Descent beginning from the origin. Rather than stopping at the point w that minimizes the cost, early stopping stops at an earlier point $\tilde{w}$. (Figure credit [21])*

Early stopping / L^2 regularization	Without	With
Without	68.12	72.29
With	76.02	**79.2**

Table 7: Influence of L^2 regularization and early stopping on the improved baseline. We save a checkpoint of each model every 500 iterations, and only report the best score. L^2 regularization and early stopping both individually provide improved performance. Also, when combined they further boost our model's accuracy.

We conclude the two previous subsections with a take home message on how to train RNNs for skeleton-based action recognition.

1. Preprocess (align, scale) skeleton sequences to have normalized inputs. In our case, following [91], we only translate the skeleton sequence w.r.t the torso joint of the first skeleton, which is quite unsatisfactory. Finding such optimal preprocessing is still an open problem in skeleton-based action recognition.

2. Design a RNN architecture (Unidirectional/Bidirectional, type and number of units, number of layers) and train it with a small learning rate and without any regularization. If this model can overfit the training set, it means it has enough capacity. Otherwise, the number of units should be increased or more RNN layers should be stacked.

3. Search for the optimal hyperparameters (e.g. initial learning rate, dropout keep probability, L^2 regularization weight in the objective function). The learning rate is the most sensitive hyperparameter. Finally, finetune early stopping by trying different learning rate decay schemes.

3.3 Learning 3D action primitives

In this part, our aim is to design neural network layers that improve the classification accuracy of our improved baseline, i.e that enable to extract more discriminative 3D action primitives. In this part, a skeleton is viewed as a set of 3D joints, and can be described at time step t with P_t a $N \times 3$ matrix that represents the 3D coordinates of N joints constituting the skeleton.

3.3.1 Skeleton transformer

Instead of feeding the raw set of joints as input to our network. C.Li et al [45] propose to build virtual joints where each virtual joint is a **linear combination** of the original joints. Then, P'_t the $N' \times 3$ matrix that represents the 3D coordinates of N' virtual joints at time t, can be computed at each time step t with the following equation:

$$P'_t = WP_t \tag{8}$$

This module, called skeleton transformer, can be implemented with a fully connected layer (without bias), and optimized end-to-end with the rest of the network. We can notice that the matrix of weight W is shared across all the times steps t, and acts as a common pre-processing step before the RNN layers.

After the matrix multiplication, we propose to divide the i-th virtual joint coordinates by $\sum_j W_{i,j}$. Thus, each virtual joints is an **affine combination** of the original joints.

$$P'_t = WP_t \text{ such that } \forall i, \sum_j W_{i,j} = 1 \tag{9}$$

However, an unconstrained combination of the initial joint seems like an unsatisfactory solution. By adding a Lagrangian constraint $-\lambda \min(0, \min_{i,j} W_{i,j})$ with $\lambda > 0$, we can implicitly constrain all the weights $W_{i,j}$ to be positive. Also, if we force the column of our weight matrix to sum to 1, the virtual joints become a **convex combination** of the original joints.

$$P'_t = WP_t \text{ such that } \forall i, \sum_j W_{i,j} = 1 \text{ and } \forall (i,j), W_{i,j} \geq 0 \tag{10}$$

When we use such convex combination, we can imagine the virtual joints as 3D points that are located inside the 3D convex hull of the original joints.

Furthermore, following the Occam's razor rule which states that we should use simpler model class in order to promote generalization, we might want to use regularization on the weight matrix W. For instance, we could use L^2 regularization strategy which drives the parameter closer to the origin. Another option would be to use L^1 regularization which results in a solution that is more sparse. In this context, sparsity implies that some parameters will have an optimal value equal to zero. In machine learning, this L^1 regularization has been used extensively as a feature selection mechanism.

In the context of skeleton based action recognition, our intuition is that each virtual joint should not be a linear combination of all the original joints but only of subset of them. In fact, in order to obtain such **sparse combination** of the original joints, we can add a constraint of the form $\mu \sum_i (\sum_j |w_{i,j}|)^2 = \mu \|W\|_{1,2}^2$ where $\mu > 0$. This constraint has two effects. It forces a subset of $w_{i,j}$ to be equal to zero for each virtual joint i. The fact that we squared the sum of each line, ensures that the number of non zero elements on each line is roughly the same.

Experiments:

In this experiment, we added the skeleton transformer module we just presented, to our improved baseline reaching 79.2% on the NTU RGB+D dataset [67] in the cross subject setting. In the original paper of Li et al. [45], the authors do not precise how much virtual joints they used. In our case, for a fair comparison, since each skeleton of the NTU RGB+D dataset contain $N = 25$ joints, we use $N' = 25$ virtual joints. In Table 8, we evaluate the strength of the different combinations that we propose.

First, all the proposed combinations (affine, positive and convex) improve over the linear baseline of Li et al. [45]. Second, the sparsity constraint does not seem to give any improvement, even if surprisingly in our early experiments we notice some minor improvements. Third, in all the cases restraining the weights to sum to 1 is improving the accuracy. Finally, with an affine combination and without any sparse constraint, we obtain 79.58% and manage to reach state-of-the-art result.

Setting \ μ sparsity constraint	0	0.001	0.01
Without	79.2	-	-
Linear combination	78.47	77.78	77.08
Affine combination	**79.58**	77.36	77.51
Positive combination	78.69	78.19	77.27
Convex combination	78.95	78.75	78.26

Table 8: Evaluation of the different proposed combinations of our skeleton transformer module on the NTU RGB+D dataset [67] in the cross subject setting. For the Convex and the Positive combinations, we use a Lagrangian constraint $\lambda = 0.01$.

3.3.2 Skeleton motion

We summarize some promising directions of research to better learn 3D action primitives.

