Making the dataset and evaluating the model for fault estimation in human pose estimation

Abstract

Background Human pose estimation, or skeleton detection, has been a topic of research for many years. With the advancement of hardware, it has become viable to apply human pose estimation in real-time applications such as games. However, human pose estimation is a difficult task that is prone to errors or faults, especially in complicated situations, such as cramped environments. These errors might cause human-computer interaction to be hampered.

Objective The goal of this thesis is to understand what causes the problems and design exercises which deliberately cause common errors and to capture them within a dataset and label the entries accordingly. Finally, a preliminary method is developed to detect the errors or faults caused by human pose estimation using the dataset.

Methods In the scope of this thesis, a method is developed that is capable of capturing and labelling multi-modal data, **F**ault **E**stimator for **S**keleton **D**etection **Data** processor (FESDData). I define four different problem sets with increasingly detailed areas, ranging from body-wise fault estimation to joint-wise fault estimation. Using the dataset that is recorded and labelled using FESDData, FESDDataset, I developed one model for each problem set, which aims to detect if an error occurs at different levels, **F**ault **E**stimator for **S**keleton **D**etection **Model** (FESDModelv1 and FESDModelv2). While version one of the models uses custom feature extraction the second version uses transfer learning.

Results The result of my research shows that the collected and labelled data is not enough to create a generalised model for error detection. The best results are achieved when detecting errors for the individual body halves and for the individual joints using FESDModelv2. The rest of the problem sets achieve bad results which indicates overfitting.

Conclusion While the results seem to be underwhelming, the research shows that the approach is viable and that two of the 8 preliminary models seem to be successfully trained. Additionally, FESDData and consequently FESDDataset, would not only prove useful for the advancement of further FESDModels but could also be applied to general human pose estimation problems, such as action recognition. With the future improvement of FESD-Data and Model, the way that human pose estimation is evaluated could be changed by adding an extra level of verification.

Contents

List of Figures

List of Tables

Acknowledgement

I would like to thank my supervisor Prof. Dr. Erwin Bakker, who guided me throughout this thesis and allowed me to use a depth camera for the thesis. I would like to thank SilverFit for giving me the necessary insight into common errors in human pose estimation, and for allowing me to use their hardware for the collection of the dataset. Additionally, I would like to thank my friends and family who have been patient enough to listen to me talk about my thesis for the past 14 months. And I would especially like to thank my partner Maria Xiberras, who has supported me throughout this thesis, even if that meant many restless weekends and delaying our plans for longer than expected.

Chapter 1

Introduction

Human pose estimation, or skeleton detection, aims at detecting the pose or skeleton of a person based on visual information only. It finds many applications, from games to medical applications[15, 6, 19]. However, human pose estimation is a difficult task that is prone to errors[23]. These errors can be caused by different factors, such as the environment, the camera, and the person. These errors can cause the joints of the pose to be in the incorrect position or missing. This can cause the pose estimation to be incorrect, and therefore, the human-computer interaction will be hampered. The goal of this thesis is to develop a method that allows for the detection of these errors such that the pose information can either be improved on them or handled accordingly.

One of the limiting factors for human pose estimation is the user itself. If the posture is not as the model expects then errors might occur. This is especially true for applications that are designed for rehabilitation and exercise purposes. In these applications, the user is often elderly and has limited mobility, as is the case for games developed by SilverFit[1]. SilverFit is a serious gaming company that develops games for rehabilitation with a special focus on geriatric patients. In their games, SilverFit uses human pose estimation to detect the pose of the player and use it to control the game to make exercise more enjoyable while promoting activity. This thesis is written in collaboration with SilverFit and aims at improving human pose estimation by giving feedback on the errors in their games.

For some exercises designed by SilverFit and other companies, human pose estimation is not sufficiently reliable to create an enjoyable experience for patients in every environment. Especially sedentary exercises often cause the human pose estimation to fail or to produce unreliable results.

In this chapter, the research question that is answered in this thesis is discussed. Furthermore, the procedure with which the research question is answered is laid out. Finally, the fundamentals of human pose estimation and the metrics that are used to analyse the results are discussed, and related work is presented.

1.1 Applications of human pose estimation

Human pose estimation finds application in many different fields. In this chapter, some of the most common applications of human pose estimation are mentioned.

[1] `https://www.silverfit.com/en/`

Gaming and entertainment Gaming and entertainment are one of the most common applications of human pose estimation. Games can use human pose estimation in a way that makes the interaction between humans and computers very natural. One of the systems that kickstarted the use of depth cameras in games was the Xbox Kinect by Microsoft. The Kinekt used a depth camera to track the movement of the player and used this information to control the games.

However, since human pose estimation requires space and a good camera, it is not as widespread as conventional input peripherals.

Autonomous Driving Autonomous driving has been in development ever since humans replaced horses with cars[14]. However, the development of autonomous driving has been very slow. The main reason for this is that autonomous driving requires a lot of information about the environment. This information is usually provided by sensors that are installed in the car. However, sensors alone do not always suffice. In some cases, cars need to be able to estimate the pose of a human to make a decision. The posture of a human can be used to determine the action and therefore the future trajectory of the person.

