ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CVL Computer Vision Lab

Studying Multiple Cues for Human Action Recognition on RGB+D Data

Semester Project for Master in Robotics, Systems and Control

Jonathan Gan

Advisor: Dr. Zhiwu Huang, Dr. Danda Pani Paudel
Supervisor: Prof. Luc Van Gool

September 15, 2017

Abstract

The growing use of commodity RGB-D sensors has given rise to the emergence of multi-modal data. For example, the Microsoft Kinect, a RGB-D sensor, provides RGB videos, depth sequences and skeleton information. However, most existing human action recognition techniques focus on various single input modalities of these data. We hence study the effectiveness of leveraging multi-modal information to improve the accuracy of action recognition. In particular, we introduce a two-stream network combining the Parts-aware LSTM on 3D joint positions and a novel LieLSTM method using Lie group representation of 3D skeletal data. The evaluation on a large-scale publicly available RGB-D dataset, the NTU-RGB+D, verify that our proposed two-stream network can outperform those using single modalities, while achieving comparable performance with state-of-the-art methods. Simultaneously, we investigate the use of action recognition methods on different modalities, such as RGB and 2D skeletal data – the latter having not been used for human action recognition till date, to the best of our knowledge – and present an optimistic case for further study into these areas.

Contents

Chapter 1

Introduction

1.1 Related Work

As commodity cameras become cheaper to be integrated into every context of our daily lives, the drive for greater scene understanding and intelligent vision algorithms accelerates. One of the important fields of vision research is the ability to understand and recognize different human actions, and has far-reaching implications to surveillance, intelligent video retrieval, and human-computer interactions.

Traditionally, human action recognition methods use handcrafted action representation on videos, using "expert-designed" feature detectors and descriptors like SIFT, Histogram of Oriented Gradients (HoGs) or Local Binary Patterns (LBP), just to name a few. Generic classifiers, such as the Support Vector Machine (SVM), are then trained to perform classification. Popular action recognition methods of such kinds include the bag-of-features video encoding model [1] and improved dense trajectories model that couples feature descriptors and dense optical flow [2]. These methods also exhibited state-of-the-art performance amongst handcrafted representation-based approaches, in many publicly available datasets.

Interest in using 3D information in videos also arose quickly with the availability of 3D cameras, such as the Microsoft Kinect, paving a new direction for action recognition. Earlier methods built on 3D feature descriptors, like Harris3D [3] and 3D HoGs [4], performed classification similar to 2D methods. Following the seminal work by Shotton et al. [5] for estimating joint locations of a human body from a depth map, many methods were developed to classify actions using skeletal information ([6], [7], [8]).

A clear shift in interest towards learning approaches, specifically "deep learning", was also evident in the human action recognition community, as "deep learning" increasingly proved its worth in recent years from image classification to automatic speech recognition. Simonyan et al. showed initial success using a two-stream convolutional neural network on RGB and optical flow images [9]. Du et al.'s approach, on the other hand, used recurrent neural networks (RNNs) for skeleton-based action recognition [10].

However, as attractive as deep learning algorithms appear in action recognition, an immense amount of data is inevitably required. Unlike image classification, many datasets for human action recognition contain considerably fewer samples – never reaching the 10,000 mark before 2016 [11].

Shahroudy et al. identified this issue and released a new human action recognition dataset unrivalled in scale, coined the *NTU-RGB+D*, and proposed a *Part-Aware LSTM network* on their

dataset [11]. Subsequently, several neural network approaches ([12], [13], [14], [15]) were tested on the NTU-RGB+D dataset, each showing purported improvement over previous methods.

1.2 Motivation

The rate at which new methods are introduced is staggering, considering that the NTU-RGB+D was only released less than one and a half years ago as of this writing. Unfortunately, much of the source code from methods tested on this dataset remained closed-source, leaving one to question the integrity and reproducibility of results in this research area.

Furthermore, many of the introduced methods use 3D skeletal data for the training of neural networks, largely attributed to its availability in the largest human action recognition dataset. The physical limitation of active RGB-D cameras, however, confines such applications for only indoor use and for considerably short range. This effectively hinders the usage of developed methods for use in the field of security surveilance.

Lastly, a couple of works have demonstrated the effectiveness of using multiple modalities to improve performance on publicly available datasets ([9], [14], [16]). Simultaneously, the NTU-RGB+D also serves as a good dataset given its multiple modalities. Both factors motivate this investigation into the effects of multiple modalities for human action recognition using deep learning methods.

1.3 Contributions

The goal of this work is threefold:

1. To introduce a collection of algorithms that have been proposed on the NTU-RGB+D dataset as a unified open-source framework, allowing for a critical examination of previously proposed algorithms,

2. To investigate the efficacy of using only 2D information, both appearance and skeletal information based, for human action recognition, and

3. To study a novel approach of using Lie group information as inputs to a LSTM architecture, which we dub the LieLSTM. We additionally investigate the performance of a two-stream network, in which the predictions of the LieLSTM and the Parts-aware LSTM are combined.

Chapter 2

Methodology

In this chapter, we introduce the NTU-RGB+D dataset and elaborate on the different algorithms tested on the dataset.

2.1 The *NTU-RGB+D* Dataset

The NTU-RGB+D [11] is by far the largest dataset for human action recognition in terms of size, with 56880 action samples, and number of modailities – RGB-D, infrared images, and 3D skeletal joint information – incorporating 60 action classes from 40 subjects and 80 views. The dataset was captured with a Microsoft Kinect v2 sensor [17] in an indoor, controlled environment with no more than two subjects in the video. The variability of non-human objects in the dataset is minimal due to its office-like setting. A detailed list of the 60 action classes is provided by the authors in [18].

A notable characteristic of the NTU-RGB+D is the deterministic and intentional variation in views for every action recorded, varying factors such as height of cameras, distance of cameras from subjects, and camera view angles. Each variation is uniquely identified by an ID, allowing for a more detailed analysis of the performance of algorithms with respect to the views. A sample of the NTU-RGB+D dataset is provided in Fig. 2.1.

Figure 2.1: Some snippets from the NTU-RGB+D dataset showing variation in the views across samples. The last row additionally illustrates the RGB, RGB+joints, depth, depth+joints and IR modalities of a sample frame. Source: [11]

2.2 The Palette of Deep Learning Algorithms

The algorithms utilized for assessing the dataset can be mainly categorized by their input modalities – skeletal information and appearance-based information (RGB). For the usage of skeletal information, the Long Short-Term Memory (LSTM), a type of RNN architecture, is used as the basis for the various algorithms developed for the purpose of human action recognition.

