TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Software Science

Allar Viinamäe, 163578 IAPM

DATA DRIVEN GYMNASTICS SKILLS RECOGNITION AND ANALYSIS

Master's thesis

Supervisors: Sven Nõmm, PhD

Tallinn 2020

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Allar Viinamäe, 163578 IAPM

ANDMEPÕHINE VÕIMLEMISOSKUSTE TUVASTUS JA ANALÜÜS

Magistritöö

Juhendajad: Sven Nõmm, PhD

Tallinn 2020

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Allar Viinamäe

17.04.2020

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Gymnastics, a sports discipline with extensive history, requires athletes to have a comprehensive set of physical traits in order to execute exercises. These traits include balance, strength, flexibility, agility and endurance. By developing these physical attributes, the athletes are able to execute motions involving aerial twisting and rotations. This masters thesis focuses on what is known as *tumbling*, a sub-discipline of gymnastics. Some of the foundation movements in tumbling, ofter categorized as flips and handsprings include backflips (aka. a back tuck) and back handsprings. These movements form the backbone of tumbling as they are needed in tumbling to progress onto more advanced movements and are also regarded as individual components of a tumbling routine.

As backflips and back handsprings are the foundational movements for more advanced exercises, they need to be trained regularly and the technique needs to be perfected till these movements become almost intuitive for a gymnastics athlete. Seasoned athletes experience the intriguing physical properties of these exercises, such as the impulse, inertia, rotation and optimal takeoff angle, without consciously thinking about them. For example, kinematic analysis of the backflip in a tumbling series was conducted on beginner and advanced set of athletes to differentiate their techniques [1]. The differences in technique between beginner and advanced athletes contribute to both competition score and the risk of injury, so optimal technique can be considered a priority for every athlete.

At the time of writing, the most common methods for analyzing gymnastics movements require either sophisticated motion capture tools, physically attached sensor data or individual focus and attention from the gymnastics coaches to give valuable feedback to the athletes. In an effort to automate the analysis of technique, give feedback and documenting the history of workouts, this masters thesis focuses on the automatic recognition of backflips and back handsprings using a combination of non-invasive, accessible and state-of-the-art human activity recognition techniques.

1.2 Human Activity Recognition Background

Human activity recognition taxonomy can be challenging as the diversity of available methods is extensive. Broad categorization of human activity recognition methods is depicted on figure 1.1, which is proposed in article [2]. The method used to automate gymnastics activity recognition in this paper is categorized as a *unimodal shape-based method*. While identifying gymnastics movements from multiple modalities (i.e. the addition of behavioral or emotional features) could provide useful data for the analysis, it would require the usage of more invasive tools, such as microphones and physical sensors. The aim of the author is to develop a recognition method less invasive and also free of physical attachments. For example, in competition environments, athletes are not allowed to wear any extra technological gear and the only viable method for recording the activity is the video modality.

Narrowing down on unimodal methods and taking into consideration the non-invasiveness of shape-based methods, brings the author to a method called *pose estimation*, a popular research topic in the last 10 years. More recently, computationally effective methods for 2D pose estimation in real-time using Part Affinity Fields have been proposed [3]. This method is also used by the OpenPose software [4], which is the choice of pose estimation software in this research.

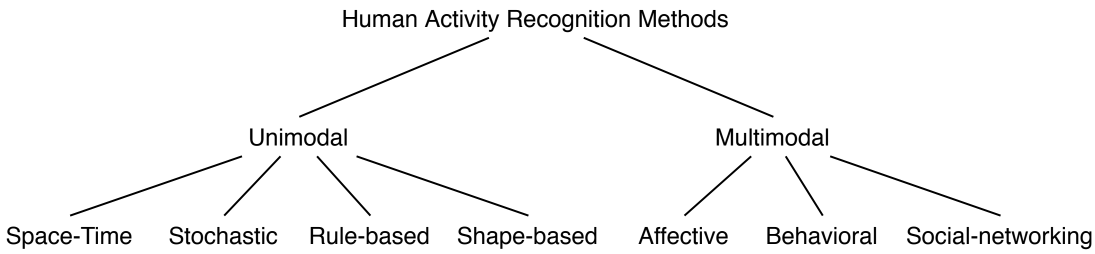

Figure 1.1: Hierarchical categorization of human activity recognition methods

1.3 Problem statement

1.4 Related work

Chapter 2

Implementation

2.1 Implementation overview

This thesis aims to propose an end-to-end type prototype implementation to automate the recognition of gymnastics exercises using computers. High overview of the steps required to achieve this is specified in the following list:

1. Data acquisition
   (a) Human activity recording
   (b) Estimating and extracting keypoints
2. Data exploration
   (a) Pose estimation limitations
3. Data pre-processing
   (a) Applying pre-processing strategies
4. Classifier Training
   (a) Methodology overview
   (b) Classifier training and validation
   (c) Analysis of trained classifiers
5. Prediction explanations for obtained classifiers
   (a) Visualizing neuron activations
   (b) Discussion

2.2 Infrastructure and Tools

2.2.1 Client Infrastructure and Tools

Hardware

Software

2.2.2 Back-end Server Infrastructure

Software

2.2.3 Research and development

Chapter 3

Data acquisition

In order to combine gymnastics activity recognition and computer vision, the data acquisition phase of this thesis begins by collecting the recordings of athletes performing gymnastic activities. The activities chosen are the backflip (section 3.1.3) and the back handspring (section 3.1.4). The motivation behind the chosen activities is that these activities are some of the foundation exercises that athletes use as building blocks for more difficult combinations. These activities are particularly interesting since even seasoned athletes use them in their training regime and try to polish their technique throughout their whole career. The recordings are collected by a consumer grade action camera (section 3.1.5). After recording the activities, human skeleton data will be extracted from the videos using computer vision's technique called *pose estimation*, an important step to transform the data from video format to a data format more suitable for training machine learning models to recognize the activities.

The data to be explored and used to train machine learning models in this thesis plays central part in our implementation. Although, machine learning promises to loosen the strictness of the data used in a system by trying to generalize and adapt the models themselves to the data, the quality of the initial data used to develop prototypes still directly influences the interpretation and value of the outcome. Interpretation of not only the outcome, but the entire solution is needed to have clear understanding of why certain outcomes exist, which in turn helps to spark discussion and spread knowledge gained during an experiment. The value of this research, reusable by peers for new scientific experiments, is also directly influenced by the quality of the initial data.

The data acquisition chapter is split into two parts. The first part (3.1) explains what kind and how much of data the author is collecting and the process behind it. The second part (3.2) explains a more technical approach on how the initial data for the human activity recognition algorithm is acquired from recorded human motions and the tools used to do so.

3.1 Recording Human Actions

3.1.1 Choosing Activities To Record

When it comes to choosing difficult biomechanical activities performed by humans, gymnastics is one on the top of the list. Gymnastic feats don't require any additional equipment by humans, but at the same time physical strength, flexibility and kinesthetic awareness are a must to perform any of the skills demonstrated by elite athletes. Gymnastics is also special in its indisputable need for utilizing every part of the human body. All major muscle groups of an athlete need to be in superior condition to perform certain rotations, jumps and holds. As age is a limiting factor when it comes to peak physical condition, many athletes start their career at a very young age. However, there is no age limit for practicing gymnastics at a recreational level.

Working with gymnastic movements and trying to automate the recognition of this kind of human activity requires a reference or in machine learning terms *training data* to train our machine. The two activities chosen to prove the hypothesis of this paper are known as backflips and back handsprings. While fast and explosive, these activities are similar in their execution and achievable by all levels of gymnastics trainees. As backflips and back handsprings are the building blocks of more complex gymnastic combinations, perfecting the technique of these activities provides athletes with the confidence and skill necessary to move on to more difficult feats. From the need for perfecting the technique of these basic activities comes the motivation to analyze them and automate the feedback loop for the athletes. From the athlete's perspective, it is faster and cheaper to get the feedback from, for example, a portable device with recording capabilities, rather than a gymnastics coach. From the coach's perspective, it is also more convenient to automate the learning process of easier activities so the coach can concentrate on guiding athletes towards more interesting and challenging exercises. Figures 3.1 and 3.2 will provide the reader a visual idea of how backflips and back handsprings are performed respectively.