Animation To exactly emulate human movements in animation, animators can either manually move the joints of a digital skeleton or they can use real human[2] actors to provide the movement for them. The manual creation of realistic movement is oftentimes very time-consuming and also error-prone. Therefore, animators often use real human actors to provide the movement for them. This provides animators with a skeleton and movement which is accurate and does not include human error. In large production studios, this is often done with motion capture or MoCap.

MoCap is a technique that uses cameras to capture the movement of a human actor. The cameras are placed around the actor and record the movement of the actor. The actor usually wears a suit that is covered with markers. These markers are used to determine the position of the actor. To reduce the amount of occlusion of the markers a large number of cameras are used. This allows the cameras to capture the movement of the actor from different angles. However, this also increases the price of development. In cases where MoCap is not a viable option, animators can use human pose estimation to estimate the pose of a human actor using cheaper RGB cameras or RGB-D cameras.

Security Another application of human pose estimation is the detection of anomalous behaviour. The detection of such behaviour can be used to prevent security incidents in public spaces.

Healthcare To aid in the rehabilitation of patients as well as to improve the quality of life of elderly people, human pose estimation can be used to detect the pose of a person and provide feedback on the pose of the person during exercises.[6]

1.2 Research question

A major problem with human pose estimation is that it is not possible to tell if the joints of the pose are faulty or not. Using faulty joints can decrease the efficacy of the training

[2]Or animal

effect of the developed games and can make them very frustrating to use and develop. A joint is considered faulty if it is not in the incorrect position, i.e. the distance from the actual position is greater than a chosen threshold, or if it missing from the skeleton.

In this thesis, first, the first research question that I am going to address is; "What problems occur during human pose estimation and how can they be reproduced?". The aim is to find which problems are the most common and which joints are most affected by the errors. Additionally, exercises are designed to emulate different scenarios with ever-increasing difficulty. This will help give an overview of the issues related to human pose estimation and help develop ways to detect these issues.

Once the issues and error sources that occur during human pose estimation are analysed, the next research question is addressed; "How can I capture a dataset of multiple modalities to analyse if the previous observations about problems during pose estimation are correct?". This allows for the creation of a dataset that can be used to train a model to detect faulty joints.

Using the dataset that is developed I train a model. Using the model I answer the research question posed; "Is it possible to train a model using multi-modal data to determine if a joint is faulty?"

1.3 Process Pipeline

In the scope of the thesis, three steps were defined that are necessary to achieve the goal of the thesis. These steps are the analysis of the problem, the data collection and processing, and the model development.

During the analysis of the problem, different factors are discussed which might influence the human pose estimation. Based on these factors exercises are designed to emulate different scenarios with different difficulties. This is discussed in section 2.1.

The data collection and processing are discussed in section 2.2. The data is captured using a custom tool, FESDData, that was developed for this thesis. The tool allows capturing predefined exercises and labelling them with error labels. The data is then processed and stored in a custom format.

Using the collected dataset a data loader and model are derived. The data loader performs the necessary preprocessing steps and the model is trained to detect faulty joints. This is discussed in section 3.

Finally, the model is evaluated on a test set and the results are discussed in section 4.

1.4 Related Work

1.4.1 Human Pose Estimation

RGB Pose Estimation

While OpenPose developed Hand Pose[22] and also Multi-Person Human Pose Estimation [4], $REPLACE_OUR$ main focus lies on their most recent pose estimator [3] and their CNN network [24]. Openpose uses affinity fields. The affinity fields are a set of 2D Gaussian distributions that are used to estimate the pose of a human. The affinity fields are used to estimate the pose of a human by estimating the probability of a joint being in a certain location. The probability of a joint being in a certain location is calculated by summing the probability of the joint being in that location for each of the Gaussian distributions.

RGBD Pose Estimation

1.4.2 Action Detection

1.4.3 Fault Detection

Human Pose Estimation using Iterative Error feedback. [5]

1.5 Fundamentals

In this section, the fundamentals that are necessary to understand the thesis are discussed. These fundamentals are human pose estimation, the fundamentals of machine learning, and the evaluation metrics that are used to evaluate the results.

1.5.1 Human pose estimation

There is a wide variety of how humans can interact with computers. Ranging from early punch cards to touch screens, the methods have evolved to be more natural and intuitive. In recent years, the use of cameras has become more popular, since they require no physical contact with a computer and therefore allow users to seamlessly interact with an application.

Pose visualisation

Different methods to capture the pose of a human have been developed. Depending on the usage of the pose estimation, different methods are used to visualise the pose of a human. These visualisations may provide different information about the pose of a human.

There are mainly three different ways the human pose can be visualised. The first and most basic way is to visualise the pose as a kinematic representation, in which a "skeleton" with bones and joints is used to represent the pose of a human. The skeleton is made up of joints, which are connected by bones. The number of joints and bones can vary, but the most common skeleton is made up of 17 joints and 16 bones, as can be seen in figure 1-1. The joints are usually labelled with a number, which is used to identify the joint in the output of the pose estimation. The representation of a joint in the data varies, but it is usually a 2D or 3D point in space. In some cases, an additional joint representation is provided with a keypoint orientation that enables the clear representation of all degrees of freedoms joints have[9]. Additionally, in some cases, especially if the human pose was estimated using a neural network, a confidence rating or score is added which can be used to determine the reliability of the joint.