Apart from theoretical descriptions of the following algorithms, details of which can be found in the corresponding papers, we also elaborate on the implementation of these algorithms for easier reproducibility of the methods.

The Kinect v2 sensor used for the creation of the NTU-RGB+D dataset additionally is able to track the joint locations of two human subjects in space over time. The practical range of skeletal tracking is between 0.8 and 2.5 meters and its efficacy is greatest when the human subject is facing the sensor [19]. The positions of up to 25 joints can be tracked by the sensor as shown in Fig. 2.2.

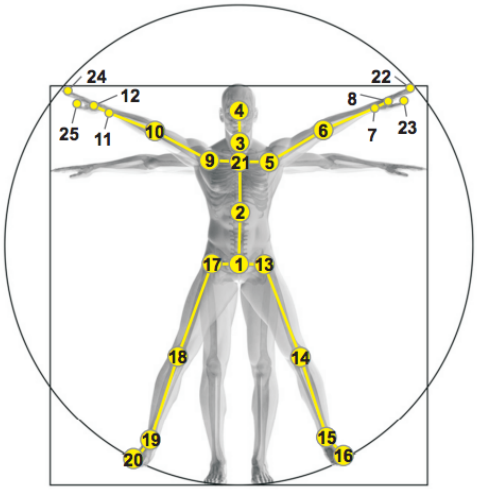


Figure 2.2: Body parts as tracked by the Kinect v2 sensor. Source: [11]

2.2.1 Vanilla Long Short Term Memory (LSTM)

To understand a LSTM network, one has to first understand the RNN. Traditional artificial neural networks (ANN), while effective in solving a large array of problems, do not take past information into account when trying to understand sequential content like a snippet of spoken text or a video. RNNs address this problem by including loops inside a neural network, allowing networks to persist. This loop in the neural network is usually expressed programmatically by "unrolling" the loop, resulting in multiple instances of the neural network as seen in Fig. 2.3.

RNNs hence present themselves as a effective network choice for human action recognition – which mainly involve a sequence of images or human poses – as noted in previous works listed in Section 1.1.

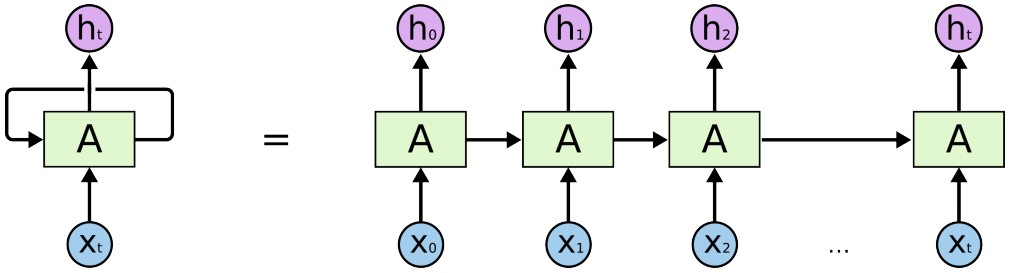


Figure 2.3: A RNN unrolled over time. Source: [20]

While appealing in theory, the practical ability for RNNs to learn long-term dependencies is usually limited as shown by Bengio et al. [21] and Hochreiter [22]. LSTM were hence introduced to solve this problem, being explicitly designed to "remember" information for long periods of time. The core idea of LSTM lies in the ability to decide what information should be added or removed from the "cell state" through a series of "gates". In contrast to the RNN which only contains a hidden state that is passed through a tanh function, the cell state of the LSTM is only optionally exposed to linear interactions and can possibly even pass through an instance of the network unchanged. We spare the reader of the mathematical formulations of the LSTM and recommend Christopher Olah's intuitive explanation on the topic [20] or the vast amount of resources available for an elaborate theoretical understanding of the LSTM.

Programmatically, the RNN (and its LSTM variant) are well established in the Tensorflow library [23] and can be easily created with the help of many independent online guides on the topic such as [24]. We particularly found Sherjil Ozair's `char-rnn-tensorflow` project [25] as a good start for implementing and training the first LSTMs. The implementation of LSTMs in Tensorflow requires that an upper bound on sequence length is given and the dataset should be padded with zeros if it is shorter than the maximum length specified. Additionally, extra care has to be taken in ensuring that the true sequence length of each piece of data is known so that the last relevant output is passed to the softmax layer for classification purposes. Danijar Hafner provides a good explanation of how this should be done [26]. For the input representation of the LSTM, we simply concatenate all 25 joints position into a $\mathbb{R}^{75}$ vector.

2.2.2 Part-Aware LSTM (P-LSTM)

The P-LSTM [11] is based on the understanding that human body joints move together in groups corresponding to a major part of the body. Rather than viewing an action as a composition of moments of individual joints, an action is usually better understood as the interaction between these major body parts.

Building upon the vanilla LSTM implementation, the P-LSTM segments the joint information of the human subject into five logical parts – a torso, two hands, and two legs – and keeps the long-term memory of each part in its own cell instead of having the entire body's motion in a single cell. The context of each body part is then independently maintained, implying an individual input, forget and modulation gate for each part's cell. The output gate, however, is then shared amongst all five body parts. A side-by-side comparison of the vanilla LSTM and P-LSTM is shown in Fig. 2.4.

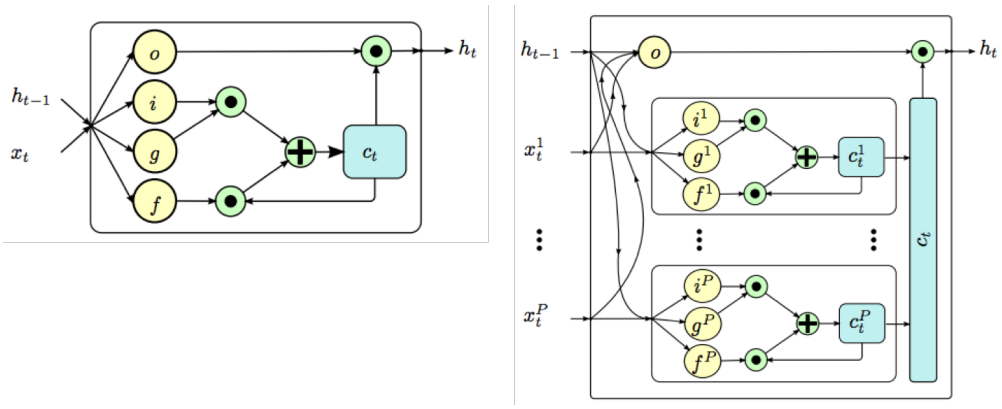


Figure 2.4: A side-by-side comparison of the vanilla LSTM cell (left) and the P-LSTM cell. o indicates the output gate, i is the input gate, g is the input modulation gate, while f is the forget gate. c indicates the cell state of the LSTM, while h_t and x_t are the hidden units and input representation respectively. Source: [11]

Programmatically, the P-LSTM does not require a significant change in the network structure on Tensorflow, but simply a modification on the `RNNCell` class, which is the base class for all variants of the RNN cell. Similar to the vanilla LSTM, the input representation is also a concatenation of joint positions.

