Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

# Human Action Recognition - A Novel Approach to Count Repetitive Actions

Arun Jayaprakash

August 19, 2022

Supervisor: Inmaculada Arnedillo-Sanchez

A dissertation submitted in partial fulfilment
of the requirements for the degree of
MSc Computer Science

# Declaration

I hereby declare that this dissertation is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at `http://www.tcd.ie/calendar`.

I have also completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at `http://tcd-ie.libguides.com/plagiarism/ready-steady-write`.

Signed: Arun Jayaprakash Date: 19/08/2022

# Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Signed: Arun Jayaprakash Date: August 19, 2022

# Human Action Recognition
# A Novel Approach to Count Repetitive Actions

Arun Jayaprakash, Master of Science in Computer Science

University of Dublin, Trinity College, 2022

Supervisor: Dr. Inmaculada Arnedillo-Sanchez

Gross Motor Action Recognition in children is a nuanced field of research with many complexities. While general human action recognition has been well researched with a plethora of off-the-shelf solutions for pose and activity detection, there has been comparatively much less research aimed towards child action recognition. Such a system aimed towards children has a large number of applications in assessing growth of children. Identifying and measuring such activities can be used as an indicator of gross motor skills development in children. Gross motor skills are closely related to cognitive development and thus such a system can be used to determine the onset of growth issues in children. By detecting and identifying such issues at an early stage, it becomes possible to take corrective actions accordingly. This project aims to create a system to identify and count 4 repetitive actions of squatting, jogging on a spot, walking on a line and running and coming to a stop on an in-house dataset of children. The project proposes a single-frame classification approach for action recognition in children. By breaking down videos into frames and assigning individual labels for each frame specifying an intermediate state for each action, a new and more interpretable approach to action recognition is proposed that does not rely on complex black-box neural networks. The efficiency of the proposed model was evaluated and possible improvements and drawbacks are identified. This dissertation also highlights the key issues that were faced while working with data pertaining to children.

# Acknowledgements

A number of people were crucial in the completion of this body work and I would like to sincerely thank everyone involved.

I thank my parents, Dr. V.L Jayaprakash and Dr. Santhi K.S for always supporting my endeavours. Their constant support and advice was vital for this body of work to come to fruition. I also thank my brother Dr. Arjun Jayaprakash for his support and feedback.

I sincerely thank my supervisor, Dr. Inmaculada Arnedillo-Sanchez for guiding me in the right direction throughout the entire course of this dissertation.

I would also like to thank my friends Manu and Rohan for proofreading the thesis and providing insights from different perspectives.

ARUN JAYAPRAKASH

University of Dublin, Trinity College August 2021

# Contents

# List of Figures

# List of Tables

# 1 Introduction

## 1.1 Motivation

Every child matures through different stages as part of their growth. Existing studies such as Jean Piaget's 4 stages of cognitive development state that each child acquires a particular set of skills during a particular stage. These skills encompass both motor skills as well as the cognitive skills necessary to process and understand information.

As every child is expected to acquire a similar set of skills around the same stage of their growth cycle, a child that does not possess a particular skill or is struggling to learn a skill may be facing growth issues. Thus there exists a strong link between motor skill activities and the cognitive development of children [1]. Determining the onset of such cognitive or motor-skill related growth issues at an early stage itself is imperative in treating such developmental problems. Thus, leveraging technology to monitor and assess the growth and development of children may prove to be highly beneficial in detecting onset of growth issues or difficulties in learning. However such technologies aimed towards children, remain a largely unexplored avenue.

## 1.2 Problem Definition

This paper is aimed towards studying 4 actions that are repetitive in nature from an in-house dataset - squatting, jogging on a spot, walking on a line and running and coming to a stop. The primary objective is to identify the action performed by a child from a video input. And secondarily, to count the number of actions performed. Such a system would have a large number of applications in monitoring growth of children. Extensive research has already been carried out in regards to pose estimation and action recognition. A majority of such systems rely on neural network models such as Long Short-Term Memory and other Convolutional Neural Network or Recurrent Neural Network models and approach human action recognition as a time-series problem to identify activities. The dataset provided contains labels for the activities performed, but does not include any annotations regarding the number of actions performed nor the timestamps at which

certain actions are done. Since this information is not available, pre-existing solutions are inadequate to solve the problem to be tackled. As a result, a new approach that breaks down human action recognition into a classification problem at individual frame level is proposed. The system identifies intermediate states for each action using heuristics and assigns these as labels for each frame. Simple machine learning algorithms are then used to solve the classification problem. And finally by monitoring the predictions of the ML models, the counts of actions is obtained.

The pinnacle of this work would be to create a monitoring system that can be deployed in today's world which is capable of identifying developmental issues in children. However, this paper aims to identify the feasibility and challenges when working towards creating such a system, by focusing on the 4 aforementioned actions.

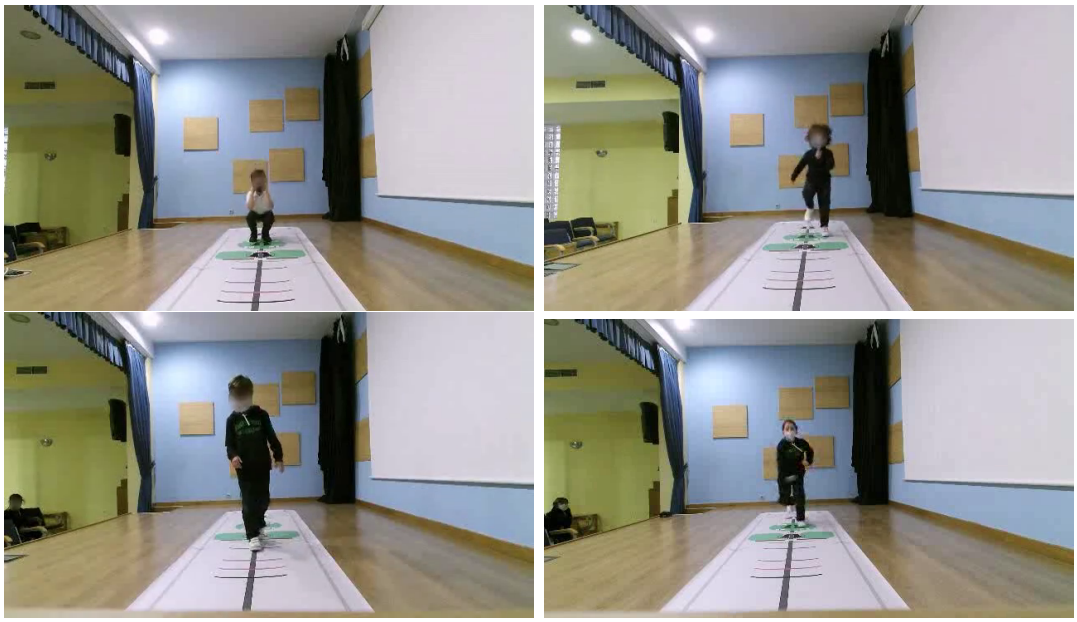


Figure 1.1: Actions - Squat, Jog on a Spot, Walk on a Line, Run and Stop

## 1.3 Dissertation structure

The next chapter of the dissertation covers the related research carried out in the field of human action recognition. The chapter outlines general human action recognition methods from recent times and also covers research done regarding child action recognition in particular. The best performing State-Of-The-Art models for human action recognition are also discussed in Chapter 2. The chapter also covers some of the common challenges faced during action recognition in general. Chapter 3 goes in-depth into the details of the proposed pipeline and all the steps taken to create the system such as feature extraction, normalization, intermediate heuristic based labelling, machine learning algorithms and the final buffer approach to obtain individual action counts. Chapter 4 details the exper-

imental setup used to test the system created and also analyzes the results to understand shortcomings and strengths. The final chapter, Chapter 5 concludes the dissertation and also states possible directions to continue the research forward by recognizing drawbacks and potential improvements.

# 2 Related Works

Extensive amounts of research have already been carried out with regards to Human Action Recognition (HAR) due to the very wide range of applications of HAR. Due to numerous factors such as background variations, lighting, behavioural variations, variations in clothing and occlusion issues, HAR is often considered to be a challenging and difficult problem to solve with accuracy in real-world use cases. This section outlines various approaches towards HAR that have been explored in the past and tries to identify approaches that align with the goals of the project. The section also delves into recent research regarding child action recognition and outlines general challenges in HAR. The best State-Of-The-Art models are also explored to choose a suitable fit for the dissertation.

## 2.1 General HAR Approaches

HAR is generally considered to be a time-series problem, due to the temporal relation between different frames and the vast majority of the currently existing state-of-the art models consider input features as time-series data. The time-series approach towards HAR problems may be attributed to the influential work by Bobock Et Al. [2]. The paper proposed concepts of Motion History Image (MHI) and Motion Energy Image (MEI), where the MEI template defines a binary image to identify where motion occurs in the video and the MHI template contains historical temporal information at individual pixel level. Such approaches based on a global representation of the human body and its attributes are termed Holistic approaches. However, holistic based approaches became less relevant over time as these approaches were less robust to common HAR constraints such the camera view point and occlusion. As a result focus shifted towards Local Representation Methods which aim to extract local features such as edges and key points of interest such as joints. These features are then aggregated and tracked over time. A large amount of research has been carried out to properly identify points of interest and to define local descriptors for the detected points [3].

Multiple different taxonomies are applicable to HAR methods since various new ap-

proaches are constantly being explored in the field. Existing approaches that utilize time-series data may be broadly grouped into 4 categories - Statistical, Shallow Machine Learning, Ensemble and Deep Learning methods. Out of these approaches, Deep Learning solutions are most commonly used since they are efficient in dealing with time-series data. A majority of these approaches use Recurrent Neural Networks and Long Short-Term Memory Networks to identify human activities [4]. An alternate, more broader subdivision of action recognition groups approaches into 2 categories - the aforementioned deep learning methods and handcrafted feature methods such as the holistic approaches mentioned earlier [3]. More recent deep learning methods are further subdivided into different groups based on the networks used and the type of data stream uses, whether its spatial or temporal. Deep learning approaches may be generally grouped into 2D CNN methods, RNN methods, 3D single-stream or multi-stream approaches and convolution-free methods [5]. The above taxonomies are vision-based HAR approaches which use video data for action recognition. However the development of motion sensors such as Kinect and improved sensors in wearable devices and smartphones have also paved the way for high quality sensor motion data which are used as input in sensor-based HAR techniques. However, these approaches are not explored further in section since this body of work is a vision-based HAR approach.