The second way to visualise a human pose is by using a 2D silhouette or 2D rectangles and shapes. These methods are also called contour-based methods. An example of contour-based methods was introduced by Yunheng Liu[17]. Contour-based methods are often used in combination with a skeleton representation. The skeleton is used to determine the location of the joints, while the contour is used to determine the shape of the body. This is useful when the skeleton is not able to determine the shape of the body, for example, when the person is wearing a coat or a jacket. This is also used for some games developed by SilverFit.

Finally, the third way to represent a human pose is with a three-dimensional volume. This volume may be simple cylindrical shapes or a body mesh. A body mesh is a 3D

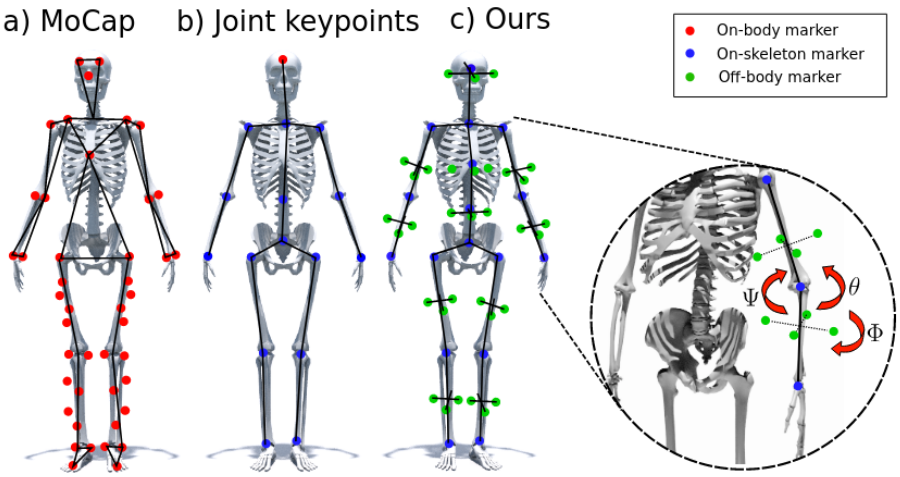


Figure 1-1: Example of a human pose captured with different methods. (a) A captured skeleton using MoCap (b) A traditional human skeleton representation. (c) A pose representation that includes the orientation of the joints as well as the bones as presented by Martin Fish and Ronald Clark[9]

representation of the body, which is made up of vertices and triangles. The three different representations of the human pose can be seen in figure 1-2.

Data sources

The data that is acquired influences the way the human pose can be estimated. The most common data source is RGB images or videos. RGB videos are any videos that are captured with normal cameras that record the colour of the scene. The provided data can be either from a video or a stream of data or a still image. There is a large number of datasets that can be used to train and test poses estimation algorithms from RGB data. Some of the most common datasets are the MPII Human Pose Dataset[2], the COCO dataset[16], and HumanEva-I dataset[21].

Additionally, some datasets are captured with depth cameras. Depth cameras are cameras that can record, in addition to the colour channels, the depth of the scene. Different methods to acquire depth data have been developed. The most commonly used depth cameras emit an infrared light pattern to measure the depth of a scene. This depth information can be used to improve the accuracy of the pose estimation. Some of the most common datasets that are captured with depth cameras are the MRI dataset[1], and the Human3.6M dataset[12].

Finally, there are also methods of human pose estimation that use point clouds. Point clouds are a collection of points in space, which can be used to represent the shape of an object. Each point in the point cloud represents a point in the environment. The point cloud is created using the depth information of a scene and the intrinsics of a camera, i.e. the field of view and the focal point of the camera. A point cloud is a reconstruction of the real-world scene. Point clouds are often used in combination with RGB data. Some of the most common datasets that are captured with depth cameras are the SMMC-10 dataset[10], and the EVAL dataset[11]. However, any RGBD dataset can be used to train and test point cloud-based pose estimation algorithms if the camera intrinsics and extrinsic, such as the

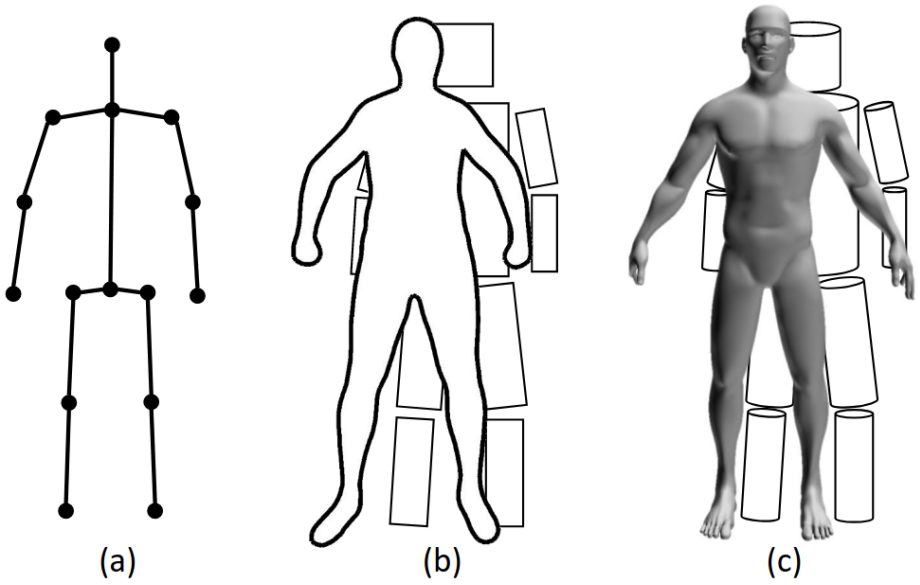


Figure 1-2: Different representations of the human pose. (a) Kinematic representation. (b) Contour representation. (c) 3D volume representation. [7]

horizontal and vertical field of view, the focal point, and the depth units are known. With the knowledge of these parameters, the depth information can be converted to a point cloud and if the RGB data is in line with the depth data, the individual points can be coloured accordingly.