2.2.3 Spatio-Temporal LSTM (ST-LSTM)

While both the vanilla LSTM and P-LSTM showed good performance vis-à-vis other non-neural network methods [11], they fail to capture the spatial relationship between different joints at each point of time due to the simple concatenation of joints for input representation. The ST-LSTM attempts to model both the spatial dependencies of joints and temporal dependencies among the frames [12]. One skeletal joint is assigned to a unit in the ST-LSTM as shown in Fig. 2.5, with each unit receiving the hidden representations and cell states of its predecessor units. Hence, the ST-LSTM is analogous to a two-dimensional RNN as introduced by Graves et al. [27].

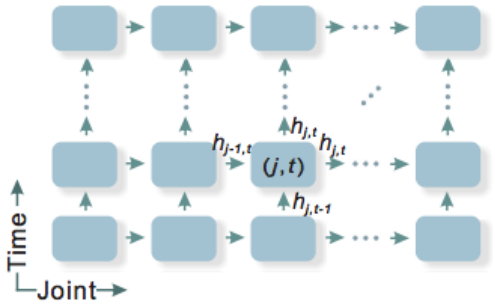


Figure 2.5: An illustration of the ST-LSTM network. In the spatial direction (x-axis), the locations of joints are supplied as a sequence while locations of all joints over time is fed in the temporal direction (y-axis). Source: [12]

Referring to "joint" axis of Fig. 2.5, one could initially represent the sequence of joints as a

simple tree structure illustrated in Fig. 2.6b. However, since trees cannot be directly provided as input into the ST-LSTM, the authors proposed a bidirectional tree traversal method to visit joints in a sequence that keeps the adjacency information of the tree structure as shown in Fig. 2.6c. This ensures that each joint posesses the contextual information of all its neighbors.

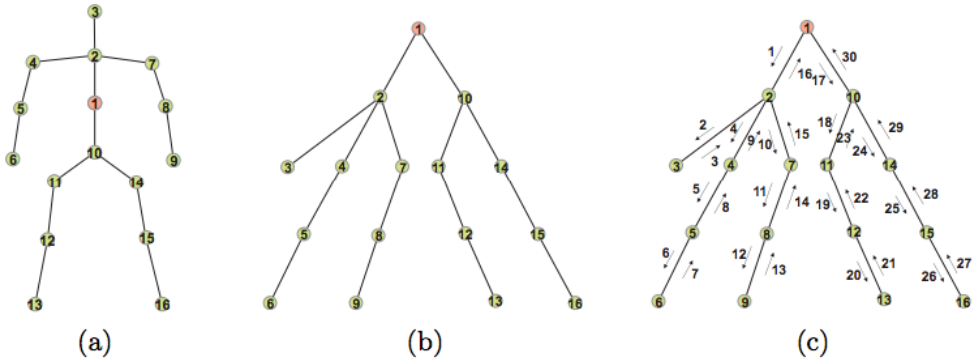


Figure 2.6: a) Skeletal joints of a human body. b) Skeleton represented as a tree structure with the spine center as root. c) Tree traversal of b), resulting in an unfolded chain with the visiting order 1-2-3-2-4-5-6-5-4-2-7-8-9-8-7-2-1-10-11-12-13-12-11-10-14- 15-16-15-14-10-1. Source: [12]

Another advantage of the ST-LSTM is the significant reduction in model parameters since the input at each unit of the ST-LSTM is simply the joint location and the weights are shared across all units in the ST-LSTM. This regularization effort also leads to better generalization [12].

Programmatically, the ST-LSTM requires the use of two for-loops for the unrolling of the LSTM network in both the spatial and temporal dimensions. This is intuitively understood from an earlier illustration Fig. 2.3 where the unrolling of a vanilla LSTM would entail a single for-loop for a single dimension. Tensorflow provides the function `tf.scan` to repeatedly apply a user-defined function to a sequence of elements from first to last [28].

A simplified pseudocode of the function provided to `tf.scan` is shown in Algorithm 2.1. The `stepTime` function can be interpreted as the unrolling of the one-dimensional RNN of joint sequences in the y-axis (time) when referring to Fig. 2.5.

Algorithm 2.1 Procedure for `stepTime` function provided to the `tf.scan` function for creating a ST-LSTM network, which is itself a two-dimensional RNN.

```
1: function stepTime(currentState, input) ▷ currentState represents the accumulated values of the spatio-LSTM states from preceding calls to this function.
2:     inputList ← input
3:     outputAll ← []
4:     for i in numLayers do
5:         cell ← STLSTMCell( numHiddenUnitsAtLayer[i] )
6:         if isTrainingMode then
7:             cell ← DropOutWrapper(cell, dropoutProbabilityAtLayer[i])
8:         end if
9:         output ← static_rnn(cell, inputList)      ▷ static_rnn is a function from TensorFlow that creates a 1-D RNN
10:        inputList ← output
11:        Append output to outputAll
12:    end for
13:    return outputAll
14: end function
```

Additionally, a new cell type has to be defined since the conventional vanilla LSTM cell provided by Tensorflow does not allow for accepting more than one hidden state and one cell state.

2.2.4 LSTM on Lie Group Features

So far, the previous LSTM methods presented simply use joint locations as the network's input representation. Vemulapalli et al., however, showed that one could explicitly model the 3D geometric relationships between various body parts using rotations and translations in 3D space, and hence result in a more meaningful description of an action than their absolute locations in space [8].

Rigid body rotations and translations in 3D space are members of the special Euclidean group $SE(3)$, a matrix Lie group. This results in the relative geometry between any given pair of body parts being a point in $SE(3)$, and the whole human skeleton a point on the Lie group $SE(3) \times ... \times SE(3)$, with $\times$ denoting the direct product between Lie groups. Therefore, human action recognition can be expressed as the classification of curves on $SE(3) \times ... \times SE(3)$.

However, noting that the Lie group is a non-Euclidean manifold, standard classification methods, such as the SVM, cannot be applied. Curves in $SE(3) \times ... \times SE(3)$ have to be mapped to its Lie algebra $\mathfrak{so}(3) \times ... \times \mathfrak{so}(3)$ by using a logarithm map with rolling maps. This method proposed by Vemulapalli et al. reduces distortions introduced by otherwise flattening the Lie group using the logarithm map at a single point [29]. This is pictorially illustrated in Fig. 2.7.