Figure 3.1: Example of a backflip

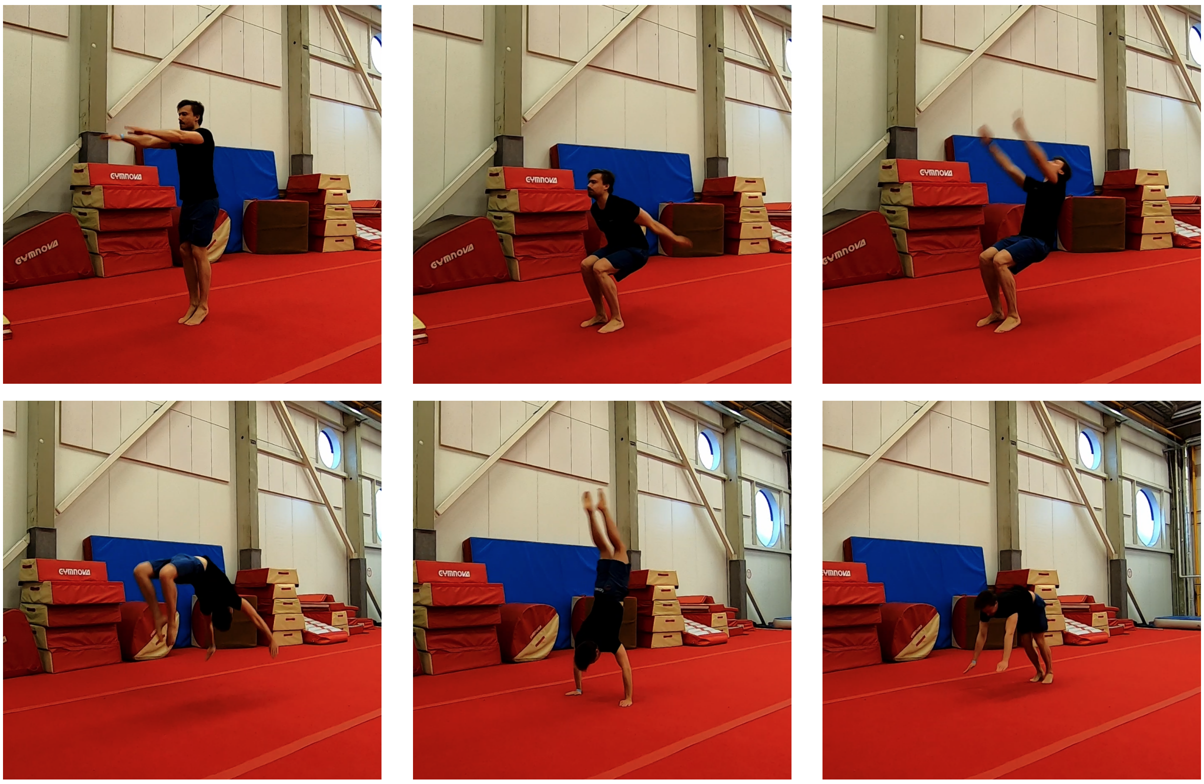

Figure 3.2: Example of a back handspring

3.1.2 Identifying Activities

As with all identifiable activities, it should be clear when the activity starts and when it ends. This paper proposes specific start and end markers for both activities, so they could be easily recognized and validated by a human observer. The markers should cover the full duration of a movement, all while sustaining the integrity and focusing on the period of interest of each activity.

A more detailed explanation of each activity and references to the visual markers can be found in sections 3.1.3 and 3.1.4.

3.1.3 Backflip

A backflip is a sequence of body movements in which a person leaps into the air and rotates backwards over the body's horizontal axis. For the backflip, we mark the start of a backflip as the frame when athlete's both arms pass the horizontal line at shoulder level moving downwards and generating momentum. We mark the end of a backflip as the frame when both heels of the feet touch the ground again. We choose the heels of feet so we can include the amortization part of the landing phase of the backflip in the recording. The red bars in figure 3.3 demonstrate the visual markers for trimming the sample recording to include only the activity under investigation. These markers were chosen by the author to the best of his knowledge of the domain as the clearest points for a human observer to recognize a backflip. Choosing different markers is a potential discussion topic for future improvements.

3.1.4 Back Handspring

A back handspring is similar to a backflip in that the athlete also rotates his body around the horizontal axis. However, during a back handspring, the athlete also moves backwards, while during a backflip the athlete should ideally not move in any direction. The other clear distinction between a back handspring and backflip is that during a back handspring, the athlete's arms extend and push off the floor to create the spring part of the activity and keep the athlete moving backwards. Thirdly, the takeoff angle differs for both back handspring and backflip. The takeoff angle decides in which direction the athlete moves during the rotation. For the

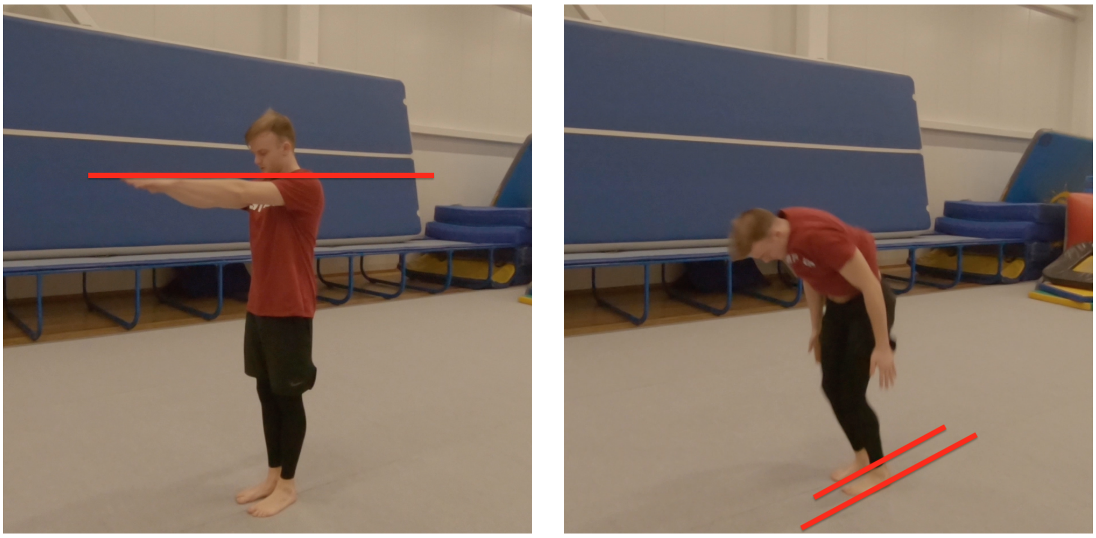

Figure 3.3: Start and end markers for covering the full duration of a backflip

backflip investigated in this paper, the goal of the athletes was to keep the takeoff angle as perpendicular as possible to the floor to keep the athlete from moving either backwards or forwards during the rotation. However, for the back handspring, the takeoff angle should be around 45° backwards to the floor, to help the athlete move backwards and land on the hands. The green bars in figure 3.4 demonstrate the visual markers for covering the full duration of a back handspring. The color of markers are in both cases irrelevant and are chosen primarily for better visual distinction.

3.1.5 The Recording Process and Results

Since the types of movements under investigation are performed for a duration of time, just still images of activities are not sufficient. We need the activities recorded in some kind of motion picture format. The device used to capture activities for this paper is a GoPro Hero7 Black, with the following basic settings:

- *RES (resolution)* — 1080p
- *FPS (frame rate)* — 60
- *FOV (field of view)* — Linear
- *Low Light* — Auto
- *Stabilization* — Auto

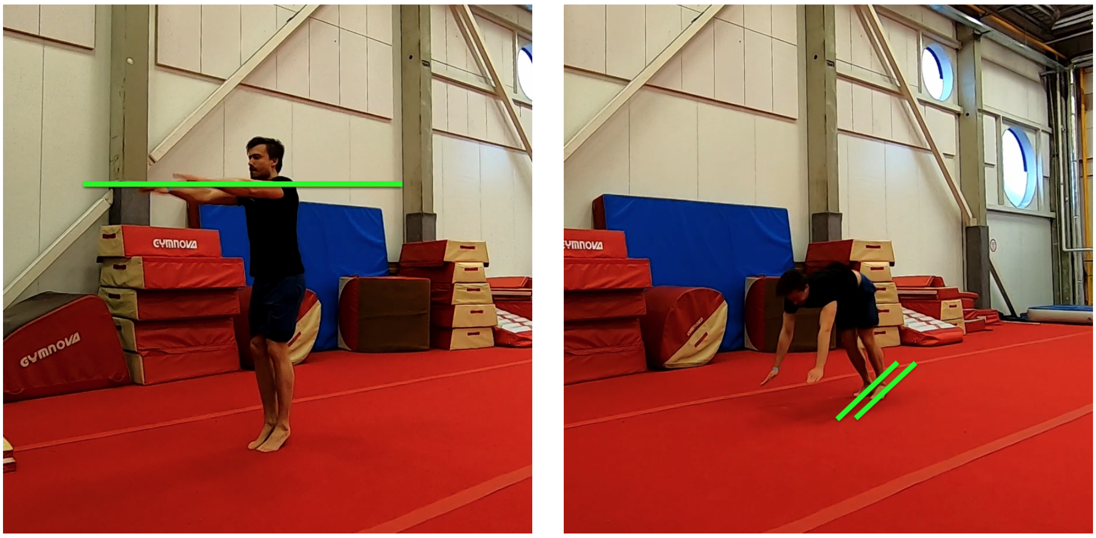

Figure 3.4: Start and end markers for covering the full duration of a back handspring

- *Protune* — Off

Each activity is recorded as one still view from eye level fully showing the subjects body from head to toe. The *three-quarter view* (*two-thirds view*) is chosen to capture each activity. Choosing this angle avoids the limbs of the subject stacking behind each other in the recording. Similar works in this domain use either sophisticated three-dimensional motion capture systems using six cameras with depth sensors [5] or older works using bigger cameras and recording perpendicular to the plane of the movement [1]. While very thorough analyzes, taking into account the center of body mass or exact takeoff and landing angles, they are harder to reuse in a wider variety of difficult biomechanical movements. Mentioned works often require athletes to wear special gear (markers) and also require longer setup times for the recording cameras. The author of this work tries to emphasize on the potential for improving *non-invasive field-based methods* to quantify and monitor technical biomechanical movements. In order to improve these methods, the hypothesis of this work is to use a single consumer camera setup accompanied with a pose estimation solution to achieve satisfactory results for recognizing technical gymnastic activities.