1.5.2 Convolutional Neural Networks

Convolutional neural networks are a type of neural network that is commonly used for image classification and object detection. Convolutional neural networks are a type of feed-forward neural network that uses convolutional layers. Convolutional layers are layers that use a convolutional operation to extract features from the input. The convolutional operation is a mathematical operation that is used to extract features from an image.

In most cases, the features that are produced by multiple convolutional layers are forwarded into fully connected layers. These layers learn the features based on the input data. The fully connected layers are usually followed by a softmax layer. The softmax layer is used to calculate the probability of each class. The class with the highest probability is then used as the prediction of the network.

Since convolutional neural networks require a lot of data to be trained properly, transfer learning is applied in some cases. Transfer learning is a technique that uses a pre-trained model to extract features from the input data. The pre-trained model is usually trained on a large dataset, such as ImageNet[8]. The pre-trained model is then used to extract features from the input data. These features are then used to train a new model. This allows the new model to be trained with less data and therefore less time. Additionally, the pre-trained model is usually trained on a large dataset and therefore the features that are extracted are more general and can be used for different tasks.

Table 1.1: The confusion matrix as a metric for binary classification. The results of the prediction (PRED), left, and the actual ground truth (GT), top.

PRED \GT	TRUE	FALSE
TRUE	TP	FP
FALSE	FN	TN

1.5.3 Evaluation metrics

To rightfully evaluate the performance and accuracy of a model, different metrics are used. In this section, some metrics that can be used to evaluate the performance of a model are discussed.

Confusion Matrix

A confusion matrix is a graph which represents the false positives and the true negatives in a single graph. The confusion matrix can be seen in table 1.1. The confusion matrix is useful to visualise the performance of a model.

Percentage of positive Guesses

The percentage of positive guesses is an indicator of the performance of a model. For example, if the dataset is unbalanced, the accuracy might be sufficiently good when only guessing for the majority class. In those cases, the percentage of positive guesses would either be 0% or 100%. Therefore, the percentage of positive guesses is used to determine if the model is biased toward a specific class. The equation for the percentage of positive guesses can be seen in equation 1.1. In this thesis, the percentage of positive guesses is sometimes referred to as $\frac{p}{p+n}$ as it is the ratio of the number of positive guesses p to the total number of guesses $p+n$.

$$\text{Percentage of positive guesses} = \frac{\text{Number of positive guesses}}{\text{Number of guesses}} = \frac{tp+fp}{tp+fp+tn+fn} \quad (1.1)$$

Accuracy, Precision, Recall and F1-Score

The accuracy of a prediction is the ratio of the number of correct predictions to the total number of predictions. The accuracy is calculated using equation 1.2. The accuracy is a good indicator of the performance of a model if the dataset is balanced. However, if the dataset is unbalanced, the accuracy might be sufficiently good when only guessing for the majority class.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Number of predictions}} = \frac{tp+tn}{tp+fp+tn+fn} \quad (1.2)$$

The precision of a prediction is the ratio of the number of correct positive predictions to the total number of positive predictions. The precision is calculated using equation 1.3.

$$\text{Precision} = \frac{\text{Number of correct positive predictions}}{\text{Number of positive predictions}} = \frac{tp}{tp+fp} \quad (1.3)$$

The recall of a prediction is the ratio of the number of correct positive predictions to the total number of positive samples. The recall is calculated using equation 1.4.

$$\text{Recall} = \frac{\text{Number of correct positive predictions}}{\text{Number of positive samples}} = \frac{tp}{tp + fn} \tag{1.4}$$

Finally, the F1-Score is the harmonic mean of the precision and the recall. The F1-Score is calculated using equation 1.5. The F1-Score is a good indicator of the performance of a model if the dataset is unbalanced since it takes both the precision and the recall into account.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \cdot tp}{2 \cdot tp + fp + fn} \tag{1.5}$$

Cohen's kappa coefficient

Cohen's kappa coefficient is an inter-rater reliability measure[18]. This means that two different raters are compared. In the case of a binary classification, the formula for Cohen's Kappa can be seen in equation 1.6. In the case of prediction, the two raters are the ground truth and the prediction.

$$\kappa = \frac{2(tp \cdot tn - fn \cdot fp)}{(tp + fp)(fp + tn) + (tp + fn)(fn + tn)} \tag{1.6}$$

If Cohens Kappa is less than 0, there is no agreement between the two raters. If the kappa score is less than 0.5 there is a slight agreement. A score of more than 0.8 is considered almost perfect. However, the "exact" values vary from application to application.