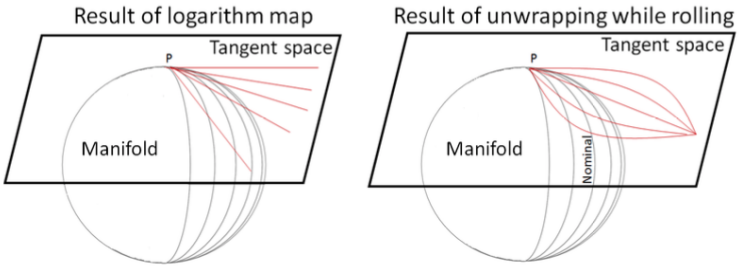


Figure 2.7: Flattening a Lie group maniform with a logarithm map at a single point P produces distortions (left), while this is reduced when unwrapping (via the logarithm map) while rolling along the nominal curve (right). Source: [29]

Upon obtaining a curve on the Lie algebra $\mathfrak{so}(3) \times ... \times \mathfrak{so}(3)$, we simply provide the curve on a high-dimensional space as the input representation into the vanilla LSTM to learn its temporal dependencies. The combination of using Lie group features with a recurrent neural network is the first of its kind, to the best of our knowledge.

2.2.5 LSTM on 2D Skeletal Information

Obtaining 2D skeletal information (x, y) from the original 3D joint positions (X, Y, Z) in camera coordinates can be trivially done by applying the perspective projection equations as follows:

$$x = \frac{fX}{Z}, \qquad y = \frac{fY}{Z}. \tag{2.1}$$

The focal length f does not need to be precisely calculated since it is a scaling constant applied on all skeletal data in the dataset.

Although attractively simple, this method is not a true measure of the efficacy of RGB-only human action recognition, since the 3D joint positions were derived from the RGB-D data. Cao et al. proposed in [30] a method using part-affinity fields for detection of multiple human poses in 2D images. This method addresses the limitations of the skeletal tracking by the Kinect v2 sensor of only being able to reliably track camera-facing bodies. Unlike the Kinect's skeletal tracking algorithm, the number of skeletal poses is also not limited to two in Cao et al.'s method, as shown in Fig. 2.8.

The treatment of 2D skeletal data is analogous to its 3D counterpart, save for the reduction in input dimensions, hence allowing us to use previously introduced methods without much structural changes. To the best of our knowledge, LSTM with 2D skeletal information has also not been previously used in the field of human action recognition.

Figure 2.8: Multi-subject 2D pose estimation using parts affinity fields is robust against occlusions, scale, and orientation of the human subject. Source: [30]

2.2.6 Temporal Segment Networks

The methods introduced thus far use skeletal data (or higher-level representations thereof) for the task of human action classification, exploiting the availability of skeletal data provided in the NTU-RGB+D dataset. However, as explained in Section 1.1, using RGB-D cameras for data acquisition might not be a viable approach in certain application fields.

Instead of preprocessing RGB images to obtain 2D skeletal data as mentioned in the previous section, the images themselves could be used directly as inputs to a learning algorithm, as proposed recently by Huang et al. with the use of temporal segment networks [13].

Instead of using a single frame or a single stack of frames in a short snippet, temporal segment networks operate on a sequence of short snippets sparsely sampled from the input video. Each snippet then produces its own preliminary prediction of action classes after training it with a convolutional neural network (ConvNet). A consensus amongst the snippets is derived for the video-level prediction.

Chapter 3

Validation

This chapter elaborates on the various methods presented in the previous chapter by attempting to reproduce reported state-of-the-art results and present the results for our proposed methods.

3.1 Experimental Setup

The following experiments were implemented on the Tensorflow v1.2 library and ran on GPU clusters with either NVIDIA's K40c or GeForce GTX Titan X GPUs, with no more than one GPU being used at each training procedure. We used the Adam optimization algorithm [31] for the training of all neural networks while varying other parameters described in the following sections.

In experiments using raw skeletal data, we normalized the joints positions relative to the coordinate system with origin defined at the center of the spine (shown as joint no. 2 in Fig. 2.2) and its X and Y axes corresponding to from vector from right (joint no. 9) to left shoulder (joint no. 5) and from spine base (joint no. 1) to spine (joint no. 2) respectively, as proposed in [11]. The normalized Z axis is then simply the cross product of the X and Y axes.

Furthermore, in cases where more than one person is present in the scene, the actor with the highest amount of 3D body motion is chosen. However, unlike [11], we do not consciously filter out noisy detections but instead treat it as an inherent property of the dataset. Since the length of video samples is on average more than 90 frames long as seen in Fig. 3.1, each video sample was downsampled to 20 frames by uniform sampling of 20 equally sized temporal segments to reduce training times as suggested in [12]. Alternatively, we also experimented with setting the maximum number of frames to 192, padding and cropping samples with fewer or more than the threshold, respectively.

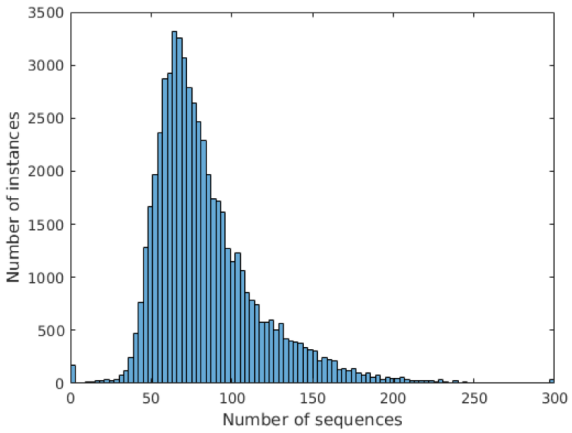


Figure 3.1: A histogram of the number of frames in each video in the NTU-RGB+D dataset.

3.2 Reproduction of State-of-the-Art Methods

We limit the reproduction of the state-of-the art methods to the methods previously introduced in Section 2.2 – the LSTM, P-LSTM and ST-LSTM, since they had exhibited good performance on the NTU-RGB+D dataset.

For all three methods, we limit the number of layers to two as stated in [11] and [12]. The learning rate schedule $\eta(t)$ is defined as:

$$\eta(t) = \eta_0 \cdot 0.95^{\lfloor (t/3000) \rfloor}, \tag{3.1}$$

with an initial rate $\eta_0 = 0.003$ and where t indicates the number of iterations (or the `global_step` in Tensorflow's context) in the training process. This implies a 5% drop in learning rate every 5 epochs assuming an average of 600 iterations per epoch.