The advent of deep neural networks have introduced a number of efficient deep learning based solutions for HAR - The efficacy of 3D convolutional networks (CNN) which use features from temporal domain in addition to the standard spatial features used in traditional 2D CNNs, have been proven by Ji Et Al. [6] in HAR tasks. But such networks require fixed input-size and defining a fixed input size along the temporal domain proved to be a challenge and many techniques involving fusion and pooling were explored. Further research of combining 3D CNNs and Recurrent Neural Networks was shown to provide good results for HAR on public datasets such as the KTH dataset and this approach for HAR gained widespread popularity [6]. A different deep learning approach for HAR involved creating 2 separate networks where one network learns spatial information from frame data while a second network learns temporal information from optical flow data [7], where ptical flow refers to the motion of individual pixels from one frame to the next, and can be used to understand the motion of a body in a video

Recurrent Neural Networks and Long Short-Term Memory Networks have a proven track record for identifying short-term actions while Convolutional Neural Networks generally perform better when dealing with long-term actions that are repetitive in nature [8]. However, both RNNs and LSTMs, as well as other time-series based methods require fixed input size where the input is a fixed number of frames from the video. This sliding window of features which correspond to a few seconds or less from the video is thus a key factor in creating an accurate time-series model. If the length of the window is chosen

to be too low, activities may inadvertently get terminated before completion and similarly too small values may terminate the window before completion [9]. Banos et al. extensively study the effects of window size in HAR and find that a window size of 1-2 seconds is ideal for most activities, including longer ones [10]. However, the dataset used in the study is that of adults [11], and does not contain the large amounts of variance that are seen in child behaviour. The dataset considered for the project contains videos of varying lengths, and within each video different children start performing actions at different points of time. In addition to this, different children perform different actions with varying duration - for example a younger child would hold a squat position for a much smaller duration than an older child. Because of this difference in lengths, LSTMs and RNNs were difficult to apply on the data. Also, it is not a straightforward to obtain the count of actions performed using a time-series approach without having annotations for the exact timestamp at which an action is performed in the video.

## 2.2 Child Action Recognition Approaches

Many of the challenges associated with the dissertation are due to the innate nature of variation in child behaviour. Unlike adults, children have not developed the cognitive skills necessary to perform certain activities accurately and as a result there is a large amount of variability in the dataset. To understand, how such variability can be tackled, existing research regarding child activity recognition was explored.

Pandey et al. propose a new method for detecting different actions on children with autism. To tackle the problem of insufficient amounts of annotated training data, the paper proposes a method that re-trains a classifier trained on the in-house autism dataset with samples from a larger, publicly available dataset that are semantically similar in optical flow. However, most of the actions that are considered are limited to primitive actions which involve basic motion of one or two body parts [12].

A majority of existing research regarding child action recognition are sensor-based methods, which utilize multiple sensor data to identify precise motion and subsequently identify/label actions [13,14]. Such non-visual approaches have very high accuracy owing to the quality and precision of the input data obtained from the sensors. However, such approaches are inapplicable in the scenario since the dataset consists of videos only without any additional sensor information.

The most similar body of work on Gross Motor Action Recognition (GMAR) in children was done by Suzuki Et Al. in 2020 [15]. In this paper, the authors propose a fully automated AI system for GMAR in children to identify growth disabilities. The authors use OpenPose pose estimation algorithm to obtain the keypoints for the children from

the video. However the keypoint detection were prone to errors such as missing keypoints and detecting the wrong person from the video. The authors use a particle filtering approach using the the neck as a representative landmark of the detected body to correct the aforemtnioned errors. The authors then use a CNN which was fed 8 frames mapped to a single activity such as running, horizontal jump etc. [16]. In the previous work by the authors on GMAR, an LSTM based approach was tested, but a CNN was opted in place of LSTMs in the new approach as it was found to be difficult to tune the hyperparameters from the LSTM and improve performance. Zhang Et Al. explore a similar approach of extracting keypoints from OpenPose and using LSTM models to create an action recognition for children with autism [17].

Olalere Et Al. [18] investigate the efficacy of existing SOTA deep learning models with respect to child action recognition. By comparing the performance of modern SOTA algorithms on adult and child datasets pertaining to sports activity, it was observed that modern day algorithms are capable of handling HAR for children. Since most SOTA models are trained on predominantly adult data, it was expected that the performance would be skewed towards adults, however this was not observed to be the case. However, the investigation finds that child action recognition, especially for complicated activities such as sports, is a much more complicated task due to high intra-class variance exhibited by children.

## 2.3 State-Of-The-Art Frameworks

### 2.3.1 OpenPose

OpenPose is one of the most widely used libraries for 2D and 3D keypoint detection. The open-source library works on real-time data and is capable of detecting multiple skeletons from a video input. The library gained traction for its high accuracy on videos with large number of people. In addition to the multi-person pose estimation, it also provides solutions for detecting keypoints from face, foot and hands as well.

OpenPose utilizes a bottom-up approach to build a non-parametric approach termed Parametric Affinity Fields to add information regarding individual limbs over the image. The model uses CNNs to create features from individual frames. Pre-existing CNN model VGG-19 is used to extract feature maps on which simultaneous body part detection and association is performed. Part Affinity Fields are then used to assemble individual parts to form the skeletal structure where the PAF is a 2D vector associated to individual limbs and encodes information pertaining to the direction of one limb to another. OpenPose has a proven accuracy and inference time on most public HAR datasets [19].

### 2.3.2 HRNet

High-Resolution Net is a Convolutional Neural Network that was popular for using high resolution representations of features throughout the learning process. While most existing algorithms learn using low resolution representations and then recover output resolution using low to high conversions, the HRNet model maintains the high-level image representations and is able to produce highly precise spatial predictions. This also results in higher accuracy for pose estimation. The neural network consists of a number of sequential as well as parallel networks of varying resolutions. The network also has higher test scores when compared to networks such as OpenPose on public datasets such as COCO and MPII [20].

### 2.3.3 MediaPipe

MediaPipe is an open-source machine learning framework that provides a plethora of machine learning solutions such as pose estimation, face detection, object detection, box detection, iris detection and selfie segmentation to name a few. It is a cross-platform and lightweight framework which makes it ideal for a wide range of applications. Various machine learning models under MediaPipe are currently integrated into various Google products such as Google Lens, Google Photos and NestCam [21]. The library is written in C++ and uses graph pipelines to ensure high inference speeds. However, MediaPipe does not attain truly real-time inference speeds and the performance is dependent on underlying hardware and quality of video input. MediaPipe's pose landmark solution utilizes BlazePose GHUM 3D model and this solution was used for keypoint detection in the project owing to its efficacy on fitness related activities. This model is discussed in detail in Methodology Section 3.2.

## 2.4 Challenges in HAR

Even SOTA models in HAR are prone to misclassifications on real-world data due to a number of reasons [22]:

- Background clutter - In real-world scenarios, the background of the subject of interest can have highly varying levels of noise. Outdoor scenarios with multiple background elements adversely affect many SOTA models. For the in-house dataset considered for this research, all videos are indoor with only a few constant backgrounds, which makes object detection easier. However, several videos were identified which had adults in motion in the background which could cause issues for SOTA models.
- Variation - HAR variations are generally classified as Intra-Class and Inter-Class

Variations. The former refers to variations within a single class/action owing to the fact that different people perform actions differently. Considering the example of Jog on a spot action, the pace of the jog is open to interpretation and hence different subjects will jog at varying speeds. This issue of intra-class variation is even more severe when dealing with dataset of children as children tend to have much higher variance with respect to activities when compared to adults. Inter-class variations refer to similarities between different actions such as jogging and running which could lead to misclassifications. Since the four actions considered for this research are different form each other, inter-class similarity is of less importance in the context of this project. However, when additional similar actions may need to be considered in the future, intra-class variability could become more of a challenge.

- Lack of labelled data - Modern day deep networks typically require large amounts of labelled data. Such well accurately annotated data is difficult to create with regards to HAR. Many of the larger public datasets such as YouTube-8M provide large amounts of data, but the annotations cannot be guaranteed to be accurate. The issue of annotations is present in the dataset considered for the research as well - although individual videos have been labelled as belonging to a particular class/action, the counts of the action performed are not provided. Manually annotating over 2000 videos to find activity count was also not feasible over the duration of the study.

- Predicting discriminative frames - For any video pertaining to an activity, not all frames from the video will be relevant or required for HAR. Identifying such key frames and removing redundant frames is difficult due to differences in video durations and differences in when key activities begin in videos.

- Occlusions - Occlusions in tracking occur when the spatial state of an object to be tracked is present in a video, while the key features used to detect the object are not available. Self-occlusion occurs when one part of the tracked object's body occludes another, while inter-object occlusion refers to the case when two objects occlude one another. Occlusions may also be partial or full based on whether only a segment of the tracked object or the entire object itself is occluded respectively [23]. For any proper human action recognition model, it is necessary to include occluded images in the training set. Angelini Et. Al [24] account for occlusion in images by deleting key points using a random distribution to simulate occlusions while developing a HAR model. However most research regarding occlusion handling is aimed towards inter-object occlusion and not self-occlusion. Methods of handling self-occlusion such as those used by Huang et al. utilzie multiple cameras to correct occluded landmark points [25].

# 3 Methodology

## 3.1 Overview

A six-stage approach is proposed to identify an activity and enumerate the number of actions successfully completed in the video. The proposed pipeline involves extracting key points from the video using existing SOTA framework MediaPipe. The extracted key points are normalized to improve stability and then the problem is converted to a classification problem at individual frame level by assigning intermediate state labels using different heuristics for each action. Finally various machine learning models are utilized to solve the classification problem and finally a buffer to count transitions between predicted frame labels is used to obtain the total count of actions from the video. The overall pipeline is pictured in Figure 3.1.

## 3.2 Data

The data provided consists of approximately 500 videos of children performing each of the 4 actions jog spot, squat, walk on a line and run and stop. Since the data consists of video footage of minors, the data is highly sensitive. As a result, protecting the identities of the children and ensuring that the system is designed in a secure and ethical manner that is foolproof was of the utmost importance. Prior to data handover, the videos were completely anonymized by blurring faces of all the children. In addition, the children also wore masks in the video which further removed distinguishable facial features.

Each subject was given 3 attempts for each action, where Level 0, 1 and 2 videos correspond to the first, second and third attempts respectively. Thus, in general Level 2 videos are the videos in which the children performed the actions more accurately as they learned from practice and grew more accustomed to the nuances of each activity. Level 0 and some Level 1 videos were cases where more mistakes were made in performing the action and these videos would thus form the trunk of the incorrect activities that are crucial in creating a proper machine learning model that can efficiently identify wrongly performed activities.