- In computer vision, it is well known that instead of learning from images, we can learn action recognition from motion [87] and combining images and motion in a Two Stream Convolutional Network [71] brings improved accuracy. In the context of skeleton-based action recognition, if P_t denotes the pose coordinates at time t, then following [45], we define a skeleton motion as $M_t = P_{t+1} - P_t$. In other words, if the skeleton sequence $\{P_t\}_t$ was of length T, then the skeleton motion sequence $\{M_t\}_t$ is of length $T-1$.

In an early experiment, we found that by training on skeleton motion sequence only, i.e. without skeleton, we can reach as high as 75.57% on the NTU RGB+D dataset [67] in the cross subject setting. Combining motion and skeleton should help improve the performance. In fact, the latest approach and current state-of-the-art 3D pose based action recognition method [45], gets its best result by taking both skeleton and skeleton motion sequences as input, validating our intuition.

- However, we feel that the skeleton motion is currently ill-defined since all the motion of the 3D joints are highly corrolated. Indeed, the information corresponding to the global motion of the actor in the room is contained in every joint motion vector. In machine learning, it is common knowledge that decorrolating the data is important. PCA is largely exploited as a preprocessing tool, and can also be used successfully for data augmentation [41]. Likewise, a potential direction of research could be to decorrolate the skeleton motion, by decomposing the skeleton motion hierarchically w.r.t to the human body kinematic chains.

- Another challenge is that by definition an action only appears in a few frames. In the Appendix Section A, we propose a novel method tailored to train Bidirectional RNN, that aims at better remembering discriminative information over long range period.

4 Toward action recognition for videos "in the wild"

This section is organized as follows. First, in Section 4.1, we describe the state-of-the-art 3D Pose estimation method from [60], and propose an extension to videos using a temporal smoothing mechanism. Then, we design a dataset with YouTube videos recorded "in the wild" in Section 4.2. For each video, we extract a skeleton sequence, preprocess it, and predict action labels by adapting our improved baseline model (introduced in Section 3) that was trained on skeleton sequence recorded in a constrained environment. To the best of our knowledge, we are the first to realize a video-based action recognition pipeline using as intermediate 3D pose sequence extracted from real videos.

4.1 Extracting 3D poses from videos

In this subsection, a human pose (p, P) is defined as the 2D pose p, i.e., the pixel coordinates of each joint in the image; and the 3D pose P, distance of each joint from the body center in the real world (in meters). We consider poses with 13 joints. Our system consists of two key components: 1) a 3D/2D human pose detector that produces multi-modal pose estimations, 2) A temporal consistency stage that selects the most probable modes at each time step.

4.1.1 Localization classification regression network for human pose estimation

Our method builds on top of the state-of-the-art architecture for human 3D pose detection presented in [60]. As shown on Figure 13, inspired by the latest work on object detection [58], they use a CNN architecture to localize classify and regress 3D poses, hence the name of the method: LCR-net. The LCR-net generates, scores and refines a number of pose proposals per image, which allows to predict 2D and 3D pose of multiple people simultaneously. These pose proposals are computed by clustering the pose space into K pose classes, each class $k \in \{1 \dots K\}$ corresponding to a coarse orientated 3D poses centered on the torso. The final pose estimations are obtained by integrating over neighboring pose hypothesis. This results in pairs of full-body 2D/3D pose hypothesis $\{(p_i, P_i)\}$ with associated pose classes k_i and classification scores s_i. One of the advantages of LCR-net is that it can output several modes in cases of ambiguous poses.

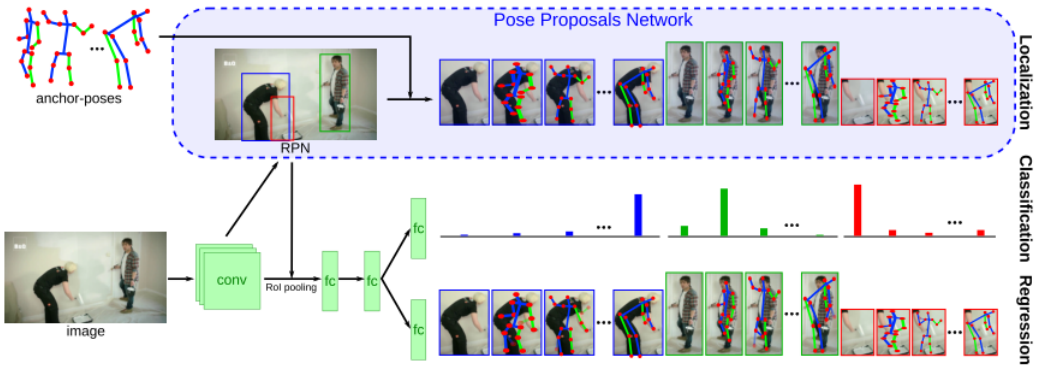


Figure 13: LCR-net architecture. A region proposal network (RPN) enable to extract local features used to classify and regress 3D poses (Figure credit [60]).

4.1.2 Temporal consistency of skeleton sequences

In this section, our goal is to extend LCR-net to videos. In all cases, we exploit LCR-net to extract 2D/3D pose hypothesis $\{(p_i, P_i)\}$ independently on all the frames.

2D tracking. A first tracking algorithm consists in selecting the top-scored 3D pose for each frame. Then, to avoid switching between actors, we prune any 3D pose that does not have a sufficient 2D overlap with the 3D pose $\widehat{P}^{t-1}$ selected in the previous frame. The 2D overlap is computed as the 2D Euclidean distance between 2D poses $\{p_i^t\}_i$ and $\widehat{p}^{t-1}$. However, such tracking might be insufficient in the case of ambiguous poses, e.g. when it is difficult to differentiate between the front and the back of the actor. Indeed, the best 3D pose might abruptly change compared to the 3D pose from the previous frame, despite the fact that their corresponding 2D pose have a non-negligible overlap.