While the number of hidden units in each layer was specified in [12] to be 128, no information on such details was provided in [11], hence leaving us to set an upper boundary to 512 hidden units per layer and experimented with different configurations of the number of hidden units.

The experimental results of the three state-of-the-art methods are presented in Table 3.1 and their corresponding confusion matrices for the LSTM and P-LSTM methods are provided in Fig. 3.2. Additionally, the top 5 errors as obtained from the confusion matrices are summarized in Table 3.2. We also note our inability to reproduce the state-of-the-art results reported on the ST-LSTM.

We observe clear limitations of only using skeletal information for human action recognition in the actions the model fails to distinguish – actions that involve the interactions with other objects or subjects that are not captured in the skeletal data used for training. Distinguishing between "finer" movements such as clapping and rubbing hands or putting on and taking off a shoe is understandably difficult due to the lack of additional appearance-based information or accurate information of the smaller body joints such as fingers and hands.

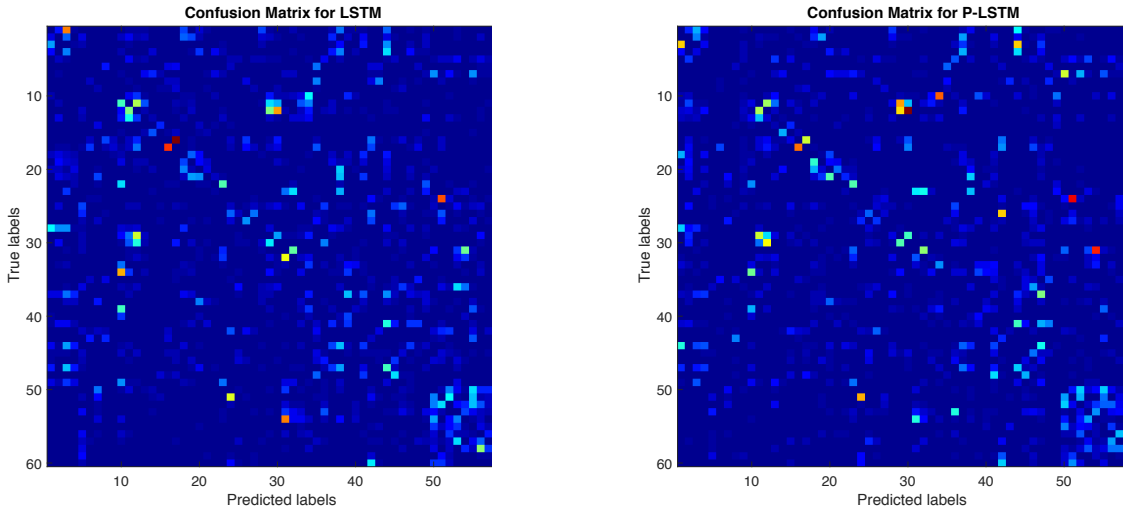
Method (No. hidden units per layer)	No. frames per sample	Cross-subject accuracy (ours)	Cross-subject accuracy (reported)
LSTM (512+256)	20	57.9%	60.7%
P-LSTM (500+500)	192	62.3%	62.9%
ST-LSTM w/o Trust Gate (128+128)	20	52.7%[1]	65.2%

Table 3.1: Performance of various state-of-the-art algorithms for human action recognition using 3D skeletal information.

Method	Top 5 Errors	
	Truth	Predicted
LSTM	"Put on shoe"	"Take off shoe"
	"Take off shoe"	"Put on shoe"
	"Kicking something"	"Kicking another person"
	"Drink water"	"Brushing teeth"
	"Pointing to someone"	"Pointing to something"
P-LSTM	"Writing"	"Typing on Keyboard"
	"Kicking Something"	"Kicking another person"
	"Pointing to something"	"Pointing to someone"
	"Clapping"	"Rubbing hands together"
	"Take off shoe"	"Put on shoe"

Table 3.2: Top 5 prediction errors made by the LSTM and P-LSTM.

[1] We achieved this result only with a 2-layer ST-LSTM each with 256 hidden units, while [12] uses 2 layers of 128 units each.



(a) Confusion matrix for the LSTM method.

(b) Confusion matrix for the P-LSTM method.

Figure 3.2: Confusion matrices for both LSTM and P-LSTM methods. Note that the true positives (i.e. diagonal entries of the matrices) have been excluded from the matrix for better visualization.

3.3 Experimental Results on our Proposed Methods

This section presents the results of our proposed methods – the LSTM on Lie Group features and the (P-)LSTM on 2D skeletal information.

3.3.1 LSTM on Lie Group Features (LieLSTM)

Unlike the LSTM and P-LSTM described in the previous section, all action curves are required to have the same number of samples, regardless of how long each action sequence is. We fix the number of samples for each video sequence to be 20 for computational tractability, given that each point on the Lie algebra space has a dimension of 1656 – significantly larger than the concatenated 3D coordinates in the case of the LSTM and P-LSTM.

Using the LSTM trained on Lie group features, we achieved results comparable to the P-LSTM with a cross-subject accuracy of 62.0%. The model was trained with two layers of 512 and 256 hidden units, with a dropout of 0.2 at the first layer and no dropout at the second. The learning schedule was initiated at $\eta_0 = 0.003$ and is defined by (3.1). Intuitively, like the previous state-of-the-art methods, the LieLSTM also fails to differentiate between almost the same types of action classes as with the state-of-the-art as shown in Table 3.3. In this respect, Lie Group features themselves are only as expressive as the input modality they derives from.

3.3.2 Two-Stream P-LSTM + LieLSTM

Noting the good performance of both our LieLSTM method and the P-LSTM method [11], a possible increase in accuracy could be achieved by combining the two network streams. Similar to other two-stream approaches for human action recognition ([9], [14], [16]) that exploit the spatial and

Method	Top 5 Errors	
	Truth	Predicted
LieLSTM	"Clapping"	"Rubbing hands together"
	"Put on shoe"	"Take off shoe"
	"Writing"	"Typing on keyboard"
	"Kicking someone"	"kicking something"
	"Writing"	"Fiddling with phone"

Table 3.3: Top 5 prediction errors made by the LSTM on Lie Group Features (LieLSTM).

temporal strengths of each stream respectively, our Two-Stream LieLSTM + P-LSTM leverages the spatio-temporal awareness of a LieLSTM and the spatial awareness of individual body parts of the P-LSTM.