The results of the recording process amounts to a total of 96 backflips and 84 back handsprings captured.

3.2 Estimating and Extracting Human Action Poses

The second section of this chapter focuses on the process of extracting human skeleton data from raw video files captured in the first section of this chapter. The skeleton data will be later preprocessed and used to train machine learning algorithms to recognize gymnastic activities.

3.2.1 Pose estimation

The author's aim is to approach the problem of extracting skeleton data from raw videos using the most state-of-the-art methods. In recent years, some scientific papers have been released using machine learning based methods for estimating the human pose in images. For example, in 2015 an article was published, where convolutional neural networks were used to estimate human poses in videos [6]. These new methods exceed previous solutions in terms of prediction accuracy.

In recent years, even more complete solutions have been proposed, enabling to estimate human poses in real time thanks to their computational efficiency. For example, a method called *Part Affinity Fields*, essentially a set of vectors encoding the direction of a body part, has been proposed. Every limb is described using an *affinity field* between body parts [3]. Such method is used by, for example, an open source solution called *OpenPose* [4]. OpenPose allows to estimate the skeletons of multiple humans from a video recorded with a monocamera. This enables the athlete's skeleton to be identified in an environment with bystanders in the background (a common sight in gymnastic centers).

The author of this work hopes to build a practically usable prototype based on previous work in the pose estimation field. The author uses the free software OpenPose to estimate poses during gymnastic movements. Alternatives to OpenPose performing pose estimation include *PoseNet* [7] and *wrnchAI* [8].

3.2.2 Infrastructure and Tools

The mandatory requirement for using OpenPose's pose estimation in a reasonable time is a dedicated GPU unit and an access to its general purpose computing API [9]. The two main API's supported are CUDA for Nvidia GPU and OpenCL for an AMD GPU. Other tools for using OpenPose include CMake for compiling the OpenPose software and also Python programming environment to access OpenPose's API.

A CPU-only setup of OpenPose is also supported, but highly advised by the author of this work to be used only for testing purposes. For software exploration purposes, the author compiled OpenPose on a 2015 MacBook Pro with a *2,7 GHz Dual-Core Intel Core i5* CPU. The time required for pose estimation in a sample recording averaged to around 30 seconds for a single frame. Since using this setup for pose estimation for a total of 180 samples (recorded at 60 FPS) would result in an unreasonable time spent on this process, the author decided to use an Amazon GPU instance with the following basic settings:

- Operating system — Ubuntu Server 18.04 LTS (HVM), SSD Volume Type
- GPU instance — g4dn.xlarge
- Storage — 16GB

After compiling OpenPose on the Amazon instance, an average pose estimation time of 0.08 seconds was achieved for a single frame in sample recordings.

3.2.3 The Pose Estimation Process and Results

A separate Python module for extracting body key points was developed by the author [10]. The module uses the Python wrapper of *OpenCV* (open-source computer vision library) to access each frame of each sample video. It then uses the Python wrapper of OpenPose on each frame to estimate poses. Finally, the module dumps extracted data to separate comma separated files.

For each sample recording a separate directory is also created. Each directory contains csv files with the filename pattern *activityPerformed-sampleIndex-subjectName.mov-frameIndex-personIndex*, so for example, a valid filename would be *backflip-1-allar.mov-46-0.csv*. A total of 19119 files were generated from 180 sample recordings. The figure 3.5 shows an example output dumped into a csv file by the author's written Python OpenPose extraction module. The X and Y columns mark the corresponding coordinates and the *Confidence score* column represents the confidence factor by OpenPose when estimating body parts. Finally, the I column represents the body part index estimated by OpenPose. These indexes match the body part model *BODY_25* shown in figure 3.6.

The extraction module automatically ran for every sample and generated a total of 19119 csv files. To give it some context, 19119 csv files amount to 5 minutes and 19 seconds of backflips and back handsprings recorded. Using the Amazon GPU instance, the total duration for processing all samples took less than 30 minutes. An estimate could be made for running the same process on a 2015 MacBook Pro with CPU-only setup. The process would take around 6 days and 15 hours if estimated on the basis that the average processing time of each frame is 30 seconds. In conclusion, running the process on an Amazon GPU instance is vaguely more than 300 times faster. This concludes this chapter on data acquisition. The next chapter explores how this data is explored and preprocessed for machine learning training.

I	X	Y	Confidence score
1	1.092518798828125000e+03	4.508595275878906250e+02	8.697314858436584473e-01
2	1.095280273437500000e+03	5.039057617187500000e+02	9.449880123138427734e-01
3	1.063033203125000000e+03	5.036833190917968750e+02	8.230457901954650879e-01
4	1.015858398437500000e+03	4.213815002441406250e+02	8.257341980934143066e-01
5	9.983654174804687500e+02	3.419716186523437500e+02	8.841910362243652344e-01
6	1.124838012695312500e+03	5.068528442382812500e+02	8.021396994590759277e-01
7	1.121924560546875000e+03	4.126282043457031250e+02	8.438682556152343750e-01
8	1.092484497070312500e+03	3.214095458984375000e+02	8.586141467094421387e-01
9	1.074806030273437500e+03	6.920960693359375000e+02	8.156118392944335938e-01
10	1.051205078125000000e+03	6.891304931640625000e+02	7.759792208671569824e-01
11	9.834980468750000000e+02	7.981050415039062500e+02	8.500082492828369141e-01
12	1.045180419921875000e+03	9.069448242187500000e+02	9.030723571777343750e-01
13	1.098396118164062500e+03	6.950711059570312500e+02	7.734714150428771973e-01
14	1.024742309570312500e+03	8.127473754882812500e+02	9.561880826950073242e-01
15	1.092337402343750000e+03	9.305974731445312500e+02	9.035368561744689941e-01
16	1.092454467773437500e+03	4.478778686523437500e+02	2.429898679256439209e-01
17	1.107154418945312500e+03	4.420529479980468750e+02	7.689275145530700684e-01
18	0.000000000000000000e+00	0.000000000000000000e+00	0.000000000000000000e+00
19	1.118930419921875000e+03	4.627049255371093750e+02	5.308289527893066406e-01
20	1.048442871093750000e+03	9.688068847656250000e+02	8.219341039657592773e-01
21	1.068972167968750000e+03	9.716963500976562500e+02	8.162919282913208008e-01
22	1.101324096679687500e+03	9.423051757812500000e+02	7.726828455924987793e-01
23	1.001352600097656250e+03	9.394185791015625000e+02	7.975240349769592285e-01
24	1.004130065917968750e+03	9.305712280273437500e+02	7.877947092056274414e-01
25	1.048436645507812500e+03	9.129010009765625000e+02	7.727123498916625977e-01