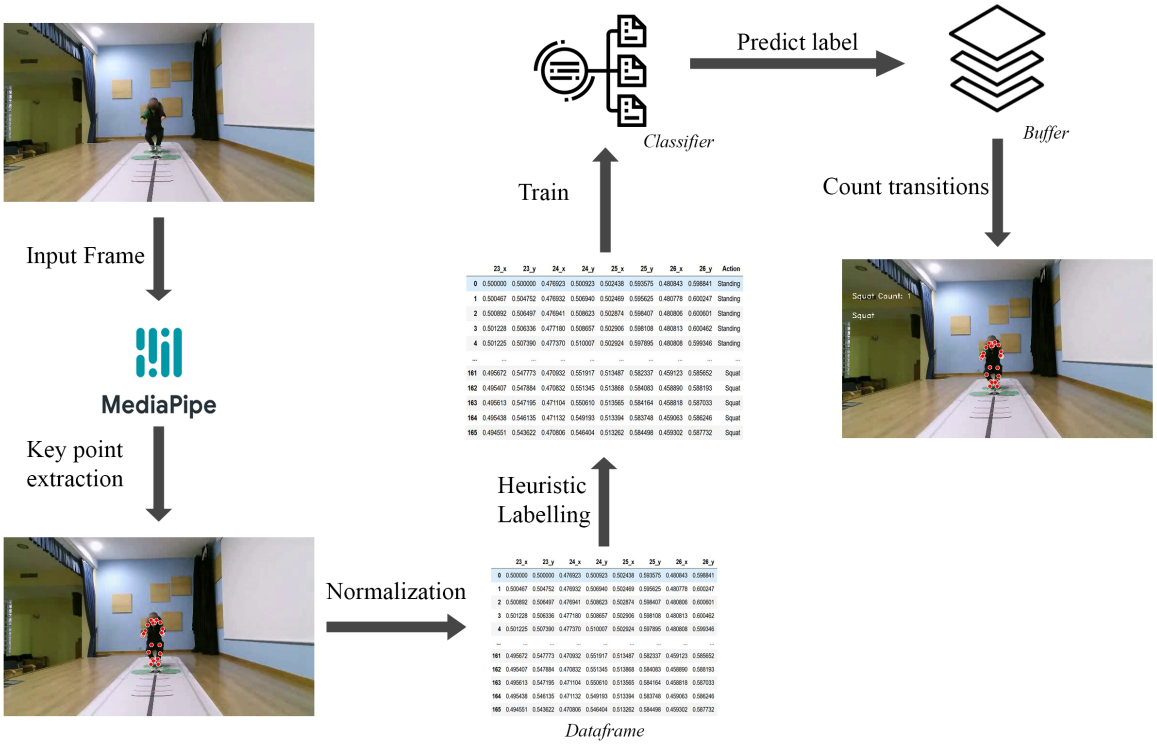



Figure 3.1: Pipeline

The dataset provided was a challenging and complicated dataset due to a number of reasons. The primary reason was the very large variations in how different children performed each activity - especially younger children performed different activities in vastly different manners resulting in large intra-class variability. Since many videos were unusable because of this issue, such videos had to be removed from the training dataset. There were also videos particularly associated with younger children in which no action was performed. However, identifying such problematic videos and manually filtering these out was an unattainable task on a dataset of almost 2000 videos in total considering the duration of the project. Another challenge with the data is the variations in the length of videos - in each video the start and end timestamps of the action performed varied significantly. As a result, identifying the exact start and end state of actions proved to be difficult. Hence the videos were not trimmed and the full videos were used for further processing.

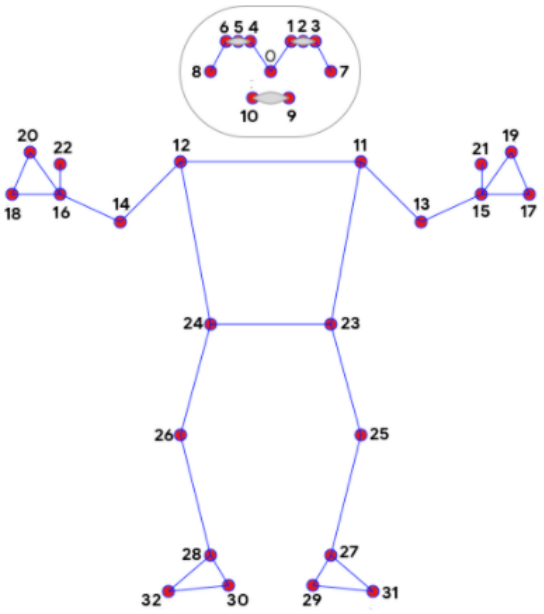
## 3.3 Key Point Extraction

The key features required to identify different actions are the x and y co-ordinates of the various joints of the children from the video feed. MediaPipe's Pose solution was used to extract the required joint co-ordinates. The Pose estimation solution provided by MediaPipe uses the BlazePose GHUM model to find key points.

### 3.3.1 BlazePose - MediaPipe

BlazePose is a lightweight Convolutional Neural Network model packaged under MediaPipe that is geared towards fitness applications [26]. The model returns 33 keypoints of a human body such as the endpoints of the eyes, nose, and various different joints. Such a large number of key points ensures that a high degree of fidelity is obtained during inference. The model is trained on 2 datasets, out of which the first dataset consists of everyday poses, while the second dataset consists of various fitness and yoga poses. Thus the model is ideal for fitness related activities. The BlazePose model utilizes a two-step pipeline in which region of interest containing the primary person is identified, and the region-of-interest cropped input is fed to a model which identifies 33 key points. The model uses 2 networks - a body pose detector which first detects the presence of a human in a frame and a pose tracker network which detects the 33 key points and also identifies region-of-interest. If the tracker network does not find a human in the current frame, the body pose detector is again invoked on the next frame. The person detector model used relies on a face detection as it was found that detecting the face is the best method for a neural network to learn the presence of a human [26]. To account for occlusion, the dataset used for training the model was modified with simulations for occlusion by adding coloured rectangles to cover different parts of the body. By adding such occluded data to the training set, the model is capable of detecting keypoints that lie outside the frame of view of the camera as long as the person detection model works.

The Python API for Pose Estimation was used to identify the key points for the dataset. The Pose API returns x and y-ordinates with unit normalization on image dimensions, the z co-ordinates indicating the depth of each detected point, as well as visibility which provides the probability of the detected key point being actually visible in the frame without occlusion. Out of the 33 key points returned by the API, only points relevant to each action were stored. The model complexity used for inference can be modified, and the maximum model complexity was used to ensure best accuracy with the drawback of slower inference speeds. Minimum tracking confidence, which represents the baseline for confidence value for successful person detection was increased from default 0.5 to 0.65. This implies that if the confidence of detection of tracked key points is below this threshold value, person detection would be invoked again, otherwise person detection




0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

Figure 3.2: Pose Landmarks BlazePose GHUM 3D [26]

is not invoked in the new frame and the previously calculated keypoints are tracked for future frames.

## 3.4 Normalization

Data Normalization is a key process in any machine learning pipeline to ensure numerical stability of ML models. Standardization of training data is particularly important in simpler ML approaches such as regression and can improve training times as well as accuracy for most ML models. Min-max normalization or rescaling, Z-score normalization and mean normalization are some of the more commonly used normalization techniques.

By default MediaPipe's Pose Landmark model returns x,y co-ordinates in unit scale [0,1] where the keypoint co-ordinates x and y are normalized by the image width and height respectively. However, to future-proof the pipeline developed, additional hip-based normalization was used. In general, the centre of mass defines the central point for any object and in the case of the human body, this point lies slightly above the hips. Also, in all the actions considered, as well as in any scenario in which the subject would face the camera, the hips are most likely to be detected correctly by SOTA algorithms since these are less likely to be occluded when compared to limbs or other extremities. As per the guidelines governing the construction of the dataset, all actions start with the subject facing the camera at the center of the screen at a fixed distance from the camera. Hence, the x and y co-ordinates of the left hip were measured from the first successfully recognized frame from the video. A new frame size for projection of the keypoints was chosen as 640x360, however this may be changed based on requirements in the future.

All keypoints tracked were then projected onto this new frame such that the co-ordinates of the left hip always lie at (0.5, 0.5) and the other points are offset accordingly in the new projected frame using the equations :

$$x_{new} = (x * width_{image} + (width_{frame}/2 - x_{hip} * width_{image}))/width_{frame}$$

$$y_{new} = (y * height_{image} + (height_{frame}/2 - y_{hip} * height_{image}))/height_{frame}$$

## 3.5 Training Dataset

The normalized keypoints returned from MediaPipe are used to construct a dataframe for the ML model. A dataframe is constructed such that each row in the dataframe corresponds to a single frame from the input video using Pandas data analysis library in Python. The columns of the dataframe are the normalized x and y co-ordinates of key points. All 33 key points returned by MediaPipe are not used for creation of the training dataset, only the discriminative key points for each action are considered. The keypoints tracked for each action are tabular in the below table.

| **Action** | **Keypoints** |
|---|---|
| Squat | Left & Right Hip, Left & Right Knee |
| Jog Spot | Left & Right Hip, Left & Right Ankle, Left & Right Foot Index |
| Walk on Line | Left & Right Hip, Left & Right Foot Index, Left & Right Heel |
| Run Stop | Left & Right Hip, Left & Right Foot Index, Left & Right Shoulder |

Table 3.1: Keypoints tracked

### 3.5.1 Outliers and Bad Data

During inference of MediaPipe it was observed that the framework was not detecting complete keypoints in all frames and in some frames no keypoints were being successfully detected. The number of frames in each video on which keypoint detection failed completely were counted. A skip frame % was calculated for each video by taking the ratio of skipped frames to the total number of frames in the video. Videos with the percentage of skipped frames greater than 40% were dropped from the study.

In addition to this it was also observed that there were videos in which no activity was being performed and the child stayed still throughout the duration of the video. Such videos would have a negative effect on the heuristic labelling approach defined in the next

section and has to be removed from the training set. In order discern such bad videos, the minimum and maximum values for the main key points from Table 3.1 were noted for each video. Videos in which the observed maximum and minimum values were close, were dropped from the dataset.

## 3.6 Intermediate Labelling

For each row in the in the dataframe pertaining to a single frame of the video, an intermediate label was assigned. The label represents the current state of the subject in the frame. The labels represent various intermediate phases/states, which performed in the correct sequence, collectively comprise the complete action which is to be recognized. The labels were created based on heuristics created based on various factors such as co-ordinate of discriminative keypoints, as well as distances between keypoints. Various different heuristics were tested to identify an ideal fit. The best performing heuristics created for each actions are discussed below.

### 3.6.1 Squat

The criteria for a proper squat activity was defined as when the child faces the camera and bends the knees keeping the trunk (almost) straight. Thus the objective in this case is to identify a proper squat satisfying the given criteria, and to count the number of such proper squats performed during the duration of the video. Bending the knees such that the ankles almost come to the same height as the hips is a key discriminative factor to recognize a proper squat. This trait was used to create the heuristic for squat action. In addition to the keypoints detected from MediaPipe, 2 new additional features were calculated and added to the dataframe - the distance between the left knee and the left hip and the distance between the right knee and the right hip. For a proper squat this distance would reduce significantly since the subject is facing the camera.