Each stream is trained independently, using two different modalities – Lie group features and normalized joint positions. Subsequently, a weighted average of the softmax layers of both streams is taken for the final prediction. A graphical illustration of the two-stream network can be seen in Fig. 3.3. We experimented with several weighting combinations and noticed a significant increase in performance compared to that of each stream itself as shown in Table 3.4.

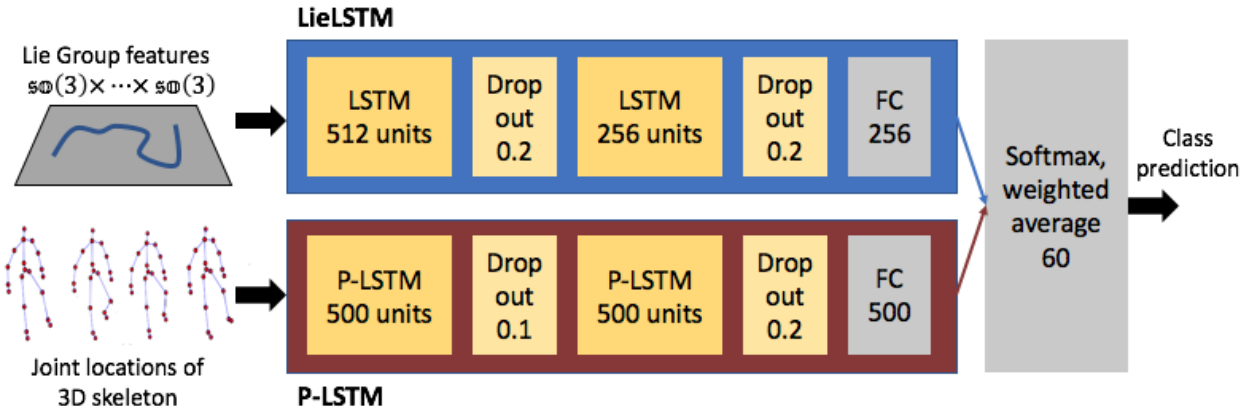


Figure 3.3: An illustration of our proposed Two-Stream P-LSTM + LieLSTM.

An overview of the performance of several state-of-the art methods is presented in Table 3.5. We observe that our two-stream result is also comparable to that of the ST-LSTM although still considerably less accurate than the GCA-LSTM.

3.3.3 LSTM / P-LSTM on 2D Skeletal Information

For experiments using the LSTM and P-LSTM on 2D skeletal information, we also fixed the length of each video sequence to 20 as described in Section 3.1. Unlike for the 3D skeletal data, we do not perform any normalization on the 2D skeletal data and simply represent them in camera coordinates. The results are shown in Table 3.6. We note a significant performance decrease with a reduction in dimension in skeletal information. The comparably worse performance of the P-LSTM to the LSTM points towards a possible counterproductiveness in trying to learn long-term

Methods	Cross-subject accuracy
LieLSTM	62.0%
P-LSTM	62.3%
Two-Stream 30% P-LSTM, 70% LieLSTM	64.1%
Two-Stream 40% P-LSTM, 60% LieLSTM	65.8%
Two-Stream 70% P-LSTM, 30% LieLSTM	66.2%
Two-Stream 60% P-LSTM, 40% LieLSTM	67.7%
Two-Stream 50% P-LSTM, 50% LieLSTM	**68.9%**

Table 3.4: Action recognition accuracies of weighted Two-Stream P-LSTM + LieLSTM networks compared with the single stream variants.

Method using skeletal information only	Cross-subject accuracy
2-Layer LSTM [11]	60.7%
LieNet-3Blocks [13]	61.4%
2-Layer P-LSTM [11]	62.9%
2-Layer ST-LSTM [12]	69.2%
GCA-LSTM network [15]	**74.4%**
Two-Stream P-LSTM + LieLSTM	68.9%

Table 3.5: Action recognition accuracies of the various state-of-the-art methods along with our proposed LSTM on Lie Group features method tested on the NTU-RGB+D dataset.

dependencies of parts where depth information is lacking. Similarities across action classes at different body parts might be accentuated especially where depth information would have played an important role in distinguishing one action from the other.

We also note that the most common prediction errors are still mainly those that are inherently difficult for LSTM methods using 3D skeletal information, and not from action pairs that could have been difficult to distinguish with the lack of depth information, as seen from Table 3.7.

Using 2D Skeletal Information where 3D Information fails

The authors of [11] explicitedly noted 302 video samples from the NTU-RGB+D dataset where 3D skeletal information was missing or incomplete, while a couple of samples where skeletal information was present but noisy was shown in [12]. While the noisy videos were not specifically identified in [12], a list of the 302 video samples, which we call the *Kinect failure cases* for ease of future reference, was provided [18]. Upon visual inspection of the Kinect failure cases, we concluded that the 2D human pose could be estimated with state-of-the-art methods. To this end, we applied the parts affinity fields-based method by Cao et al. for 2D human pose estimation [30] on all Kinect

Method	Cross-subject accuracy
2-Layer LSTM 2D	51.8%
2-Layer P-LSTM 2D	47.7%

Table 3.6: Action recognition accuracies of LSTM and P-LSTM trained on two-dimensional skeletal data.

Method	Top 5 Errors	
	Truth	Predicted
LSTM on 2D skeletal information	"Clapping"	"Rubbing hands together"
	"Writing"	"Typing on keyboard"
	"Writing"	"Fiddling with phone"
	"Patting someone on the back"	"Point finger at another person"
	"Reading"	"Fiddling with phone"

Table 3.7: Top 5 prediction errors made by the LSTM on 2D skeletal information.

failure cases.

Cao et al.'s method was able to correctly identify the human poses in 220 of the 302 Kinect failure cases and extract 2D skeletal information from the RGB video sequence. A histogram of both failure and success of the 2D skeletal information extraction algorithm is shown in Fig. 3.4. Some snapshots from both success and failure cases with this method is highlighted in Fig. 3.5 and Fig. 3.6 respectively.

We noticed Cao et al.'s method succeeded in detected human poses even when the subject is not facing the camera – a use case that the Kinect v2 sensor was not designed to undertake according to its documentation. Amongst the failure cases were also videos with front facing subjects, which we presume were out of the Kinect's effective range for skeletal tracking. In these samples, even at a video resolution of 400 x 224 pixels, Cao et al.'s method proved to work as seen from the first and third images in Fig 3.5. However, scenes such as "hugging" proved to be a challenging due to heavy occlusions of a subject's body parts. In most of the cases involving the "hugging" action class, Cao et al.'s method simply estimated a single human skeleton of the, however with inaccurately positioned arms (usually the arms from the other subject).