3D temporal smoothing. As used in [31] to disambiguate upperbody hypothesis in a multicamera framework, we employ a Viterbi path finder algorithm to select the optimal mode. In practice, given a sequence with estimated poses, we list all the classes in all modes of all frames and build a dictionary of possible 3D pose states. We then compute the average 3D pose of each state π and build a transition matrix based on Euclidean distance between 3D poses to represent the transition probability $p(\pi_t|\pi_{t-1})$. We assign observation probability $p(O_t|\pi_t)$ using the LCR-net scores. Finally, we find the most probable sequence of states $\pi_0,\ldots,\pi_T$ maximizing:

$$p(\pi_{0:T}|O_{0:T}) \propto \prod_{t=1}^{T} p(\pi_t|\pi_{t-1}) \times \prod_{t=0}^{T} p(O_t|\pi_t)$$

For each frame, we select the 2D/3D poses of the mode corresponding to the estimated state or, if the state is not in the proposed modes, we keep the average 3D pose of the selected state. Note that although we focus on single person case, the method could be applied to multi-persons.

4.2 Video-based action recognition with 3D poses

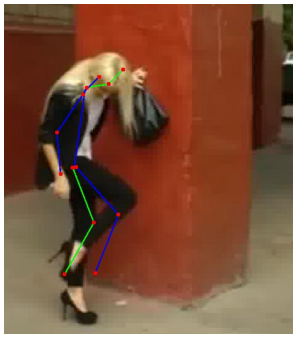
4.2.1 Design of a dataset from YouTube videos

In this final part, we design a small dataset extracted from YouTube videos. This dataset contains 30 videos from 10 actions classes of the NTU RGB+D dataset (pickup, take off a shoe, fall, jump, read, stand up, hopping, sit down, throw, bow) involving a single actor.

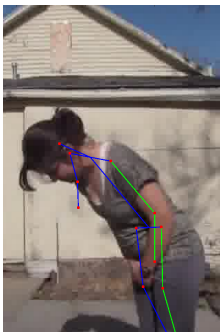
Our goal was not to evaluate our skeleton-based action recognition model on particularly difficult videos. On the contrary, our main selection criterion was to find video clips that are visually close to the actions realized in the NTU RGB+D, i.e. that contain similar 3D motion. Also, to be as close as possible to the NTU RGB+D dataset, we temporally cropped YouTube videos around the action of interest.

Despite the fact that it was trained on static images, LCR-net is able to estimate 3D poses in various complex settings on our YouTube dataset (see Figure 14). Also, in the case of multiple actors, to ensure that our temporal consistency algorithms will track the desired actor, we manually specify the actor of interest in the first frame.

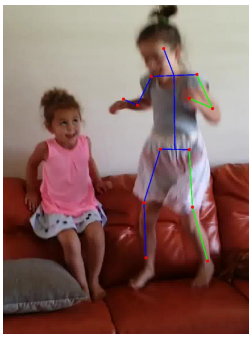
(a) Fall



(b) Take off a shoe



(c) Bow



(d) Jump

Figure 14: 3D poses estimated on our YouTube dataset (shown in 2D for readability). Skeleton sequences (Figure 14a) are extracted from videos "in the wild" that can contain ambiguous poses (Figure 14b), bodies partially visible (Figure14c), multiple actors and motion blur (Figure 14d)

4.2.2 Bridging the gap between constraint environments and the real world

Our aim is to train models on skeleton sequences from well-controlled environments and use them on sequences estimated from real world videos. In order to make possible such domain transfer, we need to construct a common representation space for sequences estimated from videos, and sequences acquired in constraint environments, while keeping good classification performances on the training domain. Below, we summarize the steps that we propose to contruct a common representation between the skeleton sequences estimated from YouTube videos and the sequences from the NTU RGB+D dataset.

- **Dropping joints**. NTU RGB+D skeletons are composed of 25 joints, whereas skeletons estimated from videos with LCR-net only contain 13 joints of the NTU RGB+D skeletons. To bridge this gap, we drop 12 joints of the NTU RGB+D skeletons (fingers, toes, ...).

- **Translation**. In constraint environments, the position of the actor in the scene is known. For videos "in the wild", due to potential camera motion, we can only estimate the position of the actor w.r.t to the camera in each frame independently. Therefore, we lose the information concerning the actor's displacement inside the scene. In order, to eliminate this difference, for NTU RGB+D skeleton sequences, for each frame we translate the skeleton coordinates with respect to the torso joint of the current frame.

- **Rotation**. The view angles between YouTube videos and constraint environments are often different. We solve this problem by rotating in both datasets the entire skeleton sequences w.r.t to the first frames. First, since actors perform their actions into different directions, we define as x-axis the vector from the right shoulder to the left shoulder. Second, since the cameras view angle is different between datasets, we constrain the y-axis to lie in the plane defined by the two shoulders and the middle of the hips.
- **Scaling**. Finally, between NTU RGB+D and our YouTube dataset, the actors' sizes are not the same. An additional challenge was that the scale of the 3D pose estimated by LCR-net can vary along a sequence. Our solution was to compute the length of body bones on the training set of the NTU RGB+D dataset. Then, based on these statistics, we compute for each skeleton sequences of our YouTube dataset a single rescaling parameter s, and scale each skeleton with the diagonal matrix $s \times I$.

All the steps presented above, were necessary to at least recognize the action label in a few YouTube videos. From now on, **pose alignement** will refer to the three following steps: translation, rotation and scaling. As illustrated on Table 9, pose alignment and using 13 joints instead of 25, reduces by 5% the performance of our improved baseline on NTU RGB+D.