Figure 3.5: Sample body parts estimated by OpenPose for one frame

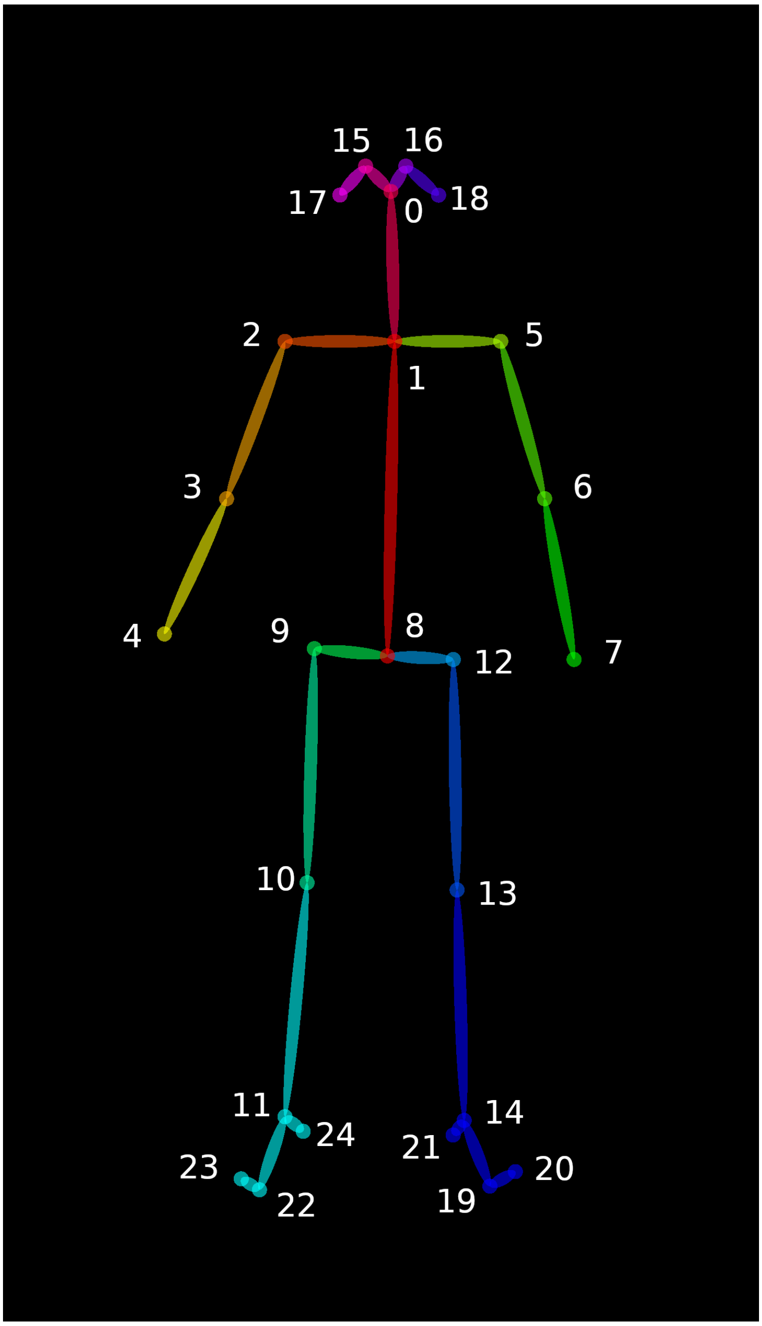

Figure 3.6: Pose Output Format — BODY_25

Chapter 4

Data pre-processing

An essential step between acquiring of the data and passing it to machine learning algorithms is ensuring that the dataset used for training represents our learning objectives. In this case, our learning objective is to train the system to recognize two of the well-known gymnastics movements - the backflip and the back handspring. A well-balanced and interpretable dataset is necessary for avoiding overfitting or biasing the machine learning model to a specific dataset.

In section 4.1, we start by sampling the data and visualizing it to better understand the pose estimation results obtained in the previous chapter. Then, based on the observations, pre-processing strategies for overcoming the shortcomings of the dataset are proposed by the author in section 4.2 and lastly the final dataset, ready to be used for machine learning, is described in the 4.3 section of this chapter.

4.1 Data exploration

During the data acquisition phase, individual csv files (figure 3.5) were generated for each frame of each sample recording. Since the data is generated for each frame, one suitable visualization method is the time-series line plot. Each frame's data contains 25 estimated keypoints, both X and Y coordinates for each keypoint and also the Confidence Score issued by OpenPose. Given the amount of dimensions for such dataset, a sample entity with the following parameters is chosen for demonstration purposes:

- *Sample activity* — Back handspring
- *Sample no.* — 17
- *Keypoint index* — 21 (left heel), figure 3.6
- *Axis* — Y

There is no particular reason for choosing the left heel of the subject, other than given the rotation of the subject's body during the sample activity, we can expect the data range of the left heel's position to vastly differ on the Y axis. This is because the subject's body will be upside down at some point during the movement.

The figure 4.1 represents the subject's left heel's trajectory along the Y axis during the back handspring. For the first second of the activity, the pose estimation seems to not have problems recognizing the left heel during the momentum generation phase. For the takeoff, rotation and landing phase the pose estimation's confidence score falls under the threshold and results in filling the low confidence frames with the zero values. This makes the data not usable by machine learning algorithms, since such strong deviations can be labeled as anomalies and will most likely strongly affect the gradient computed during the backpropagation, ultimately making the algorithms learn something else besides the activities investigated.

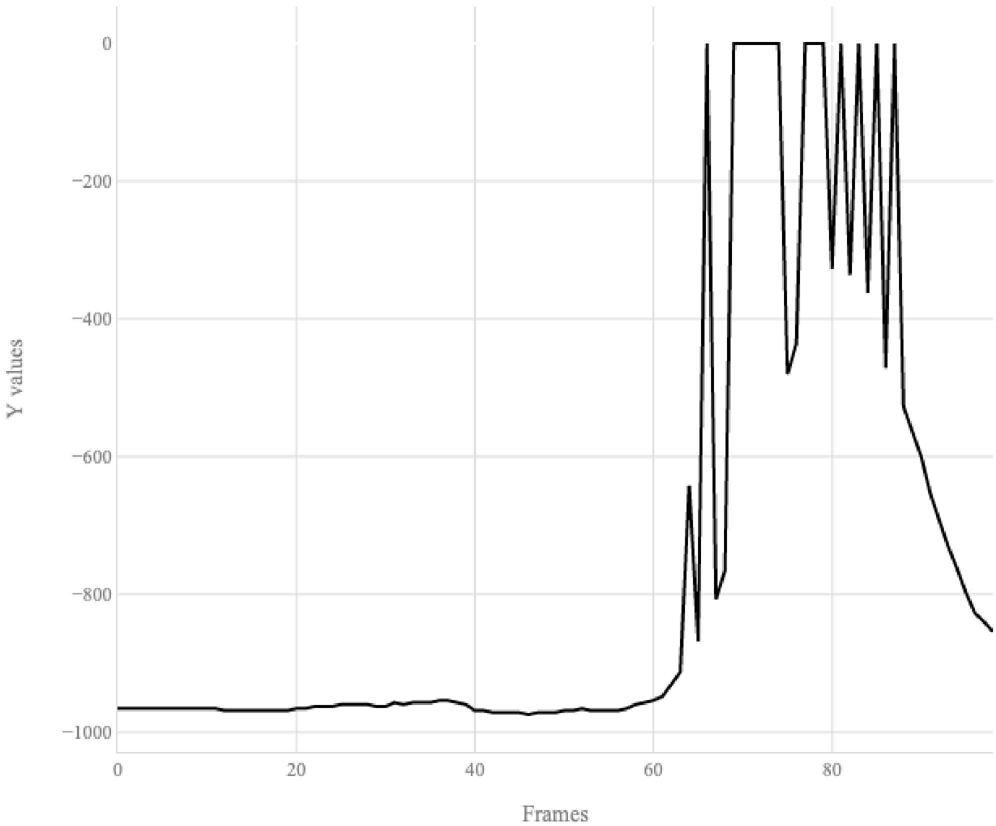

Figure 4.1: Subject’s left heel trajectory during the back handspring - raw data

4.2 Data pre-processing strategies

Four mitigation steps are chosen to overcome the shortcomings of initial pose estimation results. It is worth mentioning, that for clear interpretation purposes the strategies described in next sections are demonstrated on a single sample. Python scripts were developed by the author to automatically apply all pre-processing strategies on every sample's every body part for all frames.

4.2.1 Dealing with low confidence keypoints

During the pose estimation process described in Chapter 3, a confidence score in the range of 0-1 is issued for each keypoint by OpenPose. The confidence score represents OpenPose's certainty when determining keypoints of the subject. Confidence score of 0 by OpenPose means the system fails to estimate a particular body part in a frame. For a keypoint with confidence score approaching 1, translates to a high confidence in detecting the specific keypoint.

A simple algorithm was implemented by the author to fill the missing keypoints - moving through frames and filling missing keypoints with the averages of existing keypoint with positive confidence score, figure 4.2. The improved sample's subject left heel trajectory can been seen on figure 4.3.

4.2.2 Moving average smoothing

During the pose estimation process described in section 3.2.3, the OpenCV library is used to access each frame of the recordings. The frame's data is then fed into OpenPose's estimation function and estimated keypoints are obtained for each frame. The process is stateless by design, so nor previous or next estimations are taken into account when estimating the current frame. This results in fine-grained variations between frames. *Moving Average Smoothing* is a technique applied to time series data in hope to remove noise and better expose the signal of the underlying process.

We define the moving average in the formula 4.1. More specifically, the moving average formula defined can be categorized as the *centered moving average*. We use the centered version of the moving average, since all values in the set are known prior to the smoothing phase. In the formula, we use n of 3 and define it as the

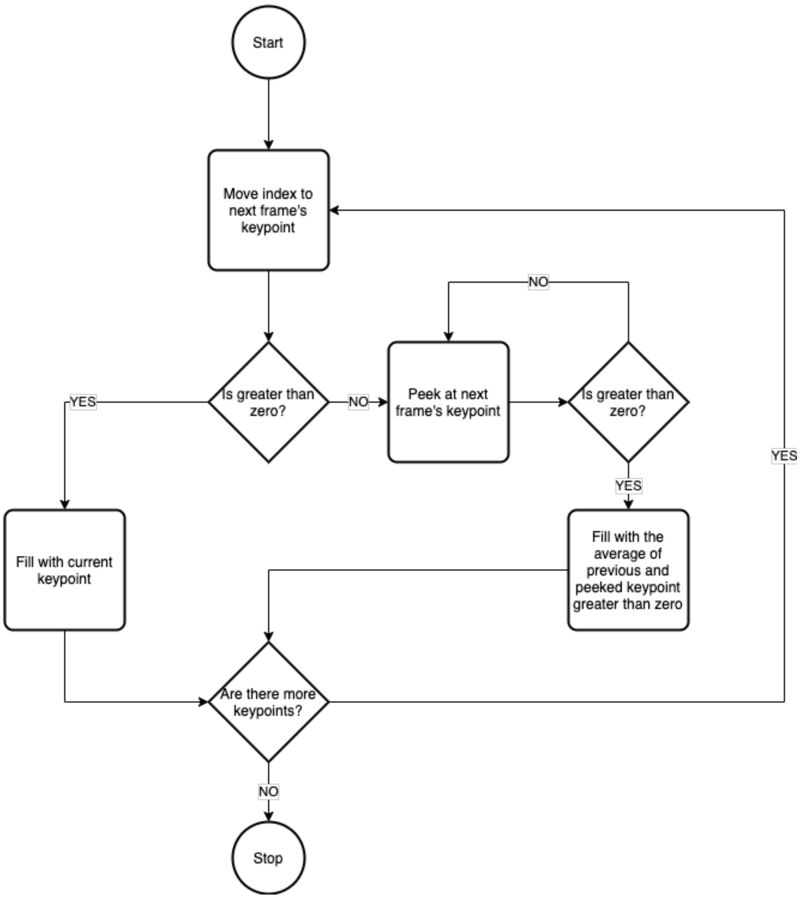

Figure 4.2: Moving average filling algorithm

width of the moving window. The x defined in the formula is the set of values for each body part coordinate for every frame of the activity. Figure 4.4 displays the transformed left heel's trajectory after applying the smoothing technique.