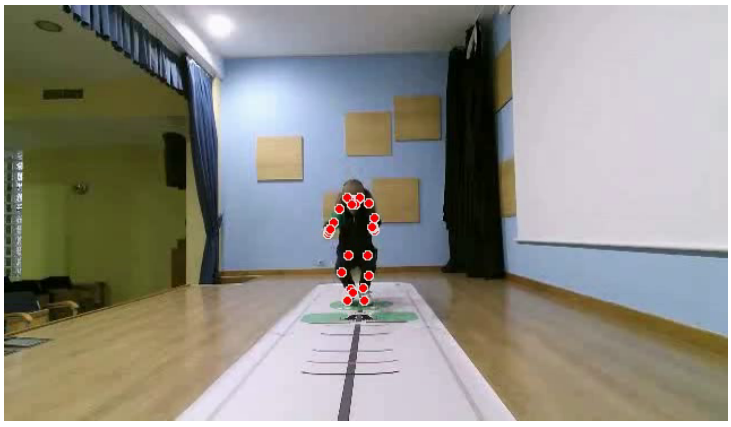


Figure 3.3: Squat Posture

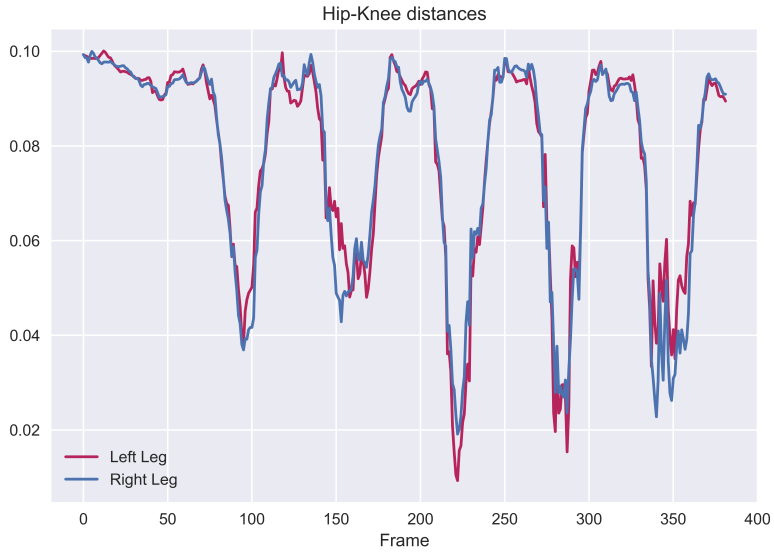



Figure 3.4: Hip-Knee distance vs Frame

Thus 2 intermediate labels were assigned based on the hip-knee distances of the legs as follows: if left knee-hip distance and right knee-hip distance $<$ threshold, then the intermediate label was assigned as Squat. If the distances were greater than the threshold then an intermediate label of standing was assigned.

- **Determining Threshold** The value of the threshold is key in obtaining a proper classifier, especially since there are only 2 class labels assigned. From plots of hip-knee distance vs frame for Level 2 videos as shown in Figure 3.4 it was observed that in a large majority of the videos an absolute cut-off value could be defined as clear valleys and peaks can be observed in the graph. However defining one single value for the entire set of videos would be difficult. Since possible outliers in hip-knee distance, such as cases where the child is standing still were removed as part of outlier removal 3.5.1, a drop in the hip-knee distance could be guaranteed for all videos. The mean of the hip-knee distance for each videos was used to obtain an approximate value for the threshold. Subsequently various different values around the mean were tested using trial and error by visualizing the quality of the heuristics on a set of random videos and the value was fine-tuned to 0.05.

### 3.6.2 Jog Spot

Jogging on a spot action was defined as taking individual steps on each leg, while staying at a single stationary spot at a fixed distance from the camera. A step in this context was defined as the action of lifting and landing a single foot on the ground. Thus the final objective is to obtain the number of such steps taken on each individual leg.

Various different heuristics were tested for the action to find the heuristic that returned

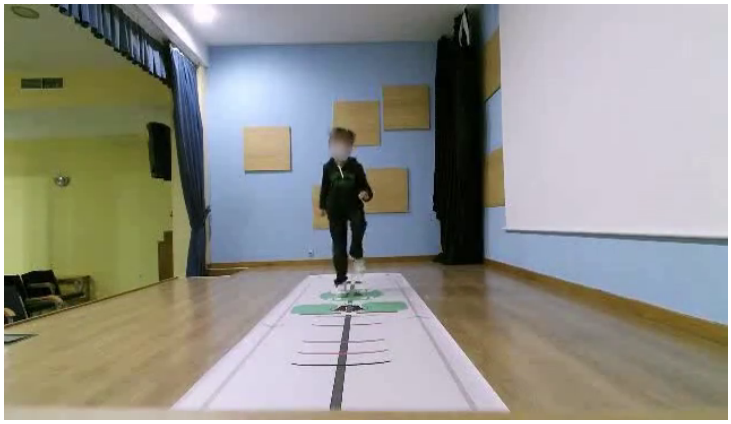


Figure 3.5: Jog on a spot

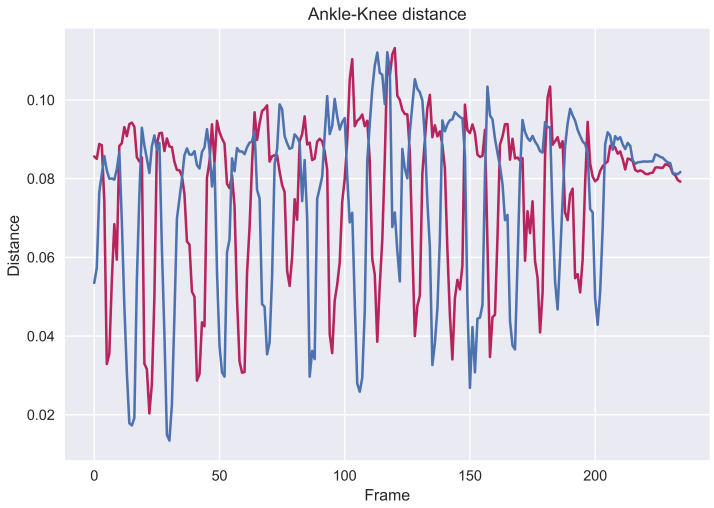



Figure 3.6: Ankle-Knee distance vs Frame

the best results. The best performing heuristic was obtained using distances between the ankle and the knee of the subject in the video. Alternate heuristics using the y-coordinate of the foot was explored, but yielded poorer results and is not discussed further.

The heuristic was created based on the bend in the lower half of each leg during jog activity. This approach is similar to the heuristic used for squat action and is dependent on the fact that the subject is facing the camera and is always at a constant distance from the camera. From plots of Ankle-Knee distance vs frame number for the set of Level 2 videos, clear peaks and valleys was be observed in the graph. The plot for a single video is pictured in Figure 3.6. However, unlike the case of squat action, the peaks and valleys of both legs do not coincide and instead have a temporal shift. However, from the bend in the knee it is still possible to ascertain which leg is currently lifted off the ground and hence the single-frame labelling is still applicable in this scenario. Therefore, two intermediate frame labels were assigned based on the condition: if left ankle-knee distance $<$ threshold and right ankle-knee distance $>$ threshold, then the label was assigned as

'Left Leg Up' and for the alternate case of left ankle-knee distance > threshold and right ankle-knee distance < threshold label was assigned as 'Right Leg Up'. Frames that did not satisfy any of the aforementioned conditions was assigned the label 'Unrecognized'.

Similar to the case of squat action, obtaining a good value for the threshold is a key factor in model efficacy. The same approach of finding the mean and further fine-tuning was used to obtain a threshold value of 0.065 for the threshold value.

### 3.6.3 Walk on a Line

The walk on a line activity consists of a number of heel-toe steps on a fixed line on the ground. Thus the objective is to identify a proper activity of walking on a line and to count the number of true heel-toe steps. A heel-toe step is defined as when one foot goes in front of the other along a line where the heel of the front foot touches the toe of the foot in the back.

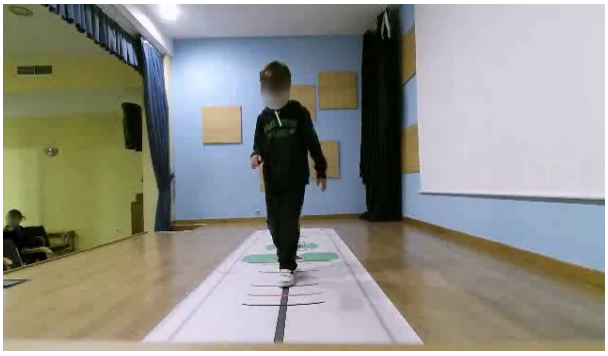


Figure 3.7: Walk on a line

Walk on a line activity is a more complicated activity because of a number of reasons - firstly, unlike previous actions which were always done at a stationary position at a fixed distance from the camera, this activity involves forward motion towards the camera. Secondly, occlusion issues which were absent in the previous two actions are a major concern in this action since each foot constantly eclipses the other during each heel-toe step. In addition to this, in order to distinguish between walking on a line and a simple walk activity towards the camera, it is imperative that the feet are placed on the line provided to the subjects. Thus, this line has to be detected and any step outside the line is not a proper step.

#### 3.6.3.1 Line Detection

Owing to background clutter as well as other noises in the video input, there are a large number of lines in each frame of the video. However, only the thick line on the mat on which the subjects stand is of interest to identifying walking on a line activity. In all videos, the position of the camera and the mat on which the subjects perform the

actions on are constant, which means that the co-ordinates of the thick line are constant. However, minor variations to both these parameters were observed across different videos. This meant that a robust pipeline to detect the thick line that the subjects walk on is required. To detect the co-ordinates of this line, a pipeline was developed using OpenCV package in Python. OpenCV is an open source Python library that offers a large number of real-time image and video processing functions.

- Dilation: Dilation is one of the most commonly used morphological operators in image processing. A morphological process is simply an image processing method which applies a structuring element on an image to produce a new modified output image. Based on the structuring element applied, different features can be obtained from a raw input image.

  Dilation involves convolving an input image with a kernel B. Dilation process works only on grayscale images. In this process, the kernel is moved over the entire input image such that the center of the kernel B, called the anchor point, coincides with each pixel in the input image. Then the maximal pixel value of the values in A overlapped by the kernel B is taken and the pixel value at the anchor points is set to this new maximal value. Since each pixel value is changed to the maximal value in its surroundings, this implies that generally brighter objects in an image grow in size when dilation is applied, since white pixels have higher values than darker pixels in grayscale images.

Figure 3.8: Dilated Image

OpenCV library in Python provides a direct function for dilating an image, and multiple passes/iterations of dilation can be applied. After 2 iterations of dilating the input image, it was observed that the extraneous thinner horizontal lines on the mat surrounding the thick line were removed as observed in Fig 3.8. Thus due to the fact that white regions grow after dilation, all thin black lines on the mat were removed, except for the thick line which reduced in thickness. However, there were still a large number of lines in the background of the image which have to be removed. In addition to this, due to vanishing point effect the thickness of the line

to be considered reduces farther away from the camera. Thus, after dilation the full length of the thick line may not exist in the dilated image. This means the the actual end points of the thick line are not available for further processing.

- Region-Of-Interest Mask: Applying a line segment detection algorithm to the complete image would return a a large number of lines as shown in Figure 3.11 . In order to create a pipeline which is less prone to noise and capable of extracting the correct line accurately, a region of interest mask was applied to the image. Since the mat is generally fixed near the center across all videos, a fixed set of 3 endpoints defining a triangular region such that the bottom end of the thick line would always fall within the triangle was defined. The endpoints of the triangle were defined in terms of image height and width so that the approach would be suitable for any such scenarios with a central thick line. The endpoints were chosen from trial and error by running the line segment detection pipeline on different background from the training data.

  To create a Region-Of-Interest mask, a black image with the same dimensions as the input image was created and using fillPoly function in OpenCV the triangular region within the black mask was filled with white color. The bitwise AND operation between the input image and the mask was then computed. Since the bitwise AND operation between white (hex 255, binary 11111111) and any other colour returns the colour itself, the final output image of the operation contains the triangular section from the original image surrounded by black as shown in Fig. 3.9.