Pose alignment / # of joints	With	Without
13	74.43	75.94
25	76.28	**79.2**

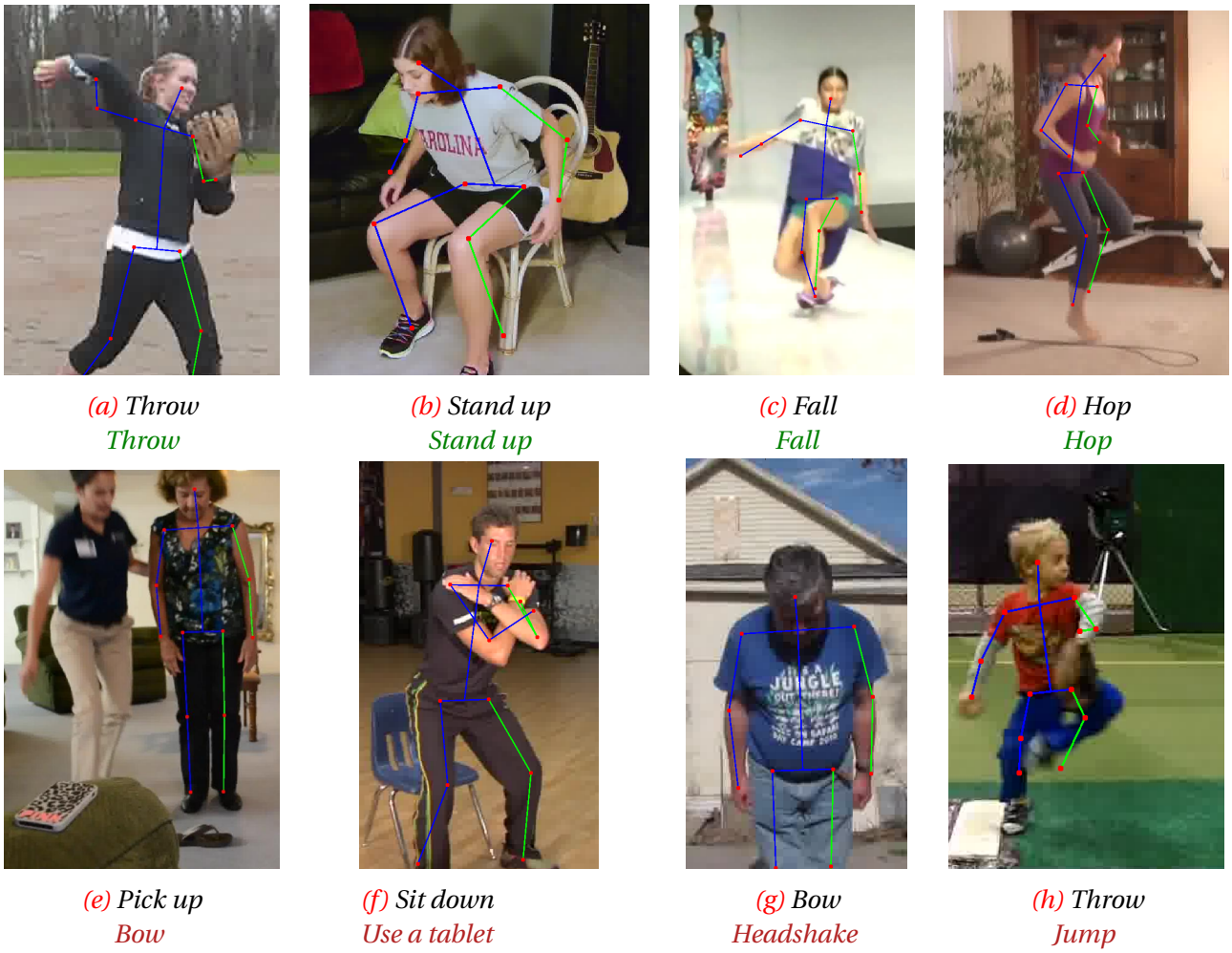
Table 9: Influence of the pose alignment and the reduction of the number of joints on the improved baseline evaluated on NTU RGB+D dataset in the cross subject setting. As expected, they both individually provide worse performance. Also, when combined they further degrade our model's accuracy. However, 74.43 *% is still a reasonable accuracy.*

4.2.3 Action recognition with skeleton sequences estimated from videos

In order to bridge the gap between skeleton sequences estimated from YouTube videos and sequences from the NTU RGB+D dataset, we use a model scoring 74.43% on the NTU RGB+D dataset, as described in Table 9. As shown in Table 10, this model can predict successfully the action label of some videos of our YouTube dataset. Our modest result at least show that domain adaptation (between constraint environments skeleton sequences and sequences estimated from videos "in the wild") is possible. Also, temporal smoothing does not bring much improvement, potentially due to the fact that our videos does not contain ambiguous poses.

Classifier	Pose estimation method	Accuracy (%)
Random	-	1.7
Improved baseline	LCR-Net + 2D tracking	10
Improved baseline	LCR-Net + 3D temporal smoothing	**13.3**

Table 10: Quantitative results on our YouTube dataset.



(a) Throw
Throw

(b) Stand up
Stand up

(c) Fall
Fall

(d) Hop
Hop

(e) Pick up
Bow

(f) Sit down
Use a tablet

(g) Bow
Headshake

(h) Throw
Jump

Figure 15: Selected results on our YouTube dataset. Ground truth labels are written in black. Good predictions are on the top line, and erroneous predictions are on the bottom line.

Finally, in Figure 15, we select some qualitative result of our model. Some erroneous predictions are due to closeness between actions classes (Figure 15g 15e) or to unsual motion of the arms (Figure 15f) or of the legs (Figure 15h). Such mistakes could likely be avoided by exploiting the scene context (e.g the object on the floor in Figure 15e, or the chair in Figure 15f) to better discriminate between actions.

Also, NTU RGB+D dataset's actions are extremely specific, and despite the high number of videos, for many actions their dataset only represent a single possible realization of each action label. Thus, even if our improved model has a good score on NTU RGB+D dataset, it potentially might just means that we overfit on the definition that the authors gave to each action label. Unfortunately, we did not manage to find YouTube videos that were fully matching the NTU RGB+D skeleton sequences, and this could partially explain our modest prediction results. Potentially, a solution would be to create a dataset with 3D poses extracted from videos to better represent the diversity of realization of each action class. We leave the design of such novel dataset, and the improvement of this early attempt as future work.