Chapter 2

FESDData - Data Acquisition and Labelling of FESDDataset

One of the most important aspects of machine learning is the data acquisition. In this thesis, a custom tool is developed which is capable of capturing and labelling multi-modal data. The tool is called **F**ault **E**stimator for **S**keleton **D**etection **Data** processor (FESDData). It is capable of capturing RGB-D data from multiple different RGB-D cameras as well as pose data calculated by Nuitrack. Nuitrack which was developed by 3DiVi Inc[1]. Nuitrack was chosen as the pose estimation backbone since it provided the support of multiple different camera models as well as the ability to record the RGB-D data and pose data simultaneously.

Additionally to Nuitrack, FESDData is capable of calculating the pose information using OpenPose after the recording is done. This offers future support for other pose estimation backbones. However, OpenPose is not used in the scope of this thesis.

FESDData is designed to be easy to use, by anyone and to require minimal setup or tweaking. It allows for the rapid capture of RGB-D datasets with the pre-labelling of multiple different recordings. The parameters can be adapted to capture datasets for many different applications, e.g. action detection, where human pose estimation is used to identify an action.

For this thesis, exercises have been derived which are designed to emulate different scenarios with different difficulties. These exercises are discussed in section 2.1. The labelling of errors in the data requires domain-specific knowledge and is therefore done manually. The resulting dataset, FESDDataset, is discussed in section 2.4. It consists of the RGB-D data as well as the pose data and the labels for the errors for each of the joints in the pose.

In this chapter, the approach to the design of the exercises is discussed by showing the difficulties of human pose estimation. Furthermore, the data acquisition and labelling process is discussed. Finally, the features of the dataset are presented.

2.1 Challenges in Human Pose Estimation

In this chapter, possible faults and difficulties that occur during human pose estimation are discussed. These difficulties are caused by different factors, such as the environment, the camera, the person, and the software.

These difficulties are important to understand, as they can cause the joints to be in the

[1] `https://nuitrack.com/`

incorrect position or missing. Leading to an incorrect estimation of the pose estimation, and therefore, the human-computer interaction will be hampered.

2.1.1 Environment

The environment in which the data is recorded is crucial to the quality of the pose estimation. The environment can be split into multiple categories, which are discussed in the following sections. The mentioned difficulties depend on how the method itself works. For example, if a method uses RGB data, the background might be an issue, whereas if a method uses depth data, the lighting is more of an issue.

Background

The background of the scene can cause difficulties in the human pose estimation process. Methods using RGB cameras might have difficulties detecting the difference between the foreground and the background. In RGB-D-based methods, the background can cause depth ambiguities which might even prevent the human from being detectable, especially if the background is reflective, or too close to the user.

Lighting

While RBG cameras are to some extent insensitive to the amount of incoming light, RGB-D cameras are highly influenced by the lighting. Most RGB-D cameras use infrared light to determine the depth of the scene, some use a pattern of infrared light that is projected onto the scene and distorted by physical objects and some use the time-of-flight method to determine the depth of the scene. The issue that arises with infrared light is that it is also emitted by the sun. This means that light emitted by the sun can interfere with the infrared light emitted by the camera. This can cause the depth of the scene to be incorrect or missing in parts with a high intensity of sunlight.

To reduce the effect of the sunlight, the camera can be placed in a room with curtains or blinds. This will reduce the amount of sunlight that enters the room and therefore reduce the effect of the sunlight on the camera. Since lighting is hard to perfectly reproduce without a controlled environment, the lighting is not varied throughout the experiments. Therefore, all of the experiments were conducted in a room without any natural light or during the night.

However, if a method heavily relies on RGB data for the pose estimation, eliminating any form of light is not a valid option since then the data that can be gathered by RGB cameras in low light environments is limited and may result in a wrong pose.

Objects

The objects in the scene may cause issues in the human pose estimation process if they either occlude the user or are too close to the user. Occlusion can cause inaccurate or missing joints. Whereas objects that are too close to the user can cause joints to move to these objects instead.

Chair

If the exercise is performed in a sitting position the chair might influence the accuracy of HPE. For example, wheelchairs pose a problem in some estimators but a significant part of

users who use pose estimation for rehabilitation games are using wheelchairs due to health conditions. Furthermore, bulky chairs that go higher than the head may prevent accurate head detection and silhouette estimation, which is also sometimes used instead of the pose.

2.1.2 Camera

The two main difficulties that can occur with the camera setup are the camera position and the camera angle. Additionally, the camera itself can make the pose estimation process more difficult.

Distance

The distance of the camera to the user affects the quality of the pose estimation in multiple ways. Firstly, if the user is too close to the camera, the camera might not get the complete body into the frame. The legs and arms might be off the frame preventing them to be estimated properly. Additionally, depth cameras can only detect the depth for a specific range, so if a user is too close they might not be visible to the depth camera.

However, if the user is too far away from the camera, the person might be too small to be detected reliably. Additionally, depth cameras operate at different depth ranges. If a user is too far away from the camera it will not be visible to the depth camera. This range varies from camera to camera and depends on the method of detection.

Angle

When considering angles the main focus lies on pitch and roll. For the experiments, the yaw is assumed as fixed and directed facing the user, such that the user is in the centre. The camera is set up such that there is no or minimal roll as all applications ensure that the camera is not tilted to the side.