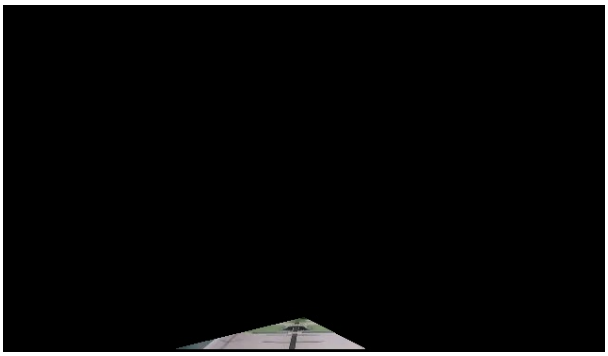


Figure 3.9: Region-Of-Interest Mask

- Line Segment Detector: OpenCV provides a Line Segment Detector class which returns the end-point co-ordinates of all lines detected in an image passed to it. The algorithm used for detection requires little to no parameter tuning and detects locally straight contours within an image where a contour is defined as a fast transition from bright to dark pixels or vice versa [27].

  The Line Segment Detector returns all the lines detected in the image. Even after dilation and applying a ROI mask on the input image, there were still small artefacts

in the image near the thick line within the ROI which were wrongly detected as true lines. Also the 2 edges of the thick line were detected as separate lines. Thus only one line of all the returned values need to be considered.

From the returned line endpoints, a function was created to find the line segment that had x co-ordinate of lower endpoint closest to half the width of the image, since this would correspond to the right edge of the central thick line. The line segment found using this approach would always be the right edge of the central line to be detected.

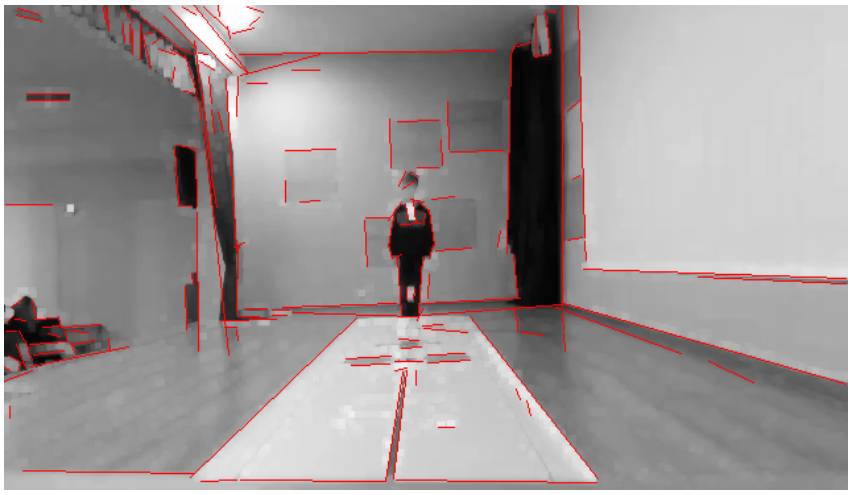


Figure 3.10: Detected Lines

- Extrapolation: As mentioned earlier, due to dilation process the entire length of the detected line is not available. In addition to this, the ROI mask applied does not contain the entire length of the thick line. As a result, the line segment returned from the line segment detector contains only a part of the complete line and the endpoint of the line farther from the camera is not accurate. To account for this shortening, the detected line was extrapolated to 0.4 times the height of the image. The value of 0.4 was chosen based on the assumption that that the start of the actual line would be approximately be near the half the height of the image, which was the case for most videos in the dataset provided and from further fine tuning from trial error it was observed 0.4 times the height returned the best approximation of the actual line.

The endpoints of the central line obtained using the aforementioned steps were normalized using the same normalization applied to the features obtained from MediaPipe. These co-ordinates were then added as 4 new features to the training dataset to represent the maximum and minimum x and y values within which each step has to be taken. To identify whether a step falls within this bounding box, the x and y co-ordinates of the left and right foot indices were taken. If at any frame the foot indices were outside the

allowed minimum and maximum values, a label of 'Stepped Out' was assigned to signify incorrect steps.

#### 3.6.3.2 Forward movement heuristic

In order to obtain the count of heel-toe steps taken during the activity of walking on a line, it is necessary to create labels to specify which foot is closer to the camera at each frame. MediaPipe's Pose Estimation model returns z-axis co-ordinates for depth, however these values were found to be highly inaccurate. Hence the y co-ordinates of left and right foot indices were used to ascertain the foot closer to the camera. When the subject moves closer to the camera, the y-coordinate of the index of the foot increases in the positive direction. This factor was used to assign two new labels to the dataset - if the y co-ordinate of the left foot was greater than that of the right foot, the label was assigned as 'left foot front' and 'right foot front' otherwise. An alternate heuristic using the probability of occlusion returned by MediaPipe was also tested such that if the visibility of the left foot index was less than a certain threshold value, then this would mean that the right leg was completely eclipsing the left and vice versa. But this heuristic approach yielded very poor results and was not explored further.

### 3.6.4 Run and Stop

The run and stop action involves the child running towards the camera and coming to a complete stop at a fixed distance from the camera. The number of steps taken by the child once the child crosses the black horizontal lines on the floor mat are to be counted as well.

Figure 3.11: Run and Stop

In order to identify the point at which the black horizontal line on the ground is crossed, a line segment detection pipeline similar to the previous case was used. Only 3 key changes were made to line segment detection - Firstly, dilation was not used since the line to be detected is not a thick line and applying dilation would remove the thinner line. Secondly, a rectangular Region-Of-Interest mask was used to capture a large portion

of the mat on which the children were standing. And finally, out of the line segments returned by the Line Segment Detector, the line with 0 slope and lowest y co-ordinate was used to find the first horizontal black line. The normalized y co-ordinate of the horizontal line thus detected was then added as an additional new feature to the dataset.

Similar to the previous case of walking on a line, the z-axis co-ordinates could not be used to identify forward motion. Hence the y co-ordinate of foot indices were used to determine the foot closer to the camera. 3 intermediate labels were assigned as follow : if the y-coordinate of the foot index is less than the y co-ordinate of the black horizontal line, then a label 'Behind Line' was assigned. Once the y co-ordinate of the foot exceeded the detected line, then 2 labels were assigned as 'Left Step' or 'Right Step' depending on which had the larger y co-ordinate value.

## 3.7 Machine Learning Models

After assigning labels to each frame in the dataset, the problem has been simplified down to a classification problem. Different supervised machine learning models were tested to check which model returned the best results for this classification problem. Prior to training models on the dataset, the newly created features to assign intermediate class labels, such as ankle-knee distance were removed from the dataset since the model would assign highest weight to these features and in the worst case, the model would not learn the relationship between the class and other key point features. Thus the additional step of machine learning algorithms in the pipeline will improve scalability and add generalization to the system developed as the system learns to identify intermediate labels from data on the joints alone (and co-ordinates of the line for walking on a line action).

### 3.7.1 Random Forest Classifier

Random Forest Classifier is a supervised machine learning algorithm which utilizes a collection of decision trees that are fitted on different subsets of the training dataset. Random Forests are known to produce good outputs for a large number of problems with very little parameter tuning. The different training data subsets used for each tree are decided based on different metrics such as Mean Squared Error(MSE), Information Gain or Gini impurity. Random Forests are an ensemble machine learning method which use majority voting among all different decision trees to predict the final class label for classification problems. The ensemble method used in Random Forests is called bagging, or bootstrap aggregation which uses sampling with replacement to create individual sample sets. Since the sampling is done with replacement, certain datapoints may be sampled more than once. To reduce correlation between different individual trees, each tree consid-

ers only a subset of the total feature space through an approach called random subspace method or feature bagging [28].

Each individual decision tree is trained on the data using algorithms such as Classification and Regression Tree (CART) algorithm. Generally decision trees can easily overfit on the dataset and are known to be biased, but because of the majority voting method employed in Random Forests, the algorithm is more robust and less prone to overfitting. This approach also reduces variance especially when working with noisy datasets such as the dataset considered for this work. The Random Forest Classifier class in sklearn's ensemble library in Python was used to train a random forest model on the dataset. The model has a number of hyperparameters such as the number of estimators which represents the number of individual decision trees in the forest, the maximum possible depth for each individual tree, the minimum number of samples required to split intermediate nodes and the minimum number of samples required to be leaf node.

### 3.7.2 Neural Network

Neural Networks are deep learning algorithms that adopt biomimicry of neurons in the human brain to create machine learning models. Neural Networks consist of a number of layers formed by nodes. The input layer and the final output layer of nodes of a neural network sandwich an n number of hidden layers. A neural network with a large number hidden layers are deep networks, while those with few hidden layers are shallow networks.

Each node in the network has a weight, bias and a particular threshold value associated with it. Each node also has a connection to all the nodes in the next layer. When data is received at a node, the output for the node is calculated using an activation function applied on the received input as well as the weight and bias of the node. If the output value thus calculated exceeds the threshold, the neuron 'fires' and sends output to its subsequent connections. A shallow neural network was implementeed in Python using the MLPClassifier class from sklearn's neural network library. Multi-Layer Perceptron (MLP) is a basic feed-forward neural network in which each node in the hidden layer is fully connected to the next layer. Since it is a feed-forward network, information flows only in one direction from the input layer to the output layer and the outputs from the output layer are not fed back to the model for weight updates as in the case of more complicated networks such as RNNs. Unlike decision trees which use a greedy search on metrics such as information gain or gini index, neural Networks learn by optimizing a loss function and for the MLPClassifier, the loss function used is log-loss or categorical cross-entropy. The MLPClassifier takes a number of key hyperparameters that decide its performance - the activation function used on each node, the learning rate parameter, the batch_size as well as regularization parameters alpha.

## 3.8 Buffer Count

The predictions from the machine learning algorithms return only the intermediate class label. The intermediate labels are sufficient to identify a proper or improper intermediate state for each frame, however, in order to identify the number of proper actions, the intermediate labels are appended to a buffer during inference from a video. During inference itself, transitions from one intermediate label to another are monitored. When the right transition for a properly executed action is encountered, a counter variable is incremented. The value of the counter variable at the end of inference will return the number of correctly performed actions. The transitions for different actions and the associated heuristics are summarized below:

- Squat: For squat action, a transition from 'Standing' to 'Squat' was considered to be one squat. Thus if label predicted for the current frame is Squat and that of the previous frame from the buffer was Standing, then this implies a proper transition from standing to squatting posture. This approach was considered based on the assumption that the child would start the video in a standing position.

- Jog Spot: For jogging on a spot action, 3 intermediate labels were assigned in the previous stages - 'Left Leg Up, 'Right Leg Up' and 'Unrecognized'. Two separate counters were maintained to monitor the transitions for each leg. Thus if the label prediction for the current frame is Left Leg Up and the prediction stored for the previous frame from the buffer is not Left Leg Up, then the counter for the left leg was incremented, and for the mirror case with the conditions for the right leg, the counter for the right leg was incremented. The above conditions ensure that actions with label Unrecognized do not increment any counter.

- Walk on Line: For walking on a line activity, 3 labels were assigned - Left Foot Front, Right Foot Front and Stepped Out. If the current label prediction is Left Foot Front and the previous prediction from the buffer is not Left Foot Front, then a counter for the left leg was incremented and similarly for the right leg, another counter variable was maintained. The above counter conditions ensure that no counter variable is incremented for Stepped Out action.