$$x(t) = \frac{1}{n} \sum_{i=-\left\lfloor \frac{n}{2} \right\rfloor}^{\left\lfloor \frac{n}{2} \right\rfloor} x_{t+i} \tag{4.1}$$

4.2.3 Unrecognizable body parts

One limitation worth mentioning when using the pose estimation technique is the contrast between the background and the athlete in the recordings. In low light and low contrast environments, the body parts of an athlete are not easily recognizable (as demonstrated in figure 4.5), which leads to low confidence score for some body parts and makes it impossible to construct a full skeleton. Possible solutions (not

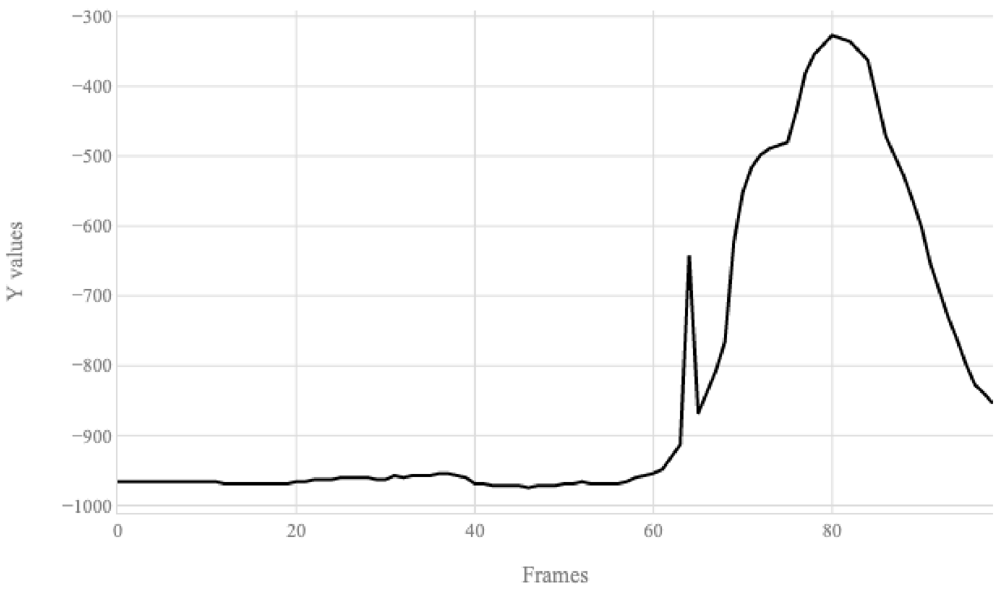

Figure 4.3: Subject's left heel trajectory during the back handspring - filled

explored in this thesis), to improve the recognition of body parts in low contrast environments, include controlling the contrast of the recordings with some post production software or rerecording the activities with athletes wearing clothes with a higher contrast against the background. For this thesis, however, the samples with unrecognizable body parts were left out of the dataset during the preprocessing phase.

4.2.4 Centering to unified coordinate origin

Inspired by data pre-processing methods in [11], the authors Yong Du, Wei Wang and Liang Wang remarkably point out that human actions are independent of its absolute spatial position. Since the starting coordinates of gymnastics movements were not defined prior to recording the actions, normalizing the samples to a unified coordinate origin greatly decreases fluctuations between the same features. All samples in this paper are normalized to the coordinate system origin using the middle hip keypoint indexed as 8 in the *BODY_25* model. The figure 4.6 demonstrates how the skeleton moves relative to the coordinate system origin after normalization.