The pitch may introduce or reduce the occlusion of joints by other joints. It also influences which area is best for human pose estimation. The pitch of a camera depends on what the main application is. For example, if the legs are not considered then the pitch could be used to focus more on the upper body.

Resolution

The resolution of the camera, or rather the resolution of the image captured by the camera influences the information that can be gathered about the human and therefore influence the performance of the pose estimation. Also here the application and specific range are important for choosing the right resolution. If a user is far away a higher resolution is needed to detect the pose reliably.

2.1.3 Person

Finally, one of the main error sources of human pose estimation is the person. The person can cause difficulties in the human pose estimation process by moving, wearing specific clothes, or having a different body posture. Body posture is of special importance for SilverFit since SilverFit specialises in games for rehabilitation and elderly people. Elderly people have different body postures than the average person, which can cause difficulties in the human pose estimation process.

Clothes

As mentioned earlier, most RGB-D cameras use infrared light to determine the depth of the scene. This means that the clothes of the user can cause more or less absorption of light and therefore influence the detected depth. This can cause the joints to be detected in the wrong position or not at all. This is especially the case for dark clothes, as they absorb more light than light clothes.

Furthermore, bulky clothes or skirts and dresses may influence the pose, since the exact position of the legs is not visible.

Training Equipment

To make exercises more challenging some physiotherapists use additional training equipment. This could be weights that are held in the hand or weights that are attached to the ankles. These weights change the outline of the body and therefore influence the pose estimation.

Exercises

Finally, the most important factor is the exercise that is carried out. In this section, both exercises that are easy to detect as well as some exercises that are difficult to detect are proposed.

These exercises might not be the most realistic, but they represent common issues with pose estimation in a reproducible manner. Furthermore, these exercises are not too difficult to perform, which makes them suitable for testing the pose estimators. The difficulty rating of the exercise might not reflect the difficulty of the exercise for the user, but it does reflect the difficulty of the exercise for the pose estimator.

The exercises are numbered according to the difficulty. An example of the naming of the exercises can be seen here:

$$\underbrace{E}_{\text{Exercise}} - \overbrace{2}^{\text{Difficulty}} \; . \; \underbrace{02}_{\text{Exercise Id}}$$

The different difficulties are numbered as follows:

0. Trivial
1. Easy
2. Medium
3. Hard

Trivial Exercises Trivial exercises are exercises that are easy to detect and are therefore good for testing the pose estimators. The exercises do not involve any movement, which makes detecting the joints easier. The two trivial exercises can be seen in table 2.1. Example images of the exercises can be seen in figure 2-1.

Table 2.1: The two Trivial Exercises, E-0.00 and E-0.01.

Exercise	Short Description	challenge
E-0.00	Arms hanging to the side	This exercise is very easy since no part of the body is occluded by any other part. Additionally, the joints are not moving.
E-0.01	Arms extended to the side	The arms are no longer in a natural position but still not occluded and no joint is moving

(a) E-0.00

(b) E-0.01

Figure 2-1: The two trivial exercises. (2-1a) Arms hanging to the side. (2-1b) Arms extended to the side.

Easy Exercises Easy exercises are essential for creating a good baseline of how the pose estimators should work. The exercises include no self-occlusion and are recorded in a standing position, which is generally the easiest position to detect. The easy exercises can be seen in table 2.2. Example images of the exercises can be seen in figure 2-2.

Table 2.2: The Easy exercises, E-1.00 through E-1.03. The easy exercises include both standing and sitting exercises.

Exercise	Short Description	Challenge
E-1.00	Raising the arms to the side	Now the arms are no longer stationary. Still, there is no Occlusion.
E-1.01	Raising the arms to the front	The arms might occlude themselves and part of the body.
E-1.02	Raise the knees to the chest	The legs now occlude parts of the hip.
E-1.03	Sit in a chair motionless	Sitting positions are more challenging to detect than standing positions since there is now a chair in the background and the body occludes itself.

Medium Exercises Exercises which are performed in a seated position are harder to detect since parts of the body are occluded. Medium exercises focus on exercises, which are performed in a seated position. The medium exercises can be seen in table 2.3. Example images of the exercises can be seen in figure 2-3.

Difficult Exercises Difficult exercises are exercises that are performed in a standing position and involve leg movement. Leg joints are harder to detect than arm joints and therefore pose a greater challenge for pose estimators. These exercises will be in a seating

(a) E-1.00 (b) E-1.00 (c) E-1.01 (d) E-1.01 (e) E-1.02 (f) E-1.03

Figure 2-2: The four easy exercises. (2-2a, 2-2b) Raising the arms to the side. (2-2c, 2-2d) Raising the arms to the front. (2-2e) Raising the knee to the chest. (2-2f) Sitting in a chair motionless.

Table 2.3: The Medium exercises, E-2.00 through E-2.03. The medium exercises are all in a sitting position.