5 Conclusion

5.1 Discussions

In this master thesis, we made a thorough review of topics related to action recognition based on 3D pose sequence. Then, with the Tensorflow library, we implemented ourselves a complete deep learning pipeline, and experimented with recurrent neural networks on the largest dataset for skeleton sequence based action recognition called NTU RGB+D [67]. Thanks to a proposed neural network module, we managed to reach the main objective of the internship, which was to obtain state-of-the-art result for the task of action recognition from skeleton sequences. Afterwards, we design a dataset with YouTube videos recorded "in the wild" and extract 3D pose sequences with a state-of-the-art 3D pose estimation method [60] improved with a temporal consistency mechanism. To the best of our knowledge, we are the first to realize an action recognition pipeline for real videos using 3D pose sequence as an intermediate representation. We hope our primarily results will motivate further research on this topic.

As a personal long term direction of research, I am convinced that 3D human representation will play an important role in a near future with applications such as autonomous cars, robotics, human-computer interaction. Recently, in various domains such as object recognition, video games, and board games, deep neural networks have achieved performance that equals or even beats humans in some respects. According to Lake et al. [42], in order to reach a next step in terms of intelligence, machine needs to understand intuitively the forces of the world. In the case of human motion, biomechanics and retro-engineering might become important tools to let machine learn intuitive theories of physics. A second ingredient toward Artificial Intelligence is that "machine should build causal models of the world, rather than merely solving pattern recognition problem". Using causality to reason about 3D scenes is a great challenge especially for robotics. For example, it is extremely useful if a robot can detect people falling, but it is even better if it can anticipate the fall and prevent it.

5.2 Future work

In a potential PhD thesis, we will build on this recent effort and leverage 3D pose information extracted from real-world videos to model, recognize and predict human actions in videos using recurrent neural networks architectures. Several challenges arise from this objective.

Improving action recognition from skeleton sequences

- The level of details, i.e. number of body joints, used to represent the human body will have to be carefully studied. What is the optimum 3D representation required to model any action ? Certain activities might require, not only body pose information, but also detailed hand pose configuration, for instance when the action involves the manipulation of one or several objects. Furthermore, acquiring some knowledge in biomechanics might be needed to better design features that take into account the specific characteristic of human body motion.

- The data sets that will be employed to train (and test) the architectures will have to be carefully selected. While low level image features might be sufficient to capture context and recognize activities such as “playing basketball”, 3D poses could prove to be useful to characterize finer-grained actions such as “dribbling”, “shooting”, “blocking” or “passing”. Existing datasets might not cover such detailed annotations. If required, a novel dataset might need to be collected and annotated.

Toward action recognition from videos "in the wild"

- The 3D poses captured “in the wild” are fundamentally different from the data used to train and test existing 3D human action recognition systems. These methods consider that the entire body is fully available and that training and test sequences have been captured in the same controlled environments. These hypotheses do not hold when working with real-world sequences which implies dealing with unaligned 3D poses and inaccurate, possibly incomplete, measurements in cases of occlusions or truncations by image boundaries. While experimenting with sequences extracted either from real world or controlled environments, we might need to design a shared skeleton representation both time-invariant (speed of execution, video frame rate) and subject-invariant (length of each limb, flexibility of the body, left/right handed).

- Finally, the combination of additional image features capturing object appearance and scene context with the 3D pose features will be explored to disambiguate activities with similar pose sequences. For example, the actions brushing teeth, drinking water, smoking a cigarette or playing harmonica all involve a movement of the hand toward the mouth.

Acknowledgement: I am gratefully indebted to Gregory Rogez and Cordelia Schmid for their priceless directions, discussions and encouragements.

Appendix

A Boosting bidirectional recurrent neural network with an inattention mechanism

In this part, we explore another direction to improve skeleton-based action recognition. In particular, we propose a method tailored to train Bidirectional RNN for the task of sequence classification by better remembering discriminative information over long range period.

A.1 Introduction

It is a common held view that recurrent neural networks (RNN) are capable of learning to remember long range interaction. Our contribution is to go one step further by ensuring that at each time step, the RNN has learned a representation that enables it to predict the correct label. As teachers efficiently corrects students by pinpointing questions where they were inattentive, we should train recurrent neural network by explicitly improving its memorization capacity where it is lacking.

In this work, we focus on Bidirectional RNN, and notice that in the context of sequence classification, we can see them as an ensemble model. Indeed, at each point in time, the network gets some signal from the forward RNN and the backward RNN. Each time step has a full view on the sequence and makes its own prediction. And in the end, the final prediction is the averaging of the prediction over time.

Our intuition is that by focusing on time steps where the network is struggling, we want to enforce that the network predicts accurately at each point in time, meaning that the network managed to remember important patterns for short, long and very long temporal range.

A.2 Related work

Our approach is mostly inspired by semantic segmentation literature where several methods [88, 61] rely on **online bootstrapping** to train CNN (pick hard training images from the whole training set according to the current losses). In [88] the authors propose, for the task of semantic segmentation, to focus on hard (so more valuable) pixels during training. This adds 2 points to their accuracy, and they explain that the proposed bootstrapping method can improve the performance for categories that are less frequent in the training data.

Recently, in order to tackle imbalanced training data distribution in semantic segmentation dataset (where classifiers tend to be biased towards the majority classes during inference), Rotabui et al. [61] constructed a class-agnostic adaptive weighting scheme that bias the learner toward under performing parts of the image. The intuition is that among a family of possible weighting of the pixels, they choose the one that brings the maximal loss, and they optimize this maximal loss.

However, the comparison between semantic segmentation and sequence classification has its limits. Indeed, in the case of sequence classification, the labeling is the same for all the time steps, whereas with semantic segmentation, image pixels usually have different labeling.

Our work is also influenced by recent advances in machine learning. At the first glance of a scene, human perception usually focuses on salient parts of its surrounding. In machine learning, such process is called **attention mechanism**, and leads to improved accuracy, as the system can focus on parts of the data, which are most relevant to the task.