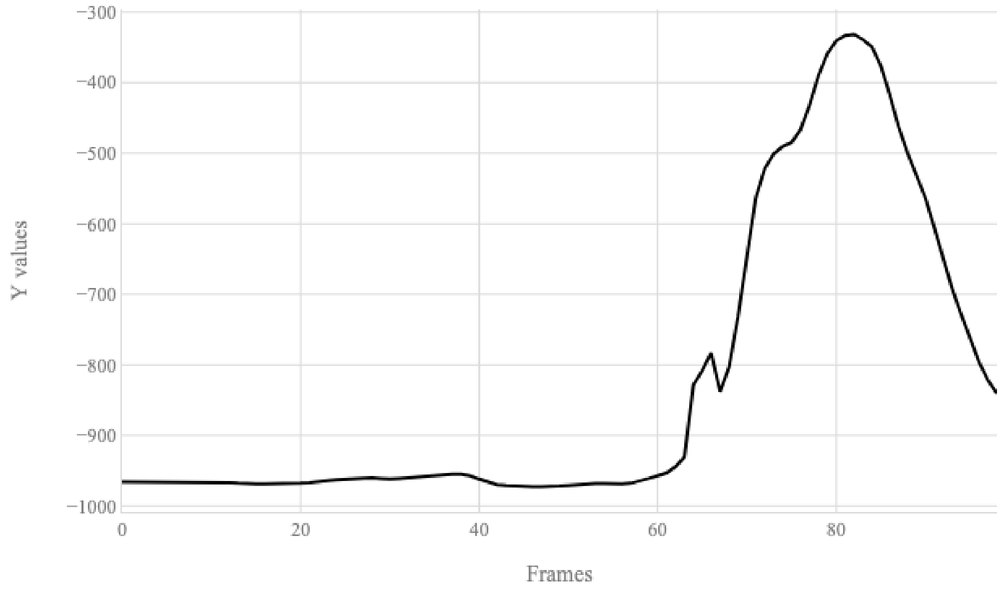

Figure 4.4: Subject's left heel trajectory during the back handspring - smoothed

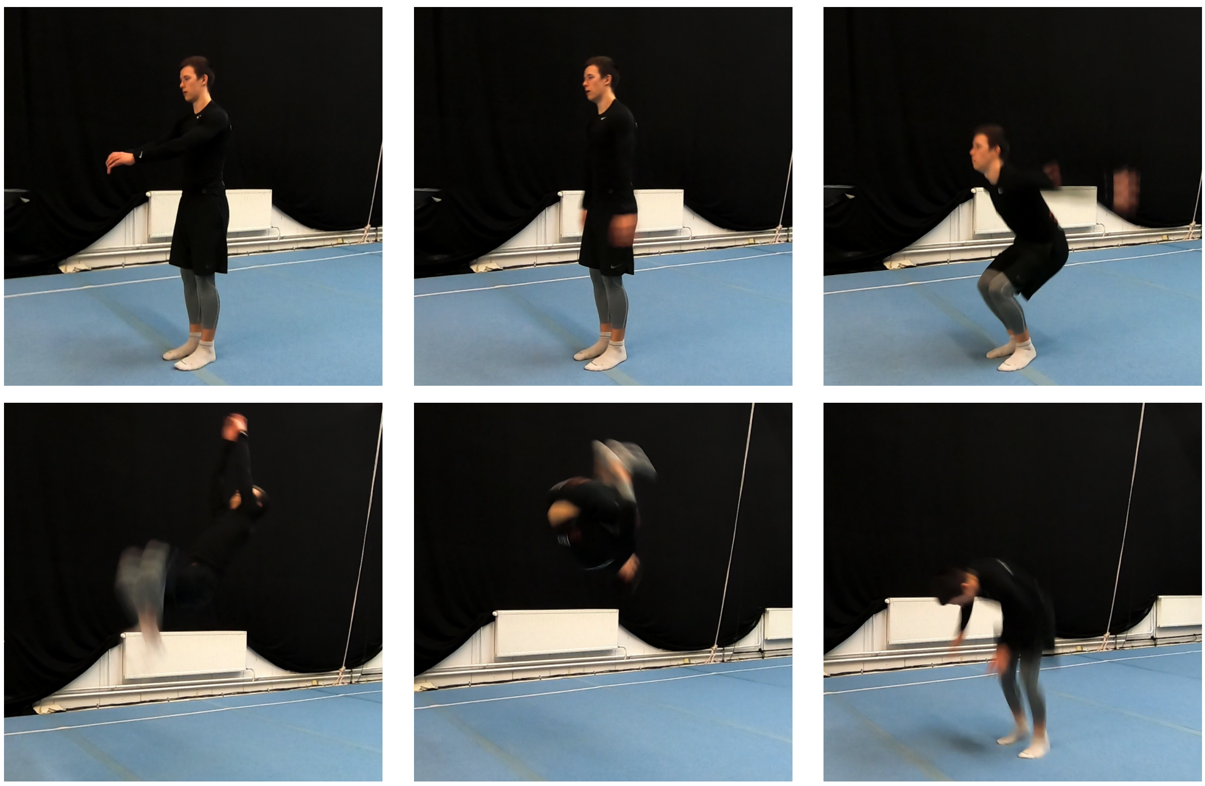

Figure 4.5: The movement of some body parts for this backflip are unrecognizable

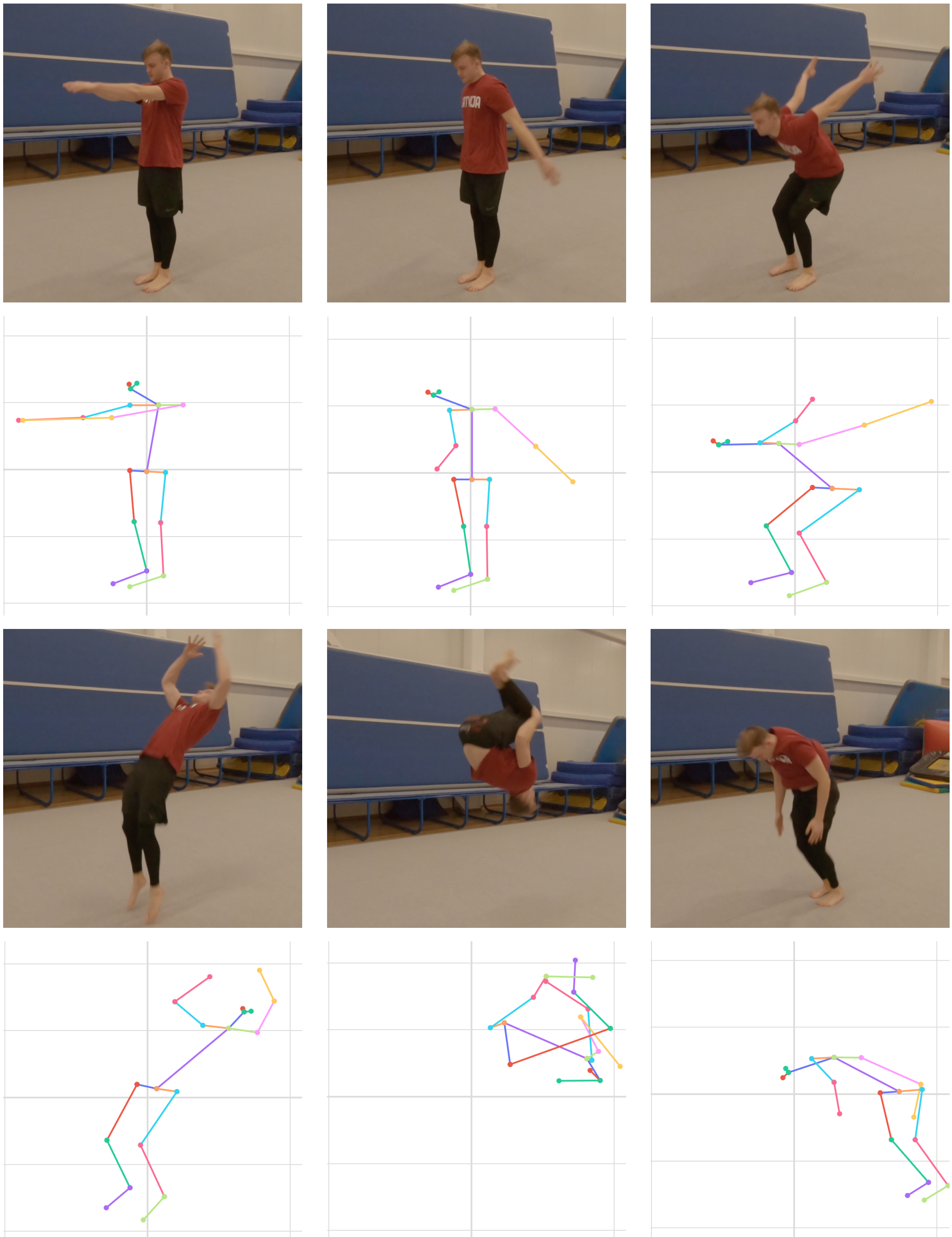

Figure 4.6: Backflip sample centered to coordinate system origin

4.3 Results

The final dataset yielded after applying all pre-processing strategies consists of 51 backflip and 62 back handspring samples, totaling to 113 samples.

Chapter 5

Classifier Training

As demonstrated in a recent research by Farzan Majeed Noori, Benedikte Wallace, Md. Zia Uddin and Jim Torresen in [12], a combined architecture of using OpenPose for pose estimation and Recurrent Neural Networks (RNN) for human activity recognition could prove as the new state-of-the-art solution for robust and non-intrusive human activity recognition. In the cited research, OpenPose was used to extract keypoints from a subset of the Berkeley Multimodal Human Action Database (BMHAD) [13]. What makes it remarkable, is the classification accuracy obtained using a Recurrent Neural Network with Long Short-term Memory (LSTM) cells over more conventional machine learning classifiers, such as Support Vector Machines, Decision Trees and Random Forests. Since the dataset contained human activities recorded from different angles, the solution is also view-invariant, which also increases the robustness of this solution. Another paper, a technical report by Chinmay Sawant [14], also supports the combination of OpenPose with LSTMs for time-series classification, reporting similar accuracy results in real-time application on the same BMHAD dataset, making use of the efficiency of this solution.

The difference of the cited work and current work is the custom dataset constructed for gymnastics based movements. In contrast to the BMHAD dataset, which includes 11 general activities, such as jumping jacks, waving and clapping, the dataset used in the current thesis uses more complex biomechanical activities, including human body rotations and airtime, such as backflips and back-handsprings. The author of this paper expects RNNs used for general action recognition to also be applicable to gymnastics movements. The goal of this research is to validate this hypothesis and investigate neural networks most suitable for gymnastics action recognition.

5.1 Methodology

To develop neural networks capable of recognizing gymnastics movements, a combination of empirical knowledge with the mathematical theory of artificial neural networks is used as the starting point. The development process starts by training a simple recurrent neural network with gymnastics data and analyzing the results. Simple RNN is chosen primarily for its known property of being able to represent information from context window [15]. RNNs have *long-term memory* in the form of weights, enabling the network to *remember* a gymnastics element represented as a sequence. An iterative process is used to train, validate, analyze and compare the results of each classifier. The steps followed in the current thesis are represented in figure 5.1.

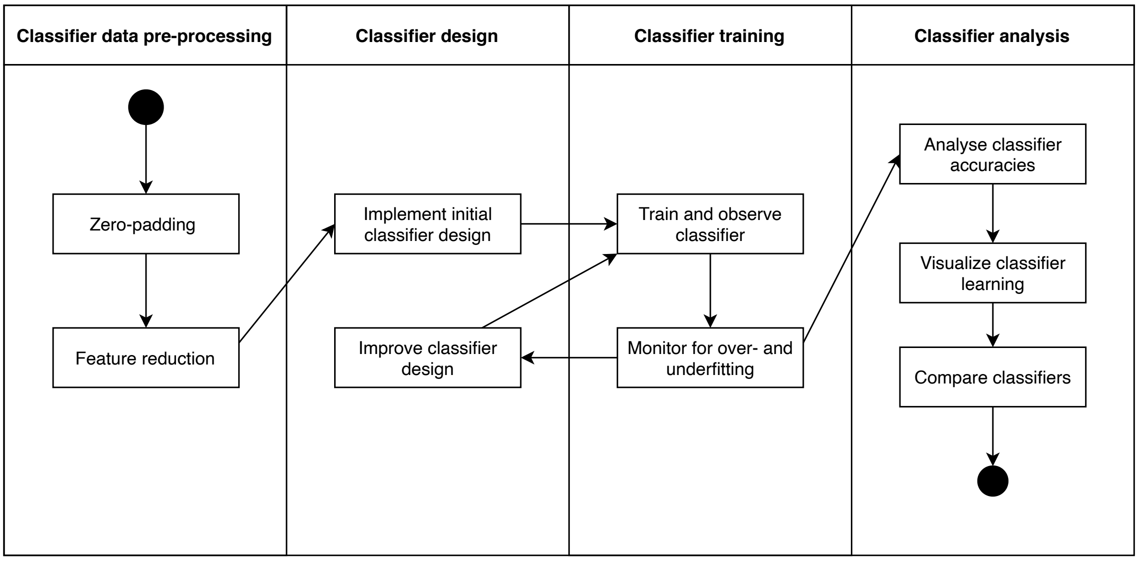

Figure 5.1: Tasks completed regarding classifier development in the thesis

5.1.1 Classifier Prerequisites

Two prerequisites before feeding training data to the classifiers include:

- *Zero-padding* — Gymnastics activities performed vary in their length and in order to keep the classifier dimensions static, the samples are zero-padded. Zero-padding is done by finding the longest sample performed and increasing the timesteps of other samples with zero values until they are equal in length to the longest sample.

- *Feature reduction* — Recurrent neural networks are prone to overfitting and with the combination of the author's domain knowledge about gymnastics, the 25 total keypoints obtained during pose estimation are reduced to 15, filtering out less prominent keypoints necessary to recognize gymnastics movements. The 15 keypoints used for classification include the trunk, head and limbs of the skeleton.