Exercise	Short Description	Challenge
E-2.00	Raising the arms to the side while sitting	Now the arms are no longer stationary. With added difficulty since the user is now sitting.
E-2.01	Raising the arms to the front while sitting	The arms might occlude themselves and part of the body. Additionally, the person is now sitting down
E-2.02	Crossing the arms in front of the chest while sitting	The arms now occlude large parts of the upper body.
E-2.03	Raising the knee to the chest while sitting	More parts of the upper body are occluded and not seen by the camera.

position and with a difficult posture, such as leaning forward. The difference in posture aims at creating a realistic representation of real-world exercises.

Tölgyessy et al. found that facing away from the camera decreases the accuracy of HPE due to self-occlusion. [23] Therefore, some of the exercises will be performed facing away from the camera. The difficult exercises can be seen in table 2.4. Example images of the exercises can be seen in figure 2-4.

2.2 Data acquisition

Different modalities were captured to create a dataset that reproduces a real-world application of human pose estimation for RGB-D cameras. The different modalities are RGB data, depth data, and joint data. While the Nuitrack SDK offers to capture the data from the RGB-D cameras and the joint data, the recorded files cannot, at the time of writing, be read without using the Nuitrack SDK. Therefore, FESDData, a custom RGB-D+HPE capturing and labelling tool was developed.

FESDData has two main uses which are interlocked. Firstly, it allows capturing predefined, as well as custom, exercises repeatedly automatically making the capturing experience when capturing many different actions more comfortable. Secondly, it allows reviewing and labelling the captured data with error labels. The lightweight nature of FESDData allows it to seamlessly capture both the RGB-D stream and the skeleton data at the same time while ensuring a stable fast framerate. The dataset that is used by FESDModel was captured at

(a) E-2.00 (b) E-2.00 (c) E-2.01 (d) E-2.02 (e) E-2.02 (f) E-2.03

Figure 2-3: The four medium exercises. (2-3a, 2-3b) Raising the arms to the side while sitting. (2-3c) Raising the arms to the front while sitting. (2-3d, 2-3e) Crossing the arms in front of the chest while sitting. (2-3f) Raising the knee up to the chest while sitting.

Table 2.4: The Difficult exercises, E-3.00 through E-3.02. The difficult exercises are all in a standing position.

Exercise	Short Description	Challenge
E-3.00	Bowing toward the camera	The head is often used as an anchor point for the skeleton as it is quite stable. When bowing forward the head is no longer easily detectable.
E-3.01	Raising the knee leaning forward	Leaning forward increases the difficulty since the position is not natural
E-2.02	Raising the knee leaning forward while sitting and facing away from the camera	Facing away from the camera makes it more difficult to detect the pose and induces more occlusion.

a framerate of 30 frames per second. The interface of FESDData can be seen in figure 2-5. On the left side of the image the interface can be seen and on the right, there is an example for data labelling of Exercise E1-02.

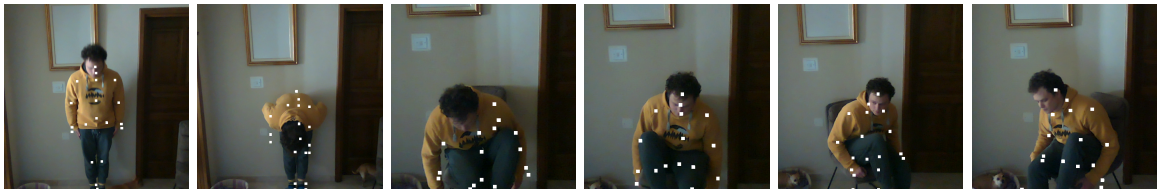
2.3 Data layout

As mentioned earlier, the data is made up of multiple different streams or modalities. There are two separate visual streams, the RGB stream, and the depth stream, as well as the estimated human pose, the time stamps, and the recording metadata.

The visual streams are normalised and combined into a single file. OpenCV is used to store the RGB and the depth data into a single matrix and after the stream into a single file per frame. The RGB data is normalised to have values between 0 and 1, whereas the depth data is stored in meters.

The pose that is estimated by Nuitrack, as well as the error labels, are stored in a separate JSON file. The separate frames are stored in a list of frames. Each frame contains a list of all people that were detected. Each person contains a list of joints as well as an error label. Each joint is stored with real-world 3D coordinates which are stored in meters. These real-world coordinates are labelled x, y, and z. Additionally, the 2D projection and depth of the joint are stored in image coordinates and meters for the depth. The 2D projection is labelled u and v and the depth is labelled d.

The error label is an integer which is the error id specific for joint and skeleton er-



(a) E-3.00 (b) E-3.00 (c) E-3.01 (d) E-3.01 (e) E-3.02 (f) E-3.02

Figure 2-4: The three difficult exercises. (2-4a 2-4b) Bowing toward the camera. (2-4c, 2-4d) Raising the knee leaning forward. (2-4e, 2-4f) Raising the knee leaning forward while sitting and facing away from the camera. Some of the exercises contain errors in the sample.