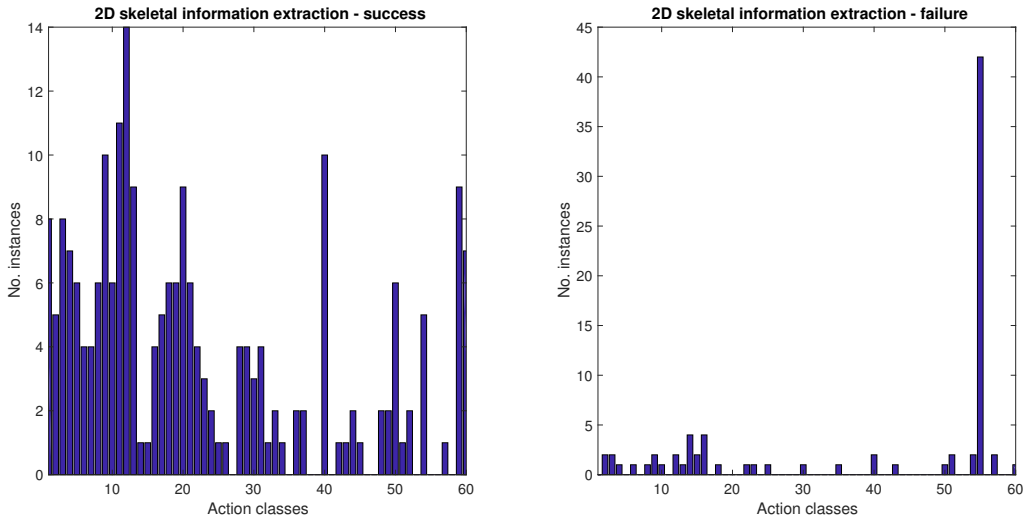


Figure 3.4: Histograms on both success and failure cases in the extraction of 2D skeletal information from the 302 "failure cases" in the NTU-RGB+D dataset. The top 5 classes where 2D skeletal information could not be extracted from videos are "hugging", "wearing jacket", "putting on shoe", "eat meal/snack" and "brushing hair".

Figure 3.5: Snapshots of videos where the Kinect v2 sensor was not able to detect 3D human poses. However, upon applying Cao et al.'s method [30] on these samples, 2D human pose(s) could be successfully estimated.

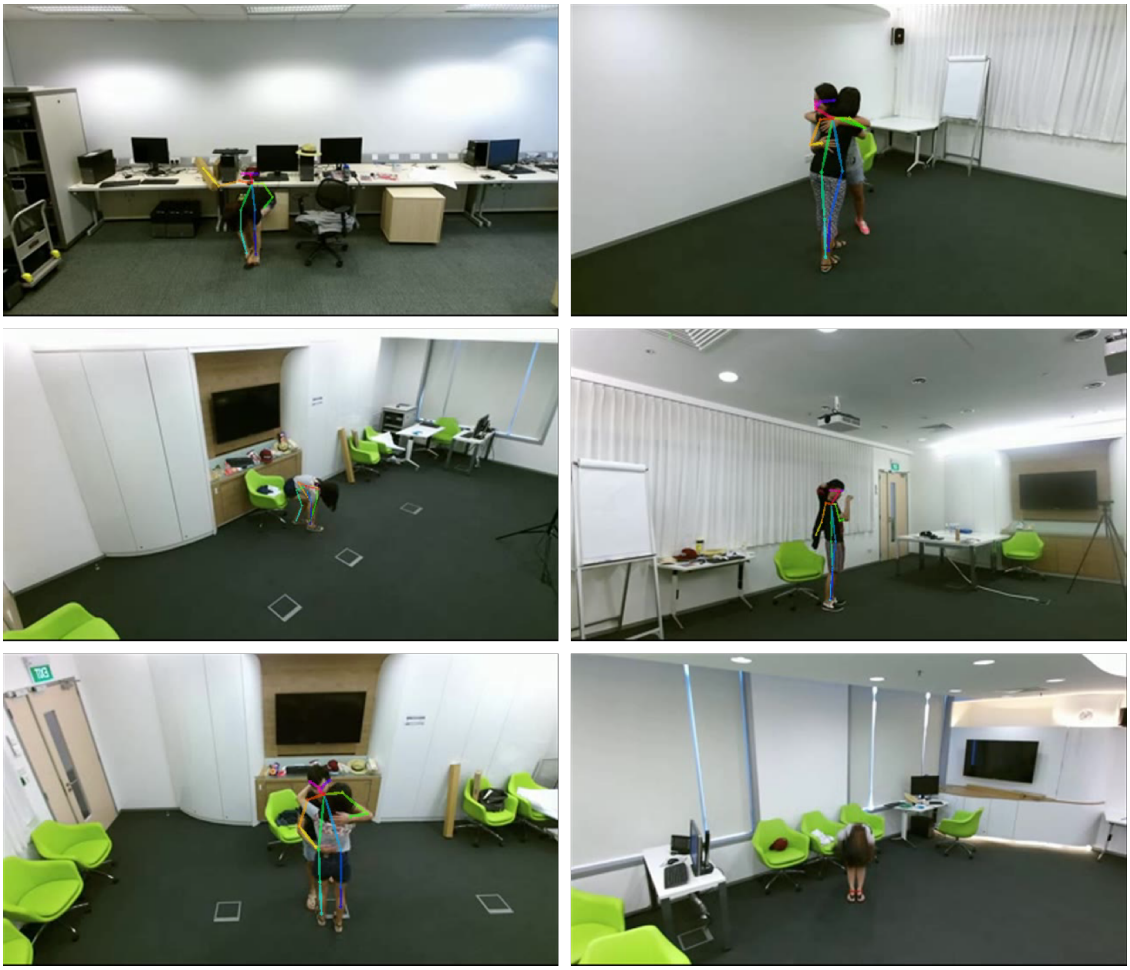


Figure 3.6: Snapshots of videos where the Kinect v2 sensor was not able to detect 3D human poses. Cao et al.'s method [30] also showed here its limitations and could not or wrongly estimated 2D human poses.

3.4 Experimental Results using Temporal Segment Networks

Appearance-based recognition on the NTU-RGB+D dataset exhibited state-of-the-art performance as shown in [14], using the 3D-CNN method [32]. Motivated by these results, we train the temporal segment network solely on RGB videos of the NTU-RGB+D. As proposed in [16], we also choose the Inception with Batch-Normalization model [33] as a building block and fine-tune the weights for the NTU-RGB+D dataset.