- Run and Stop: For run and stop activity 2 counters for each leg were maintained. The counter for the left leg was incremented when a transition from 'Right Step' to 'Left Step' is encountered and vice versa for right leg. Thus the counters are only incremented when both legs have crossed the horizontal black line and 'Behind Line' label is not observed in the previous frame.

# 4 Experiments and Results

## 4.1 Introduction

This chapter covers the experimental setup used to evaluate the action recognition pipeline detailed in the previous chapter. The chapter explains the results achieved by each model tested for the 4 actions. This section also analyzes the heuristics used in each action and outlines the advantages and shortcomings of each heuristic used. Finally the chapter summarizes the key challenges faced during experimentation with the pipeline developed and addresses general drawbacks in the proposed approach.

## 4.2 Experimental Setup

For all 4 activities, the entire set of videos were considered. However, videos with a high number of frames for which MediaPipe failed to perform key point extraction on, were dropped from the training dataset. A threshold of 40% was set such that if the percentage of skipped frames exceeded this value, the video was dropped. The total number of videos available and the number of videos skipped are summarized in 4.1. Since there is sufficient data to train a simple 2/3-class model, train-test split was used to evaluate the generalization capabilities of the models on unseen data. The same splits of the dataset were used on both models for each action so that a sound comparison could be done between the two. A standard 70-30 split was used for all experiments. The test dataset results for the ML models for all actions are summarized in 4.2.

Although the dataset provided contains labels for the actions performed, the count of each activity performed in each video was available only for a small subset of around 50 videos per action. As a result, there was no straightforward method to test the efficacy of the buffer count approach without manually checking each video. Since manual verification is tedious for over 2000 videos, the performance of the pipeline was manually tested on 10 videos each for each action. The 10 videos considered for manual verification were chosen completely at random to avoid any bias. Thus any mention of counts within subsequent sections of this chapter refer to the total counts observed from the 10 videos

checked manually and the final count results are summarized in Table 4.3.

| Activity | Total Videos | Skipped Videos |
|---|---|---|
| Squat | 516 | 502 |
| Jog Spot | 504 | 483 |
| Walk Line | 497 | 461 |
| Run Stop | 511 | 445 |

Table 4.1: Training Data

| **Action** | **Model** | **Label-Test Accuracy** |
|---|---|---|
| Squat | RF | 92 |
| | NN | 98 |
| Jog Spot | RF | 85 |
| | NN | 93 |
| Walk Line | RF | 72 |
| | NN | 66 |
| Run Stop | RF | N/A |
| | NN | N/A |

Table 4.2: Model Results

## 4.3 Squat

From the model results in 4.2 it is observed that both the Random Forest Classifier and the Shallow Neural Network both achieve very high accuracies on the test dataset with scores of 92% and 98% respectively. The exceptional performance of both algorithms is expected since the problem is a 2-class classification problem with sufficient amounts of data.

However accuracy values alone are not a good indicator of model performance. From the confusion matrix for the random forest classifier on the test split shown in Fig. 4.1, it is observed that the number of false negatives, i.e the actual squat cases which are wrongly detected as standing are higher than false positives. Since the ultimate goal of the pipeline is to create a system to monitor motor skills development in children, false positives could allow children with growth issues to inadvertently pass the system if it falsely labels incorrect postures actions as squat. Hence more priority should be given to false positives and the less number of false positives from the confusion matrices is a good indication of model performance. The neural network exhibits more stable predictions with an almost equal number of false negatives and false positives. However, the accuracy values stated are completely dependent on the quality of the heuristics. As a result the model metrics such as the accuracy values and the confusion matrices are insufficient to analyze overall system performance.

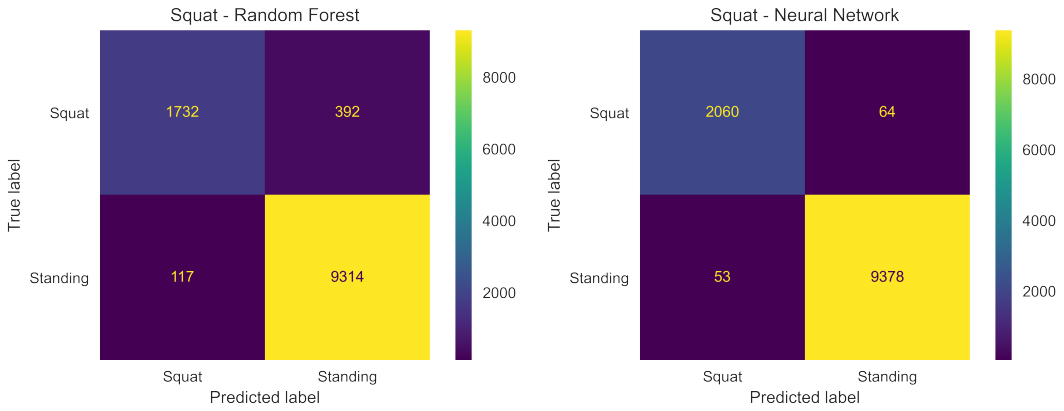



Figure 4.1: Confusion Matrix - Test Set

To determine the overall accuracy of the pipeline in determining counts, the results returned by the system are cross-checked with the values obtained through manual verification. The results from the better performing model for each action, which is the Neural Network in this case, are summarized in Table 4.3. From Table 4.3 it is observed that the count prediction accuracy is fairly high at 93%. This shows that the system is well capable of obtaining the count of squat actions from a video, with only a small margin of error. However, the accuracy stated is obtained from manual analysis of a small subset of the complete training set. Without analyzing the performance over a much larger set, it is difficult to ascertain the efficacy of the system and the obtained value of 93% is most likely to drop when tested on the full data.

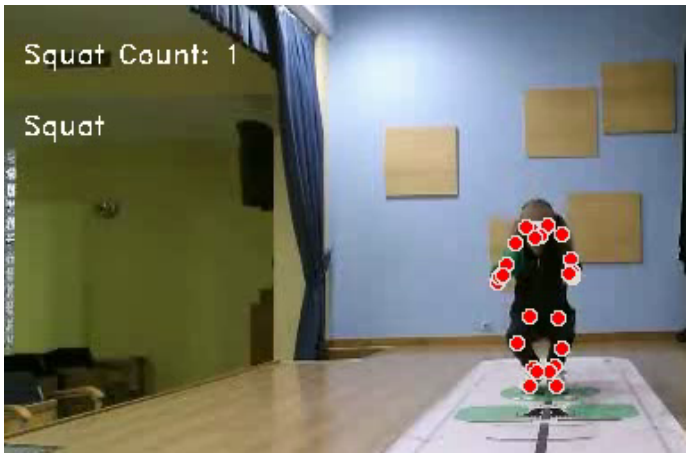



Figure 4.2: Squat - Label predictions and counter

| Action | Model | Predicted Count | Actual Count | Accuracy |
|---|---|---|---|---|
| Squat | NN | 77 | 72 | 93% |
| Jog Spot | NN | 164 | 206 | 80% |
| Walk Line | NN | 92 | 128 | 61% |
| Run Stop | NN | 0 | 0 | 0 |

Table 4.3: Count results

## 4.4 Jog on a Spot

A key observation from Table 4.1 summarizing the skipped videos is that a larger number of videos were skipped for this action as MediaPipe was unable to extract features properly. This finding could be due to the faster nature of jogging activity when compared to squat action. It was observed that generally, MediaPipe had issues maintaining tracked frames from one frame to the next due to more erratic motion. From Table 4.2 the results for test accuracy of the 2 models for jogging on a spot activity indicate that the neural network is a better performing model with a test accuracy of 93%. From the confusion matrices it may be observed that misclassifications between right leg up and left leg up labels are very scarce - which shows that the machine learning model was able to learn the underlying heuristics from the training data points. However, most of the error arises from the Unrecognized label due to class imbalance as entries for this class were much higher than that of the other two. This label was used as a representation of standing posture, or any other posture that does not involve a folded leg. In many videos it was observed that the children were standing stationary during the start and end of the videos before and after performing the activity. This is most likely the reason for a large number of entries for this class. In general class imbalance can have adverse effects on any ML model and should be handled by methods such as sampling. But since the number of false positives was much lower than that of false negatives, handling class imbalance was omitted for this action.

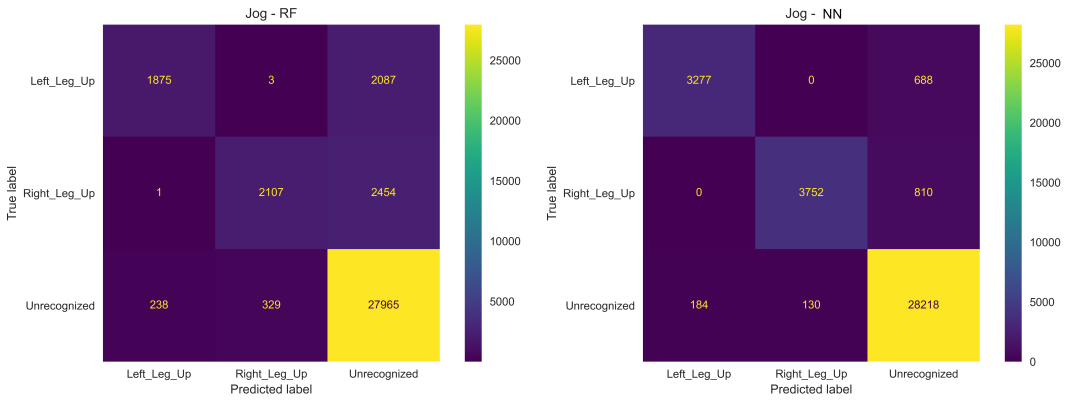



Figure 4.3: Confusion Matrix - Test Set

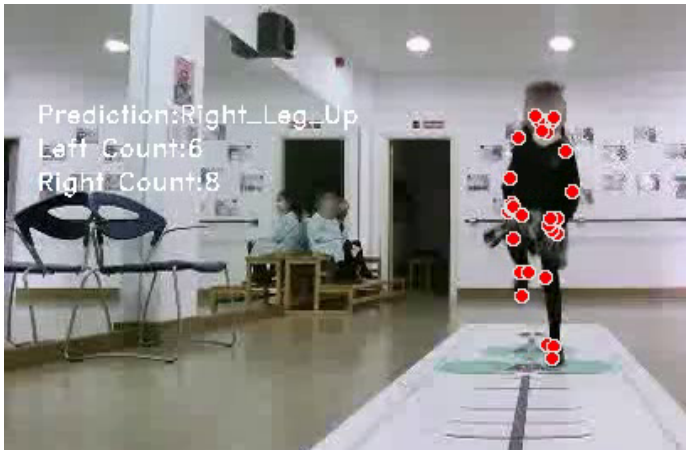



Figure 4.4: Jog Spot - Label predictions and counter

As mentioned in the previous case, the model test accuracy is not a good indicator of overall performance. From the count results calculated for 10 videos via manual verification in Table 4.3, it is observed that the system is able to achieve an overall accuracy of 80%. From the manual verification test, it was also noted that the system was accurate in most videos with older children, but count accuracy dropped significantly for younger children. This drop in count accuracy for younger children was traced back to problems in the heuristic labelling applied - most younger children did not properly bend the legs as expected for a proper jog action. Instead they simply lifted each leg from the ground by a small amount and as a result, the intermediate labelling which depends on the bend in the knee failed to label such cases properly.