5.1.2 Classifier Algorithms

Classifier algorithms experimented with in the thesis are:

- *Simple RNN* — The first classifier (shown in figure 5.2) experimented with is a network model with one simple recurrent layer. The recurrent layer consists of 2 units, chosen accordingly to the amount of outputs the network produces - the movements is either a backflip or a back handspring. A dropout layer with a 0.5 rate to prevent overfitting is added next. An activation layer with rectified linear units follows with a flattening layer to reduce dimensions and finally an output layer with softmax activation is used for representing the different activities. Categorical cross entropy is used as the loss function during stochastic gradient descent with an Adam optimizer with a learning rate of 0.001.
- *Hierarchical RNN* — The Simple RNN works well for the short duration activities analyzed in this paper, but the downside of using one recurrent layer is observability. By feeding all available dimensions into one recurrent layer, we lose the visibility of what exactly the different units in a neural network learn. A more advanced and deeper neural network (inspired by article [11]) consists of several recurrent layers, each representing some logical unit. The model used for experimentation in this paper is shown in figure 5.3. Hierarchical RNN uses multiple input units, each representing a human skeleton subsection. In this example, the skeleton is divided into left arm, right arm, trunk, left leg and right leg subsections. In subsequent layers, the recurrent units are fused together and finally neural activations are applied on a fully connected skeleton layer. One of the advantages of this method is that we can now intercept some layer of interest and observe the neuron activations only

regarding a subsection of the skeleton.

- *LSTM* — An RNN with LSTM cells was also implemented. The simple recurrent layer is substituted with a hidden LSTM layer with 2 hidden units accordingly representing the outputs of the network. Other layers remain identical with the *Simple RNN* classifier.

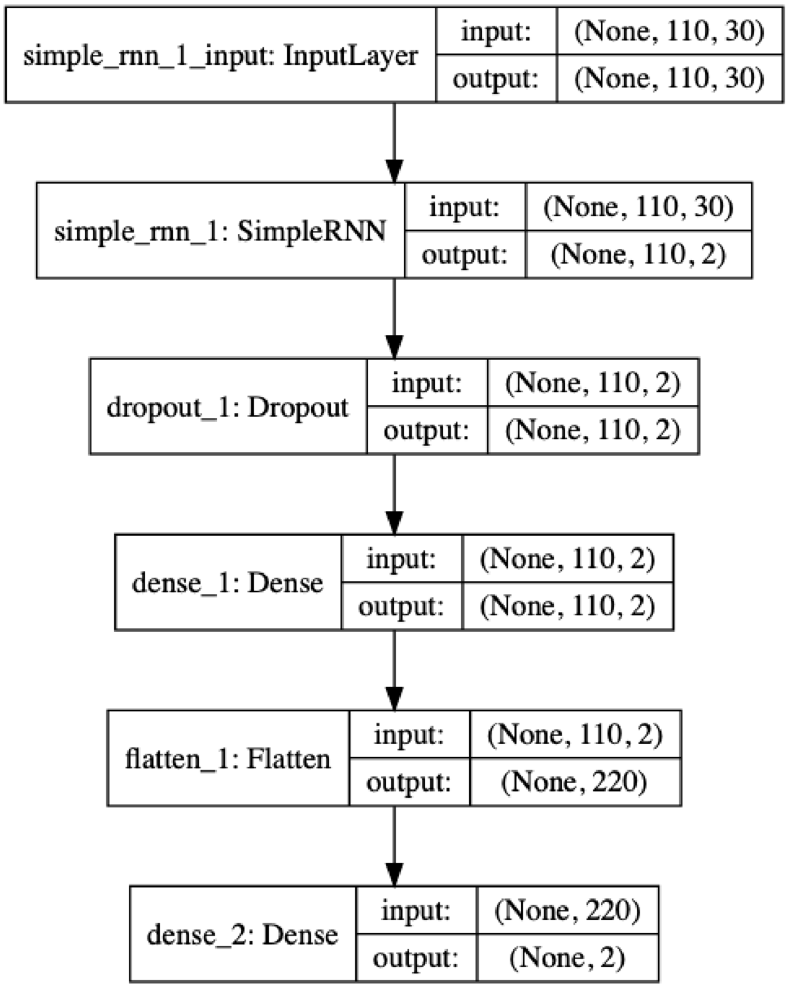

Figure 5.2: The neural network design with a simple recurrent layer

5.1.3 Classifier Validation

Validating the correctness of our RNN training requires the usage of some common classifier validation techniques. These techniques are commonly used while training neural networks to avoid overfitting.

Firstly, the sample data is split by 0.2 rate between training and testing data in order to test for unbiased results when the model training has stopped. A validation

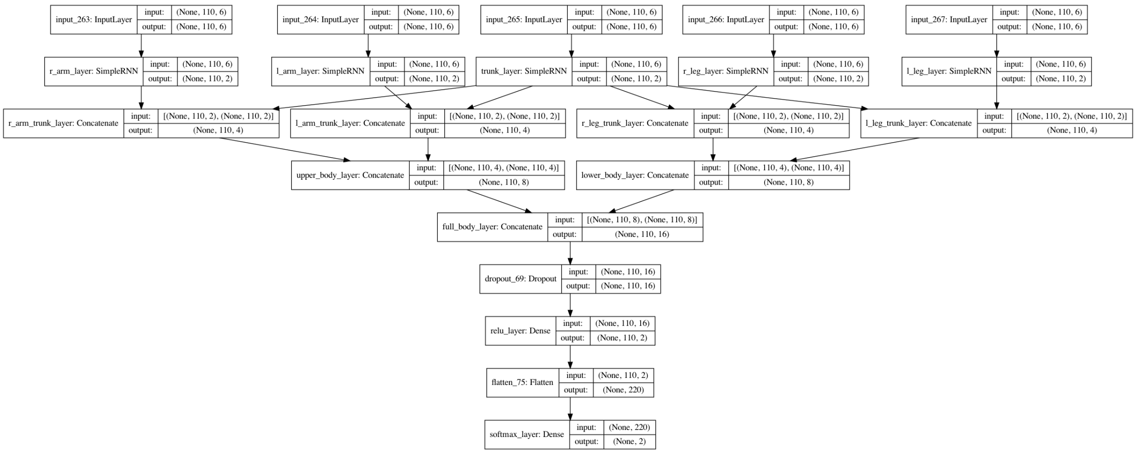

Figure 5.3: The neural network design with hierarchical recurrent layers

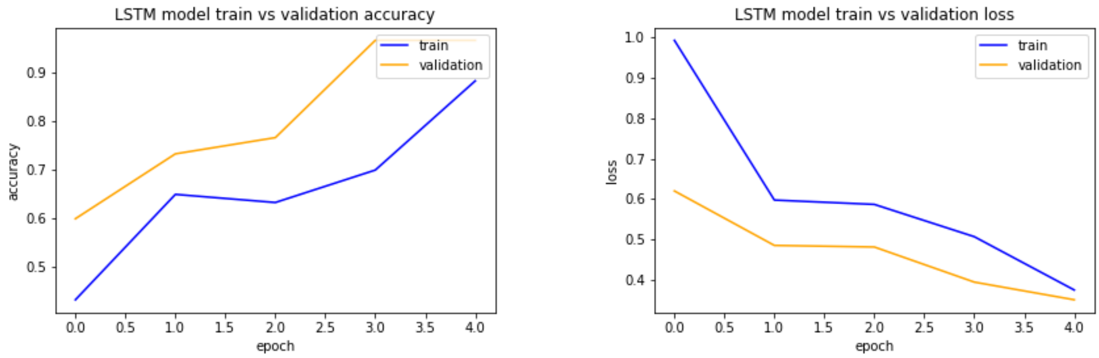

Figure 5.4: Monitoring for model accuracy and loss to detect overfitting/underfitting

split of 0.33 rate is also used while training the RNN to observe the loss and accuracy of the model, essential to help with hyperparameter tuning. Validation set is kept separately from test set so when we optimize our hyperparameters, we can validate our results against the test set. The figure 5.4 demonstrates how the LSTM model accuracy and loss are monitored during epochs in order to detect overfitting or underfitting. When the training and test set accuracies and losses do not diverge too much, we can be more confident that our classifier is generalizing and not just optimizing for the input data [16].

Lastly, the experiment is repeated on 5 unconnected models. Neural networks sometimes tend to converge to some minima that is not the most optimal (global). By running our experiment on 5 unconnected models, we avoid basing our observations and analysis under the influence of one sub-optimal (local) minima.

Repeat	Accuracy (%)
1	86.957
2	91.304
3	73.913
4	91.304
5	100.0

Figure 5.5: Mean accuracies obtained during each Simple RNN classifier run

5.1.4 Classifier Training Process

Simple RNN

Due to the small sample data size, the RNN classifier is trained for 5 epochs with stochastic gradient descent. The hyperparameters are empirically selected for the classifier, with the goal to maximize the accuracy while preventing overfitting.

An early stopping hook is also implemented to monitor for validation loss improvements. No improvements in validation loss commonly indicate that the network has stopped learning and is starting to overfit to the training data. After 3 epochs with no improvements the hook stops the learning process of the neural network.