Attention mechanisms can be either categorized as soft, differentiable attention or hard, non-differentiable attention. In his seminal work, Mnih et al. [53] propose a visual hard-attention for image classification. Such approach takes hard decision (i.e. selects a fixed number of top regions in terms of saliency) which cannot be easily learned through continuous based optimization. On the other hand, soft attention assigns a weight to each part of the observation dynamically. When the objective function is differentiable with respect to these attention weights, it makes it possible to learn by backpropagation the optimal attention weights. Soft attention was used for various applications such as neural machine translation [4] , or memory augmented neural network [25].

In the context of 3D pose based action recognition, Song et al. [73] exploit a soft-attention mechanism temporally to automatically select key-frames. On the contrary to such attention mechanism that aims to select salient areas, in this work we wish in to select time steps where the network is under performing, hence the name "inattention weights".

A.3 An online upper bound loss

The loss function, of a Bidirectional RNN for the classification of a sequence x of length T with class label y, commonly decomposes into a sum of time step-specific losses as follows:

$$L(y,x) = \frac{1}{T}\sum_{t=1}^{T} -\log p_t^y \text{ where } p_t = f_\theta(x) \tag{11}$$

where p_t^c is the probability predicted by the Bidirectional RNN f parametrized by θ for the class c at time step t. A space of weighting functions $\mathscr{W}$ designates any space satisfying the following property $\forall w \in \mathscr{W}, L_w(y,x) \geq L(y,x)$. In other words, any L_w is an upper bound of the original loss L.

In our case, we build the space of weighting functions $\mathscr{W}$ by sorting online at each forward pass $\{p_t^y\}_t$ the set of predicted probabilities w.r.t to the true label y. Let S be a set of time-indices corresponding to the lowest prediction probabilities. $\mathscr{W}$ assigns weights only to time step in S,

$$\mathscr{W} = \{w_t \geq \frac{1}{T} \text{ if } t \in S, w_t = 0 \text{ if } t \notin S \mid \sum_t w_t = 1\} \tag{12}$$

$\mathscr{W}$ is indeed a weighting space function due to the fact that for $w \in \mathscr{W}$:

$$-\sum_{t\in S}(w_t - \frac{1}{T})\log p_t^y \geq -(1-\frac{|S|}{T})\max_{t\in S}\log p_t^y \geq -\sum_{t\notin S}\frac{1}{T}\log p_t^y \text{ which brings } L_w(y,x) \geq L(y,x)$$

The goal of this design choice for $\mathscr{W}$ is to shift the focus on time steps where the loss is higher, while retaining a theoretical link to the original loss. In other words, at each training steps, we compute weights that drive the loss toward time-steps where the network is under-performing. Each weighting function tends to highlight where the network has been negligent, hence the name inattention weights for w and the name inattention loss for L_w.

During training, for each batch example, we select from that weighting functions space some inattention weights w. We notice that the original loss can be retrieve by choosing $|S| = T$. On the other hand, $|S| = 1$ tantamounts to putting all the inattention weights solely on the time step where the loss is the highest.

A.4 Experiments

First, we perform an ablation study of Bidirectional recurrent neural network (BRNN) for the task of skeleton based action recognition (see Table 11). Similarly to the baseline experimental setting (described in Section 3), our architecture contains 3 layers of BRNN. Also, we capitalize on dropout, early stopping and L^2 regularization to reduce overfitting. The only difference is that, instead of simply using the final prediction of the network, we exploit a training loss that takes into the account the prediction of the BRNN at each time step (see Equation 11).

RNN Units per Direction / RNN type	50	100	200	400
Standard RNN	53.77	62.15	66.06	66.54
LSTM	75.45	77.42	77.66	77.35
GRU	75.35	77.9	**79.03**	78.1

Table 11: Ablation study of Bidirectional recurrent neural networks for 3D pose based action recognition, evaluated on NTU RGB+D dataset [67] in the cross subject evaluation setting. Once again, on the overall, GRU beats LSTM which in turns outperforms Standard RNN.

Second, we experiment our contribution by varying the size of the set S (see Table 12). As baseline, in this experiment, the inattention weights are equally distributed between the time steps in S. In other words, such time step are given a weight equal to $\frac{1}{|S|}$ while other steps have their inattention weight set to zero.

|S|	10	20	30	40	50	60	70	T
GRU 200 Units	78.36	**79.09**	78.6	78.6	79.07	78.86	78.88	79.03

Table 12: In the context of our novel inattention loss, we vary the size of the set of selected time steps S. We exploit our best BRNN model (i.e., 200 GRU units per Direction) and evaluate on NTU RGB+D dataset [67] in the cross subject evaluation setting.

On the overall, our contribution is not improving much over the baseline $|S| = T$. And, even when it does, the gain in accuracy is not significant enough to conclude in favor of our inattention loss. However, in terms of future work we believe that sampling randomly the inattention weight could potentially act as a regularizer.

In this early work, the way we spread out the inattention weights corresponds to a hard attention mechanism. Another possible research direction could be to use soft attention, and let the network learn itself the set of inattention weights that leads to a suitable loss.

For instance, we could create another neural network called the inattention network that would predict for each batch of data the inattention weights. Inspired by recent successful generative neural networks [22], we can imagine this inattention network as a teacher that would generate a suitable set of weights, that could potentially lead the main network (we can picture it as the student) to better learn from data sequences.

References

[1] Carnegie Mellon University, Motion Capture Database http://mocap.cs.cmu.edu/.

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, Xiaoqiang Zheng, and Google Research. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv*, 2016.

[3] Ijaz Akhter and Michael J Black. Pose-conditioned joint angle limits for 3D human pose reconstruction. *CVPR*, 2014.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *ICLR*, 2015.

[5] Fabien Baradel, Christian Wolf, and Julien Mille. Pose-conditioned Spatio-Temporal Attention for Human Action Recognition. *arXiv*, 2017.

[6] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 1994.