## 4.5 Walk on a Line

For walk on line action 3 labels were assigned using heuristics - left foot front, right foot front and stepped out. However, from the training data created it was observed that the number of inputs with the label stepped out were very low, while right and left foot front labels were of equal number. This observation was not surprising since in most videos the children stayed very close to the actual line. But this class imbalance was severe and if left unhandled, could severely reduce the model's ability to learn the 'stepped out' case which is crucial in identifying wrong steps. Hence the minority class of 'stepped out' was oversampled using RandomOverSampler function from imblearn module in sklearn library to create equal number of classes. From the model results from Table 4.2 it may be observed that the Random Forest Classifier outperforms the Neural Network. From the confusion matrices shown in 4.5, it can be observed that the Random Forest Classifier as well as the Neural Network are able to completely classify all 'stepped out' labels after oversampling.

From the count results in Table 4.3, it is observed that the model attains only 61% overall accuracy and the number of total counts predicted is greater than the actual count. From manual verification it was observed that when the child is far away from the camera, the changes in y co-ordinate of the feet are very small and the system struggle to understand when one foot was in front of the other. Another observation was that the co-ordinate values returned from MediaPipe were not stable and had minor error between frames. Since the heuristic directly compares the values from MediaPipe to identify the closer foot, wrong labels were assigned in random frames in between. Consequently, this caused the counter to increment for such noisy frames which resulted in the counter overshooting the actual count.

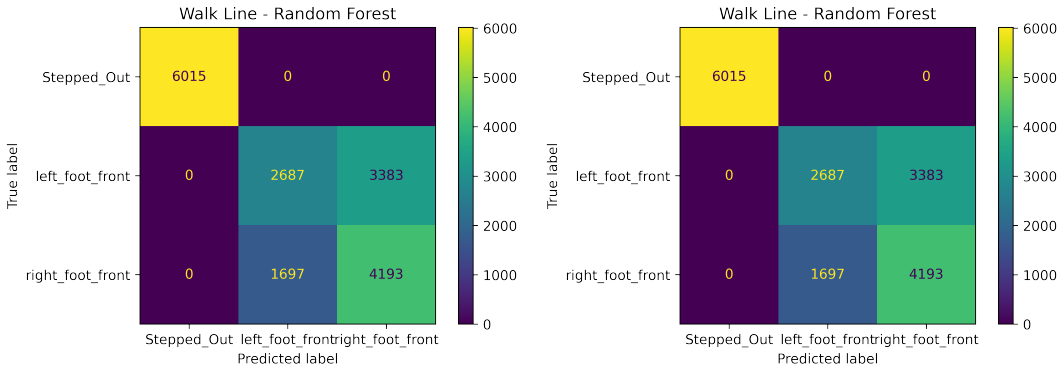



Figure 4.5: Confusion Matrix - Test Set


Figure 4.6: walk Line - Label predictions and counter

### 4.5.1 Limitations

A key limitation in the pipeline developed for walk on a line activity is the difficulty in identifying the closer foot during forward motion. If the z-axis co-ordinates returned by MediaPipe were accurate, this issue could have been bypassed. Alternate approaches to estimate depth from 2D images are complicated approaches which use complex networks to create depth images from various factors such as textures and perspective. Such

approaches were beyond the scope of the dissertation. The approach used in the dissertation of utilizing y co-ordinates of the feet to identify the forward foot is highly susceptible to noise, and since the co-ordinates returned by MediaPipe had minor variations across frames, this approach has inherent stability issues. Another drawback in this approach is the inability to measure if heel-toe steps are maintained - if a foot is placed on the line at a non-zero distance from the foot in the back, the system would still consider this as a valid step. However such a step does not satisfy the criteria to be classified as a proper heel-toe step in walking on a line action. Methods to estimate foot length using known key points were tested, however these approaches were not fruitful primarily due to occlusion issues.

## 4.6 Run and Stop

A system that satisfied all the basic conditions to identify run and stop action could not be developed using the proposed approach. The major issues faced for this activity are explained below:

- Identifying stop state: In a majority of the videos for this action it was observed that the subjects came to the stop state very close to the camera. This meant that a large portion of the skeletal structure was outside the camera's field of view. This meant that MediaPipe did not have a complete skeleton to infer key points from. As a result, in some of the cases wrong co-ordinates were returned which were completely unusable as shown in 4.7.

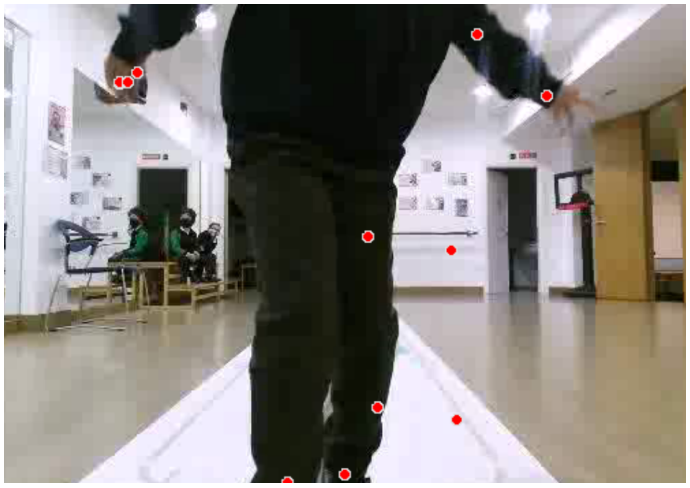


Figure 4.7: Run Stop - Stop state too close to camera

In the majority of cases, no points were returned from MediaPipe for such frames and these frames were skipped. The root cause of the failure of MediaPipe in this case can be traced back to how the framework detects a person - MediaPipe uses a face detection algorithm to detect the presence of a human. Since in most of the

video the face of the children were outside the viewframe, person detection failed. As a result, the number of skipped frames for this activity was found to be much higher than that of the other 3 actions. Since the root cause of this issue can be traced back to improper placement of the camera, very little could be done within the scope of the dissertation to correct this.

- Fast motion: When a large amount of motion is present in the input video, MediaPipe had issues extracting key points as the framework had issues tracking detected skeleton from one frame to the next. In cases where there were stationary adults in the background, once the child started running, the skeletal detection would fail and MediaPipe would detect the skeletal structure of the stationary adult in the background. Even in cases where no other adults were present in the frame, the framework returned completely false co-ordinates. In both cases, the returned values were unusable. Although the first scenario could have been corrected by cropping videos to the subject of interest, no direct approach to handle the second case was found.

- Constraints: Although forward motion of the subject in the video could be estimated using y co-ordinate increase as used in the previous case, or by using increases in lengths of limbs, determining whether a forward motion was a run action proved to be a problem that could not properly defined with the proposed system. A run action towards the camera and a normal walk action towards the camera can only be differentiated from one another based on the speed and overall motion. However, the system developed does not account for such variables. Such actions would require time or/and the previous state of the subject as inputs to determine an action. As such, the single-frame heuristic approach fails in this case as the action cannot be classified from single frames alone. Hence a time-series approach utilizing Recurrent Neural Networks or Long Short-Term Memory Networks could prove to be more accurate for run and stop actions. However, issues in feature extraction faced by MediaPipe, as well as issues due to camera placement would negatively affect these models as well.

## 4.7 Challenges

The proposed system has a number of shortcomings, the major challenges faced are summarized below.

### 4.7.1 MediaPipe Limitations

Although MediaPipe performed well in most cases where the full body of the child was visible, the framework failed to give proper feature extraction in a number of cases. Actions where one part of the body completely occluded another - such as walking on a line when one leg fully blocks the other, were particularly problematic cases. In many cases it was observed that the detected limbs would swap with each other, for example if the left leg eclipsed the right leg in a frame, the detected limbs would swap during complete occlusion. This caused the returned x and y co-ordinates to adversely affect the heuristics and consequently wrong intermediate labels were assigned. MediaPipe occlusion handling primarily considers cases where only part of the body is visible primarily and not scenarios where one body part completely eclipses another.

Another scenario that MediaPipe struggled with were long uniforms. Cases where the upper body clothing extended beyond the waist proved to be difficult for the algorithm. Figure 4.8 shows such a case where the algorithm successfully detects key points in the face and arms, but struggled to determine the points from the hip to the feet. Such cases are generally difficult for many SOTA algorithms because of how clothing can completely obscure the underlying skeletal structure.

The third case where MediaPipe faced issues is in videos with a large amount of motion. Activities like run and stop had a large amount of movement in all 4 limbs and the motion occurred with high speeds as well. In such cases point extraction was difficult as the framework could not keep track of the key points between frames. The final case where the framework faced challenges is detecting key points of smaller children. Younger kids have smaller skeletal structures and the key points detected by MediaPipe were too close together to be useful. In general, even SOTA algorithms would not have been trained extensively on data pertaining to very small children and the models are generally better aimed towards adults. It was also observed that the smaller bodies of younger children were more prone to be completely occluded by uniforms.

### 4.7.2 Camera Placement

Although camera placement was at a fixed position from the subjects in all videos, in some cases the height of the camera was not consistent. This resulted in failure of proper line detection since the the thick central line to be detected did not always fall within

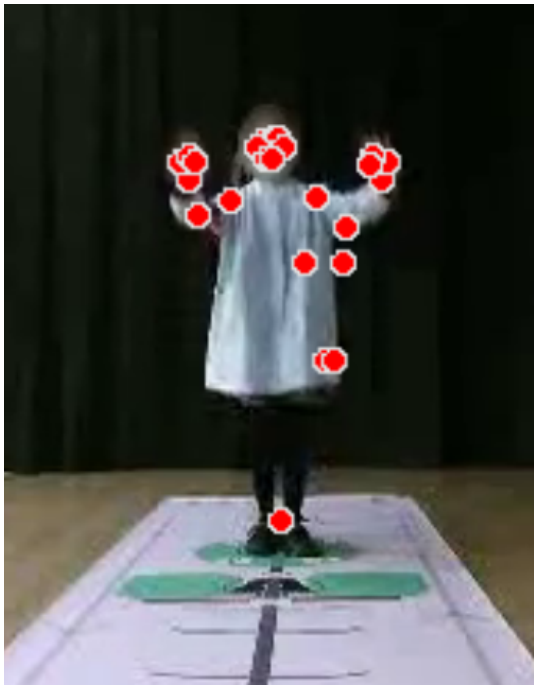


Figure 4.8: MediaPipe failure with long uniforms

the Region-Of-Interest defined in the line detection pipeline. While this affected only a small subset of the videos for walking on a line action, the camera placement issue had a severe impact on run and stop action as mentioned earlier. In large majority of videos verified manually, it was observed that the child would come to a stop after the run action too close to the camera. This meant that a large part of the body of the subject would fall outside the view of the camera, causing MediaPipe's pose estimator to fail completely since no complete skeletal structure could be determined. This issue meant that even alternate approaches using RNNs or LSTMs would not work since the initial feature extraction itself would fail.