As of this writing, after fine-tuning for 15400 iterations with a batch size of 128, a learning rate schedule $\eta(t)$ as follows:

$$\eta(t) = \eta_0 \cdot 0.9^{\lfloor (t/2000) \rfloor}, \tag{3.2}$$

where the initial rate is $\eta_0 = 0.001$ and t is the number of iterations, we obtain a best-shot cross-subject accuracy of 50%.

Chapter 4

Conclusion and Outlook

In this work, we presented a unified framework for reproducing several state-of-the-art results using deep learning algorithms for human action recognition on the NTU-RGB+D dataset. Additionally, we introduced a novel method coined the LieLSTM, a LSTM trained on Lie Group features, and demonstrated its comparable effectiveness with many state-of-the-art results. By using a two-stream approach averaging the predictions of the LieLSTM and the P-LSTM, we furthermore achieved comparable performance to the state-of-the-art, amongst methods only utilizing 3D skeletal data as input modality

Tangential to the multimodal learning algorithms proposed increasingly with the NTU-RGB+D dataset, we introduced a method using LSTMs on only 2D skeletal data as an outlook to achieving a more generalized approach for action recognition. We also showed an optimistic use-case for using 2D skeletal information for human action recognition where 3D skeletal information cannot be obtained.

Human action recognition is a broad field of research, and owing to its many applications, is also a field with no one single direction that should be taken. Nonetheless, we showed the effectiveness of applying deep learning methods on different modalities for action recognition – further echoing the now commonplace opinion in computer vision.

Next steps could be taken with a combination of 2D human pose estimation (e.g. [30]) and deep learning algorithms on 2D skeletal data to allow for action recognition on RGB videos and potential achieve comparable performances to its RGB-D counterpart. When comparing against methods using only RGB input, learning on 2D skeletal data could be computationally more performant than existing methods that learn on RGB or even optical flow images. Additionally, learning on 2D skeletal data leverages on the rapid progress in robust and realtime algorithms for 2D human pose estimation, ensuring that the learning process would potentially improve as its input quality improve.

Alternatively, building on the promising results of the LieLSTM and Huang et al.'s work [13] of using deep neural networks to learn Lie Group representations for action recogntion, future works could see the integration of the learning of Lie Group representations in a recurrent neural network to better capture the temporal dependencies of an action.

The introduction of a large-scale human action recognition dataset like that of the NTU-RGB+D has certainly been a boon to the development and validation of deep learning algorithms. However, its highly unnatural setting inadvertently limits it to remain as a research dataset. One then questions the feasibility of performing transfer learning from the NTU-RGB+D dataset to other

smaller-scale, albeit real life datasets. All these questions remain to be explored, not only for the field of human action recognition, but also for deep learning in general.

Bibliography

[1] J. Wu, Y. Zhang, and W. Lin, "Towards good practices for action video encoding," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[2] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *Proceedings of the 2013 IEEE International Conference on Computer Vision (ICCV)*, Dec 2013.

[3] I. Sipiran and B. Bustos, "Harris 3d: a robust extension of the harris operator for interest point detection on 3d meshes," *The Visual Computer*, vol. 27, no. 11, Jul 2011.

[4] A. Klaser, M. Marszałek, and C. Schmid, "A spatio-temporal descriptor based on 3d-gradients," in *BMVC 2008-19th British Machine Vision Conference*. British Machine Vision Association, 2008.

[5] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Commun. ACM*, vol. 56, no. 1, pp. 116–124, Jan. 2013.

[6] V. Parameswaran and R. Chellappa, "View invariance for human action recognition," *International Journal of Computer Vision*, vol. 66, no. 1, pp. 83–101, Jan 2006.

[7] X. Yang and Y. Tian, "Effective 3d action recognition using eigenjoints," *J. Vis. Comun. Image Represent.*, vol. 25, no. 1, pp. 2–11, Jan. 2014.

[8] R. Vemulapalli, F. Arrate, and R. Chellappa, "Human action recognition by representing 3d skeletons as points in a lie group," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[9] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proceedings of the 27th International Conference on Neural Information Processing Systems*, ser. NIPS'14, 2014.

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

[11] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "Ntu rgb+ d: A large scale dataset for 3d human activity analysis," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[12] J. Liu, A. Shahroudy, D. Xu, and G. Wang, "Spatio-temporal lstm with trust gates for 3d human action recognition," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[13] Z. Huang, C. Wan, T. Probst, and L. V. Gool, "Deep learning on lie groups for skeleton-based action recognition," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[14] R. Zhao, H. Ali, and P. van der Smagt, "Two-stream RNN/CNN for action recognition in 3d videos," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[15] J. Liu, G. Wang, L. Duan, P. Hu, and A. C. Kot, "Skeleton based human action recognition with global context-aware attention LSTM networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[16] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[17] Microsoft, "Kinect hardware," https://developer.microsoft.com/en-us/windows/kinect/hardware.

[18] A. Shahroudy, "shahroudy/nturgb-d: Info and sample codes for "ntu rgb+d action recognition dataset"," https://github.com/shahroudy/NTURGB-D.

[19] Microsoft, "Skeletal tracking," https://msdn.microsoft.com/en-us/library/hh973074.aspx.

[20] C. Olah, "Understanding lstm networks," http://colah.github.io/posts/2015-08-Understanding-LSTMs/, August 2015.

[21] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *Trans. Neur. Netw.*, vol. 5, no. 2, Mar. 1994.

[22] S. Hochreiter, "Untersuchungen zu dynamischen neuronalen netzen," *Diploma, Technische Universität München*, vol. 91, 1991.

[23] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[24] A. Damien, "Tensorflow-examples/recurrent_network.ipynb at master - aymericdamien/tensorflow-examples," https://github.com/aymericdamien/TensorFlow-Examples/blob/master/notebooks/3_NeuralNetworks/recurrent_network.ipynb.

[25] S. Ozair, "sherjilozair/char-rnn-tensorflow: Multi-layer recurrent neural networks (lstm, rnn) for character-level language models in python using tensorflow," https://github.com/sherjilozair/char-rnn-tensorflow.

[26] D. Hafner, "Variable sequence lengths in tensorflow," https://danijar.com/variable-sequence-lengths-in-tensorflow/, May 2016.

[27] A. Graves, S. Fernández, and J. Schmidhuber, *Multi-dimensional Recurrent Neural Networks*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 549–558.

[28] Tensorflow, "tf.scan — tensorflow," https://www.tensorflow.org/api_docs/python/tf/scan.

[29] R. Vemulapalli and R. Chellappa, "Rolling rotations for recognizing human actions from 3d skeletal data," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[30] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[32] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

[33] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015.