[7] Hakan Bilen, Basura Fernando, Efstratios Gavves, Andrea Vedaldi, and Stephen Gould. Dynamic Image Networks for Action Recognition. *CVPR*, 2016.

[8] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J Black. Keep it SMPL : Automatic Estimation of 3D Human Pose and Shape from a Single Image. *ECCV*, 2016.

[9] Michael M Bronstein, Joan Bruna, Yann Lecun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond Euclidean data. *arXiv*, 2016.

[10] James Charles, Tomas Pfister, Derek Magee, David Hogg, and Andrew Zisserman. Personalizing Human Video Pose Estimation. *CVPR*, 2016.

[11] Xianjie Chen and Alan Yuille. Articulated Pose Estimation by a Graphical Model with Image Dependent Pairwise Relations. *NIPS*, 2014.

[12] 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. *EMNLP*, 2014.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. *CVPR*, 2009.

[14] Maxime Devanne. 3D Human Behavior Understanding by Shape Analysis of Human Motion and Pose. *Ph.D. dissertation*, 2015.

[15] Yong Du, Yun Fu, and Liang Wang. Skeleton Based Action Recognition with Convolutional Neural Network. *ACPR*, 2015.

[16] Yong Du, Wei Wang, and Liang Wang. Hierarchical recurrent neural network for skeleton based action recognition. *CVPR*, 2015.

[17] Xiaochuan Fan, Kang Zheng, Youjie Zhou, and Song Wang. Pose locality constrained representation for 3d human pose reconstruction. *ECCV*, 2014.

[18] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *ICML*, 2016.

[19] Ross Girshick, Jeff Donahue, Trevor Darrell, U C Berkeley, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR*, 2014.

[20] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. *AISTATS*, 2010.

[21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. *MIT Press*, 2016.

[22] Ij Goodfellow, J Pouget-Abadie, and Mehdi Mirza. Generative Adversarial Networks. *NIPS*, 2014.

[23] Alex Graves, Santiago Fernandez, and Schmidhuber Jürgen. Multi-dimensional recurrent neural networks. *ICANN*, 2007.

[24] Alex Graves, Abdelrahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. *ICASSP*, 2013.

[25] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural Turing Machines. *CoRR*, 2014.

[26] F. Han, B. Reily, W. Hoff, and H. Zhang. Space-Time Representation of People Based on 3D Skeletal Data: A Review. *Arxiv*, 2016.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *CVPR*, 2016.

[28] Samitha Herath, Mehrtash Harandi, and Fatih Porikli. Going Deeper into Action Recognition: A Survey. *arXiv*, 2017.

[29] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. *A Field Guide to Dynamical Recurrent Neural Networks.*, 2001.

[30] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 1997.

[31] Michael Hofmann and uvanl M Dariu Gavrila. Multi-view 3D Human Pose Estimation combining Single-frame Recovery, Temporal Integration and Model Adaptation. *CVPR*, 2009.

[32] Zhiwu Huang, Chengde Wan, Thomas Probst, and Luc Van Gool. Deep Learning on Lie Groups for Skeleton-based Action Recognition. *CVPR*, 2017.

[33] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv*, 2015.

[34] Gunnar Johansson. Visual perception of biological motion and a model for its analysis. *Perception & Psychophysics*, 1973.

[35] Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. Grid Long Short Term Memory. *ICLR*, 2016.

[36] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale Video Classification with Convolutional Neural Networks. *CVPR*, 2014.

[37] Qiuhong Ke, Mohammed Bennamoun, Senjian An, Ferdous Sohel, and Farid Boussaid. A New Representation of Skeleton Sequences for 3D Action Recognition. *arXiv*, 2017.

[38] Tae Soo Kim and Austin Reiter. Interpretable 3D Human Action Analysis with Temporal Convolutional Networks. *CVPR, BNMW Workshop*, 2017.

[39] Diederik P Kingma and Jimmy Lei Ba. Adam: A Method for Stochastic Optimization. *ICLR*, 2015.

[40] Adam G Kirk, James F O 'brien, and David A Forsyth. Skeletal Parameter Estimation from Optical Motion Capture Data. *CVPR*, 2005.

[41] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *NIPS*, 2012.

[42] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building Machines That Learn and Think Like People. *arXiv*, 2016.

[43] Colin Lea, Michael D Flynn, René Vidal, Austin Reiter, and Gregory D Hager. Temporal Convolutional Networks for Action Segmentation and Detection. *CVPR*, 2017.

[44] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *IEEE*, 1998.

[45] Chao Li, Qiaoyong Zhong, Di Xie, and Shiliang Pu. Skeleton-Based Action Recognition with Convolutional Neural Networks. *ICME Workshop*, 2017.

[46] Sijin Li and Antoni B Chan. 3D human pose estimation from monocular images with deep convolutional neural network. *ACCV*, 2014.

[47] Sijin Li, Weichen Zhang, and Antoni B. Chan. Maximum-Margin Structured Learning With Deep Networks for 3D Human Pose Estimation. *ICCV*, 2015.

[48] Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang. Spatio-Temporal LSTM with Trust Gates for 3D Human Action Recognition. *ECCV*, 2016.

[49] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. *CVPR*, 2015.

[50] David G Lowe. Distinctive image features from scale invariant keypoints. *IJCV*, 2004.

[51] Julieta Martinez, Michael J Black, and Javier Romero. On human motion prediction using recurrent neural networks. *CVPR*, 2017.

[52] Warren S Mcculloch and Walter Pitts. A Logical Calculus Of The Ideas Immanent In Nervous Activity. *Bulletin of Mathematical Biophysics*, 1943.

[53] Volodymyr Mnih, Nicolas Heess, Alex Graves, and And Others. Recurrent models of visual attention. *NIPS*, 2014.

[54] Thomas B. Moeslund, Adrian Hilton, and Volker Kruger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 2006.

[55] Christopher Olah. Understanding LSTM Networks http://colah.github.io/posts/2015-08-Understanding-LSTMs, 2015.