5.2 Classifier Visualization and Analysis

Simple RNN

The fourth experiment run (from table 5.5) is chosen to dive deeper into the inner workings of the classifier.

By configuring the model's RNN layer to return hidden states, it is possible to intercept the output values of this layer and investigate the neuron activations. It is also possible to visualize these activations to better understand what the neurons are learning. The author compares neuron activations of backflips (figure 5.6) and back handsprings (figure 5.7). In both figures, the activations are visualized using four horizontal lines. Each horizontal line also consists of three inner lines, where the first represents timestep indices and other two represent activations of two hidden layer neurons. The outer horizontal lines are split by 30 timesteps, which correspond to 0.5 seconds of activity time since the activities are recorded at 60 frames per second.

All samples are 110 timesteps long and the missing values are zero-padded. The chosen samples are also correctly predicted by the model. It is interesting to observe how the first neuron starts to flicker it's activation when it encounters the ending of an activity. From here we can also observe how back handsprings tend to last longer than backflips, easily noticeable from neuron activations. A second observation can be made by noticing that the first neuron seems to activate throughout almost the whole duration of a backflip.

TODO: Add reference to visualization source

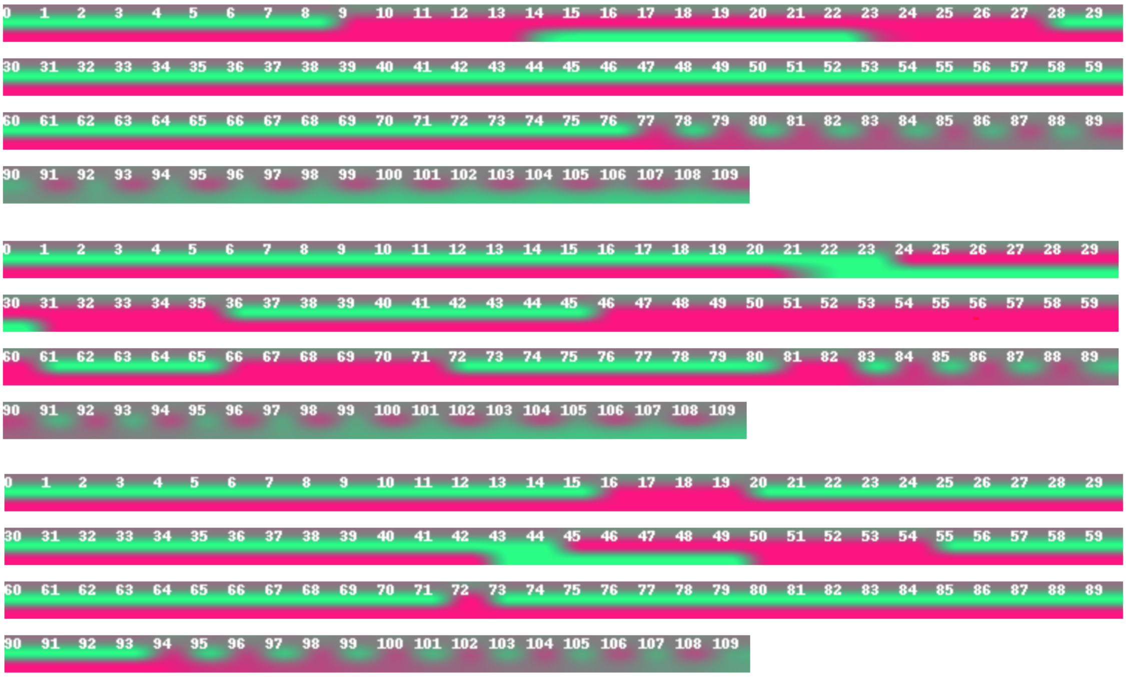

Figure 5.6: Simple RNN neuron activations for backflip samples

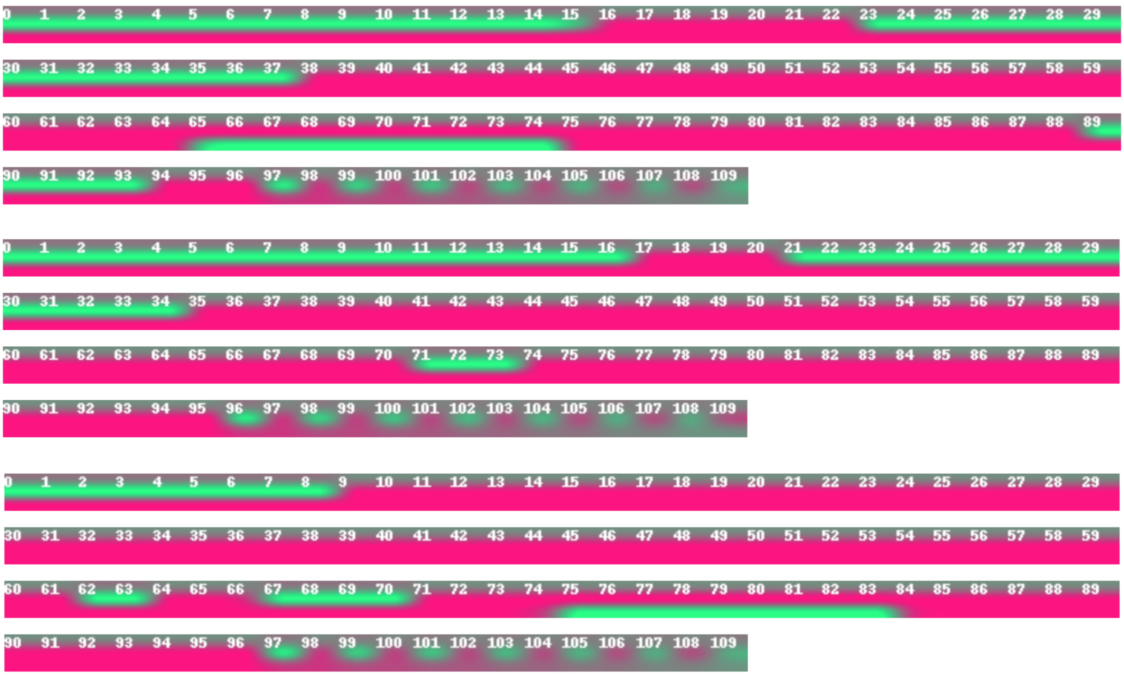

Figure 5.7: Simple RNN neuron activations for back handspring samples

Chapter 6

Explaining Predictions

6.1 Motivation

6.2 Methodology

6.3 Experimental Result

Chapter 7

Discussion

Chapter 8

Conclusion

Acknowledgments

Bibliography

[1] Robyn Burgess and Guillermo J. Noffal. Kinematic analysis of the back salto take-off in a tumbling series: Advanced vs. beginner techniques. 2001.

[2] Michalis Vrigkas, Christophoros Nikou, and Ioannis A. Kakadiaris. A review of human activity recognition methods. *Frontiers in Robotics and AI*, 2:28, 2015. ISSN 2296-9144. doi: 10.3389/frobt.2015.00028. URL `https://www.frontiersin.org/article/10.3389/frobt.2015.00028`.

[3] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. *CoRR*, abs/1611.08050, 2016. URL `http://arxiv.org/abs/1611.08050`.

[4] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *CoRR*, abs/1812.08008, 2018. URL `http://arxiv.org/abs/1812.08008`.

[5] Gabriella Penitente, Franco Merni, and William Sands. Kinematic analysis of the centre of mass in the back handspring: A case study. *Gym Coach*, 4:1–11, 01 2011.

[6] Tomas Pfister, James Charles, and Andrew Zisserman. Flowing convnets for human pose estimation in videos. 06 2015. doi: 10.1109/ICCV.2015.222.

[7] Posenet. `https://github.com/tensorflow/tfjs-models/tree/master/posenet`, 2020.

[8]

[9] Openpose requirements and dependencies. `https://github.com/CMU-Perceptual-Computing-` 2020.

[10] Openpose python pose extraction module. `https://github.com/AllarVi/openpose-extract`, 2020.

[11] Yong Du, W. Wang, and L. Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1110–1118, 2015.

[12] Farzan Majeed Noori, Benedikte Wallace, Md. Zia Uddin, and Jim Torresen. A robust human activity recognition approach using openpose, motion features, and deep recurrent neural network. In Michael Felsberg, Per-Erik Forssén, Ida-Maria Sintorn, and Jonas Unger, editors, *Image Analysis*, pages 299–310, Cham, 2019. Springer International Publishing. ISBN 978-3-030-20205-7.

[13] G. Kurillo R. Vidal F. Ofli, R. Chaudhry and R. Bajcsy. Berkeley mhad: A comprehensive multimodal human action database. In *IEEE Workshop on Applications on Computer Vision (WACV)*, 2013.

[14] Chinmay Sawant. Human activity recognition with openpose and long short-term memory on real time images. Technical report, EasyChair, 2020.

[15] Zachary Chase Lipton. A critical review of recurrent neural networks for sequence learning. *CoRR*, abs/1506.00019, 2015. URL `http://arxiv.org/abs/1506.00019`.

[16] Douwe Osinga. *Deep Learning Cookbook*. OŔeilly Media, Inc., 2018. `https://www.oreilly.com/library/view/deep-learning-cookbook/9781491995839`.