### 4.7.3 Variability

The major challenge faced across all actions is the variations in the dataset caused by the unpredictability of children. While a large majority of older kids performed most actions somewhat consistently due to improved cognitive abilities, the amount of variation in younger children was found to very high. All of the younger children approached the same actions in vastly different ways. As a result, finding a heuristic that was accurate on a majority of smaller children proved to be a very complicated problem. While squat action had less variation since there are only a limited number of ways to do a squat, there was a large amount of variation in child behaviour for the other 3 actions - for jogging, some of children simply stood in place and raised one leg after the other instead of jogging. For run and stop the behaviour was highly erratic as not all children were running in a straight line and instead took curved paths. The speed was also highly

inconsistent between age groups. Thus in general the intra-class variation when working with videos of children proved to be a major challenge. Creating separate heuristics for different age groups will most likely improve the performance of the system for squat and jog on a spot actions.

### 4.7.4 Distinguishing Actions

In human action recognition problems, distinguishing similar actions is always a challenging task. The same problems are applicable for some of the actions considered in this research. While the heuristic conditions used to identify squat action is strong and less prone to mislabelling other actions, the same cannot be said for the other actions. Considering the case of jogging on a spot action, folding each leg one by one at a slow pace would trick the system into classifying each fold as a jog on one leg. similarly for walking on a line activity, since the heel-toe distance is not checked, it is possible to deceive the system by taking large steps on a line.

### 4.7.5 Validation

The primary challenge with the pipeline developed is obtaining a concrete metric on overall system performance. Although count of actions for a very small subset of the videos was provided along with the dataset, this data did not have details regarding the particular time or frame at which a certain action occurred. And also this data was not available for the complete dataset which meant that there was no straightforward method to quantify overall accuracy without manually verifying each video.

# 5 Conclusion and Future Works

## 5.1 Conclusion

The primary motivation for this dissertation was to create a system that is capable of monitoring gross motor skill development in children belonging to various age groups. Such a system has numerous applications in understanding development of children. This body of work was primarily focused at understanding the feasibility of creating such an action recognition pipeline for children, to recognize and count 4 repetitive actions from a video input of varying complexity - squatting, jogging on a spot, walking on a line and running and coming to a stop. The proposed system uses a single-frame approach to identify intermediate states for each activity. Using key points of the skeletal structure obtained from SOTA human pose estimation algorithm MediaPipe, different intermediate labels are assigned to each frame using a heuristic approach based on different discerning conditions for each action. Thus action recognition is converted into a supervised classification problem at individual frame level for which different machine learning algorithms are utilized. Finally a buffer transition counter approach is used to monitor transitions between frame-level labels to count the number of properly performed actions.

From the experimental results, the single-frame approach proved to work well on 2 of the actions - squat and jogging on a spot. Intermediate states can be identified for both these cases from individual frames as both actions may be considered to be single-frame activities. Primarily due to issues from key point detection in MediaPipe on videos with self-occlusion, the system returned only mediocre results for walking on a line action. For run and stop action an adequate system could not be developed mainly due to issues in the data collected itself as children stopped too close to the camera for object detection to work.

## 5.2 Future Works

The primary drawback of the properly working models for squatting and jogging on a spot can be traced back to a finding a single heuristic for all children. Since the quality of heuristic labelling applied will have a cascading effect down the pipeline in terms of accuracy, more steps have to be done at to fine-tune the labelling. Creating different heuristics for different age groups and creating separate models could prove to be beneficial. But this requires additional data regarding the age of the children.

For walking on a line approach multiple different approaches have to be tested in order to identify the best fit. The occlusion issues faced by MediaPipe during feature extraction were a major limiting factor for the performance of the system developed. Using SOTA algorithms more robust to such occlusions may help circumvent this issue, but the easiest solution to fix complete self-occlusion of the legs is to use multiple cameras to capture the video. Having at least one additional camera providing a side view of the action can easily identify the correct landmarks from each leg. Another key limitation is the lack of 3D depth data. A large amount of research has been carried pertaining to monocular depth estimation from 2D images using models such as MiDAS [29]. Using such models to create depth maps could prove to be useful in identifying the foot closer to the camera, however depth estimation is a different research field in its own with a large number of caveats and introducing this to the current pipeline could prove to be complex. Using sensor data would also be an easy way to overcome the issue.

Recurrent Neural Network models such as Long Short-Term Term Memory Networks have proven to be highly efficient at identifying actions from video. Such networks would be very efficient at identifying actions considered in this dissertation. However, feeding a fixed input size and counting individual actions was found to be difficult. There is a possibility of using the existing intermediate labels as an input to an LSTM network, and then labelling each chunk of input as a transition from one action to another. Finally counting the transition predictions from the LSTM could be used to evaluate count. Another approach would be to completely annotate the data providing exact timestamps of when certain intermediate states/actions are performed in the videos. If this annotation can be performed manually, then RNNs can be applied to the data more easily and counts of actions can be obtained directly. Both these approaches are likely to provide better results but require a large amount of effort, especially for the manual annotation and these approaches need to be explored further. In addition to the complexities with fixed input size and identifying start and end frames of actions, another issue with LSTMs in Gross Motor Action Recognition is that these networks operate in a black-box manner and interpreting the outputs from these models is difficult. As a result although an LSTM system would have improved accuracy, such a system would be difficult in assessing the

quality of the motor skills possess by the child [15].

The ultimate goal of the system developed would be to create an Action Quality Assessment (AQA) system that is capable of assigning scores to the actions performed and providing interpretable feedback on improvement and identifying where mistakes were made. A large amount of research is being carried out in the field of AQA [30, 31], but creating such a system for Gross Motor Action Recognition in Children would have great outcomes in understanding growth of children.

# Bibliography


[1] B. Bossavit and I. Arnedillo-Sánchez, "A novel approach to monitor loco-motor skills in children: A pilot study," in *Transforming Learning with Meaningful Technologies*, M. Scheffel, J. Broisin, V. Pammer-Schindler, A. Ioannou, and J. Schneider, Eds. Cham: Springer International Publishing, 2019, pp. 773–776.

[2] A. Bobick and J. Davis, "The recognition of human movement using temporal templates," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, pp. 257 – 267, 04 2001.

[3] S. Herath, M. T. Harandi, and F. Porikli, "Going deeper into action recognition: A survey," *CoRR*, vol. abs/1605.04988, 2016. [Online]. Available: http://arxiv.org/abs/1605.04988

[4] e. a. K.Ishwarya, "Human activity recognition methods: A review," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, no. 9, 2021.

[5] V.-T. Le, K. Tran-Trung, and V. T. Hoang, "A comprehensive review of recent deep learning techniques for human activity recognition," *Comput Intell Neurosci*, vol. 2022, p. 8323962, Apr. 2022.

[6] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, 2013.

[7] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," *CoRR*, vol. abs/1406.2199, 2014. [Online]. Available: http://arxiv.org/abs/1406.2199

[8] S. H. X. P. L. H. Jindong Wang, Yiqiang Chen, "Deep learning for sensor-based activity recognition: A survey," *Pattern Recognition Letters*, vol. 119, no. 1, pp. 3–11, 2009.

[9] J. Ortiz Laguna, A. G. Olaya, and D. Borrajo, "A dynamic sliding window approach for activity recognition," in *User Modeling, Adaption and Personalization*, J. A.

Konstan, R. Conejo, J. L. Marzo, and N. Oliver, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 219–230.

[10] O. Banos, J.-M. Galvez, M. Damas, H. Pomares, and I. Rojas, "Window size impact in human activity recognition," *Sensors (Basel)*, vol. 14, no. 4, pp. 6474–6499, Apr. 2014.

[11] O. Banos, M. Damas, H. Pomares, I. Rojas, M. Tóth, and O. Amft, "A benchmark dataset to evaluate sensor displacement in activity recognition," 09 2012, pp. 1026–1035.

[12] P. Pandey, A. P. Prathosh, M. Kohli, and J. Pritchard, "Guided weak supervision for action recognition with scarce data to assess skills of children with autism," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 463–470, 04 2020.

[13] S. Boughorbel, J. Breebaart, F. Bruekers, I. Flinsenberg, and W. Kate, "Child-activity recognition from multi-sensor data," 01 2010.

[14] D. Liciotti, M. Bernardini, L. Romeo, and E. Frontoni, "A sequential deep learning application for recognising human activities in smart homes," *Neurocomputing*, vol. 396, 04 2019.

[15] S. Suzuki, Y. Amemiya, and M. Sato, "Enhancement of child gross-motor action recognition by motional time-series images conversion," in *2020 IEEE/SICE International Symposium on System Integration (SII)*, 2020, pp. 225–230.

[16] ——, "Enhancement of gross-motor action recognition for children by cnn with openpose," in *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1, 2019, pp. 5382–5387.

[17] Y. Zhang, Y. Tian, P. Wu, and D. Chen, "Application of skeleton data and long Short-Term memory in action recognition of children with autism spectrum disorder," *Sensors (Basel)*, vol. 21, no. 2, Jan. 2021.

[18] F. Olalere, V. Brouwers, M. Doyran, R. Poppe, and A. A. Salah, "Video-based sports activity recognition for children," in *2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2021, pp. 1563–1570.

[19] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[20] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," *CoRR*, vol. abs/1902.09212, 2019. [Online]. Available: http://arxiv.org/abs/1902.09212

[21] V. Bazarevsky and R. E. Ivan Grishchenko, "On-device, real-time body pose tracking with mediapipe blazepose," Google Research, Tech. Rep., 2020.

[22] Y. Kong and Y. Fu, "Human action recognition and prediction: A survey," *CoRR*, vol. abs/1806.11230, 2018. [Online]. Available: http://arxiv.org/abs/1806.11230

[23] B. Y. Lee, L. H. Liew, W. S. Cheah, and Y. C. Wang, "Occlusion handling in videos object tracking: A survey," *IOP Conference Series: Earth and Environmental Science*, vol. 18, p. 012020, feb 2014. [Online]. Available: https://doi.org/10.1088/1755-1315/18/1/012020

[24] F. Angelini, Z. Fu, Y. Long, L. Shao, and S. M. Naqvi, "2d pose-based real-time human action recognition with occlusion-handling." *IEEE transactions on multimedia.*, vol. 22, no. 6, pp. 1433–1446, June 2020. [Online]. Available: http://dro.dur.ac.uk/31054/

[25] C.-C. Huang and M. Nguyen, "Robust 3d skeleton tracking based on openpose and a probabilistic tracking framework," 10 2019, pp. 4107–4112.

[26] V. Bazarevsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang, and M. Grundmann, "Blazepose: On-device real-time body pose tracking," *CoRR*, vol. abs/2006.10204, 2020. [Online]. Available: https://arxiv.org/abs/2006.10204

[27] R. Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "Lsd: A fast line segment detector with a false detection control," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, pp. 722–32, 04 2010.

[28] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[29] K. Lasinger, R. Ranftl, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," *CoRR*, vol. abs/1907.01341, 2019. [Online]. Available: http://arxiv.org/abs/1907.01341

[30] S. Farabi, H. H. Himel, F. Gazzali, M. B. Hasan, M. H. Kabir, and M. Farazi, "Improving action quality assessment using resnets and weighted aggregation," *CoRR*, vol. abs/2102.10555, 2021. [Online]. Available: https://arxiv.org/abs/2102.10555

[31] H. Pirsiavash, A. Torralba, and C. Vondrick, "Assessing the quality of actions," 10 2014.