[56] Florent Perronnin, Yan Liu, Jorge Sánchez, and Hervé Poirier. Large-Scale Image Retrieval with Compressed Fisher Vectors. *CVPR*, 2010.

[57] Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Reconstructing 3D human pose from 2D image landmarks. *ECCV*, 2012.

[58] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *NIPS*, 2015.

[59] Grégory Rogez and Cordelia Schmid. MoCap-guided Data Augmentation for 3D Pose Estimation in the Wild. *NIPS*, 2016.

[60] Grégory Rogez, Philippe Weinzaepfel, and Cordelia Schmid. LCR-Net : Localization-Classification-Regression for Human Pose LCR-Net : Localization-Classification-Regression for Human Pose. *CVPR*, 2017.

[61] Samuel Rotabuì, Gerhard Neuhold, and Peter Kontschieder. Loss Max-Pooling for Semantic Image Segmentation. *arXiv*, 2017.

[62] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, 2016.

[63] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 1988.

[64] Nikolaos Sarafianos, Bogdan Boteanu, Bogdan Ionescu, and Ioannis A. Kakadiaris. 3D Human pose estimation: A review of the literature and analysis of covariates. *Computer Vision and Image Understanding*, 2016.

[65] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 1997.

[66] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *arXiv*, 2013.

[67] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis. *CVPR*, 2016.

[68] Hava T Siegelmann and Eduardo D Sontag. On the computational power of neural nets. *Journal of computer and system sciences*, 1995.

[69] Edgar Simo-Serra, Ariadna Quattoni, Carme Torras, and Francesc Moreno-Noguer. A Joint Model for 2D and 3D Pose Estimation from a Single Image. *CVPR*, 2013.

[70] Edgar Simo-Serra, Arnau Ramisa, Guillem Aleny, Carme Torras, and Francesc Moreno-Noguer. Single image 3D human pose estimation from noisy observations. *CVPR*, 2012.

[71] Karen Simonyan and Andrew Zisserman. Two-Stream Convolutional Networks for Action Recognition in Videos. *NIPS*, 2014.

[72] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ICLR*, 2015.

[73] Sijie Song, Cuiling Lan, Junliang Xing, Wenjun Zeng, and Jiaying Liu. An End-to-End Spatio-Temporal Attention Model for Human Action Recognition from Skeleton Data. *AAAI*, 2017.

[74] 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*, 2014.

[75] Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. *ICML*, 2011.

[76] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CVPR*, 2015.

[77] Bugra Tekin, Artem Rozantsev, Vincent Lepetit, and Pascal Fua. Direct prediction of 3D body poses from motion compensated sequences. *CVPR*, 2016.

[78] Jonathan Tompson, Arjun Jain, Yann Lecun, and Christoph Bregler. Joint Training of a Convolutional Network and a Graphical Model for Human Pose Estimation. *NIPS*, 2014.

[79] Alexander Toshev and Christia Szegedy. Deeppose: Human pose estimation via deep neural networks. *CVPR*, 2014.

[80] Ul Varol, Ivan Laptev, and Cordelia Schmid. Long-term Temporal Convolutions for Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[81] Vivek Veeriah, Naifan Zhuang, and Guo-Jun Qi. Differential Recurrent Neural Networks for Action Recognition. *ICCV*, 2015.

[82] Raviteja Vemulapalli, Felipe Arrate, and Rama Chellappa. Human action recognition by representing 3D skeletons as points in a lie group. *CVPR*, 2014.

[83] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *CVPR*, 2015.

[84] Chunyu Wang, Yizhou Wang, Zhouchen Lin, Alan L Yuille, and Wen Gao. Robust estimation of 3D human poses from a single image. *CVPR*, 2014.

[85] Pichao Wang, Zhaoyang Li, Yonghong Hou, Wanqing Li, and Advanced Multimedia. Action Recognition Based on Joint Trajectory Maps Using. *ACMMM*, 2016.

[86] Wenzheng Chen Wang, Huan Wang, Yangyan Li, Hao Su, Zhenhua, Chen, Changhe Tu, Dani Lischinski, Daniel Cohen-Or, and Baoquan. Synthesizing Training Images for Boosting Human 3D Pose Estimation. *3DV*, 2016.

[87] Philippe Weinzaepfel. Motion in action: optical flow estimation and action localization in videos. *Ph.D. dissertation*, 2016.

[88] Zifeng Wu, Chunhua Shen, and Anton van den Hengel. High-performance Semantic Segmentation Using Very Deep Fully Convolutional Networks. *arXiv*, 2016.

[89] Hashim Yasin, Umar Iqbal, Bjorn Kruger, Andreas Weber, and Juergen Gall. A dual-source approach for 3D pose estimation from a single image. *CVPR*, 2016.

[90] Matthew D Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. *ECCV*, 2014.

[91] Pengfei Zhang, Lan Cuiling, and Xing Junliang. View Adaptative RNN for High Performance Human Action Recognition from Skeleton Data. *arXiv*, 2017.

[92] Songyang Zhang, Xiaoming Liu, and Jun Xiao. On Geometric Features for Skeleton-Based Action Recognition using Multilayer LSTM Networks. *WACV*, 2017.

[93] Feng Zhou and Fernando De la Torre. Spatio-temporal matching for human detection in video. *ECCV*, 2014.

[94] Xiaowei Zhou, Spyridon Leonardos, Xiaoyan Hu, and Kostas Daniilidis. 3D shape estimation from 2D landmarks: A convex relaxation approach. *CVPR*, 2015.

[95] Xiaowei Zhou, Menglong Zhu, Spyridon Leonardos, Konstantinos G Derpanis, and Kostas Daniilidis. Sparseness meets deepness: 3D human pose estimation from monocular video. *CVPR*, 2016.

[96] Wentao Zhu, Cuiling Lan, Junliang Xing, and Xiaohui Xie. Co-occurrence Feature Learning for Skeleton based Action Recognition using Regularized Deep LSTM Networks. *AAAI*, 2016.