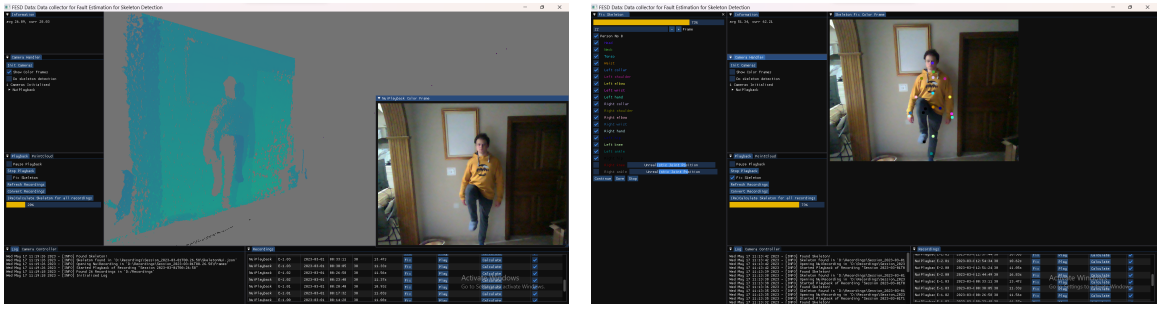


Figure 2-5: A screenshot of the user interface with its two main components. On the left side, the user interface during streaming and recording can be seen and on the right side an example of data labelling.

rors. The errors corresponding to the error ids are explained in section 2.3.1. Finally, the timestamp of each frame is stored.

2.3.1 Data labeling

A large part of the data preparation is the labelling of the data. The data is labelled with error labels. Two areas can be labelled as erroneous. First, there are skeleton errors that occur when the pose estimator detects a human in places where there are no humans. Second, there are joint errors that occur when the pose estimator detects a joint in the wrong place.

For example, the estimator might label the left foot as the right foot. This is a common error, especially when the body parts are close to each other. An estimator might also not detect a joint at all. This might be caused by occlusion, be it by another joint, an object, or by the image border. Most applications avoid the last cause for occlusion by defining a minimum distance from the camera and specific camera placement to ensure that the user is always fully in view.

In the data, no error is denoted with 0. If a joint is not detected at all, it is labelled with 1. If a joint is detected in the wrong place that is outside of the body, or somewhere where there should not be a joint, it is labelled with 2. If a joint is detected in the approximate position of where another joint should be then it is labelled with 3. If the whole skeleton is in the wrong position it can be labelled as faulty and subsequently, every joint will be labelled with 2.

Implicitly, this creates two simpler general labels, either a joint is faulty, i.e. the error

label is 1, 2, or 3, or it is not faulty, i.e. the error label is 0.

2.4 Dataset

Multiple iterations of the recording process were run to find the best possible setup, which reduces the light interference as much as possible and which offers the best results with the resources at hand. The camera setup that is used by SilverFit was reproduced. At SilverFit the camera is mounted at 175cm above the floor. The camera that is used has an accelerometer which was used to adjust the camera angle relative to the ground. The camera is angled downward at a 70° angle. To record the dataset an Intel Realsense L515 camera was used. Additionally, for the preliminary analysis of errors an Orbbec Astra+ and a Microsoft Kinect v2 were used.

2.4.1 Problem Sets

Different problem sets are used to create different versions of the model. The problem sets are defined by the number of objects that are considered and are defined as erroneous. There are four different problem sets. The first problem set is the *Joint* problem set. In this problem set, each joint is considered as a single object. The second problem set, the *Body Part* problem set, is to consider each body part, i.e. the individual arms, legs, torso, and head. The third problem set is the *Half Body* problem set. In this problem set the upper and the lower body are the areas that are considered. Finally, only the whole body is considered in the *Full Body* problem set.

To determine the threshold at which each area is considered faulty, the distribution of joints with errors was calculated for each of the areas was calculated and the 50th percentile was measured. It was found that when considering the full body as an area, the body is considered faulty if more than two joints in the pose are faulty. When considering the lower and upper body, one or more and more than two faulty joints are needed for the area to be considered faulty. For the body parts to be considered faulty one joint within the specific part needs to be faulty, except for the right and left leg, where two joints need to be faulty.

2.4.2 Distribution of Errors

An important aspect of the dataset is the structure and distribution of data and their labels. In total, all 13 exercises mentioned in section 2.1.3 were recorded twice. Each recording session consists of exactly 300 frames.

When multiple persons are detected one person might be incorrectly detected in the background. While analysing the data the person that is not labelled as faulty is selected whenever possible. If a person is labelled as faulty each joint is marked as in an unrealistic position.

An important factor in how well a model can be trained on data is the balance of the dataset. In this case, the dataset is balanced by the error labels.

Full Body Error Distribution

In Figure 2-6 the error distribution of the Full body problem set can be seen. It can be observed that the distribution of errors is reasonably equal with a difference of 10%.

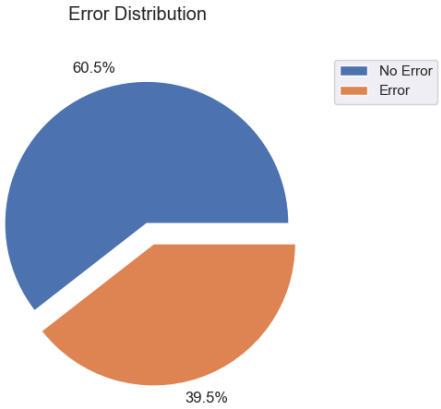


Figure 2-6: The distribution of Errors of the full body problem set.

Half Body Error Distribution

Figure 2-7 shows a discrepancy between the error distribution of the lower body (65.4% Errors) and the upper body (32.4% Errors). The upper body is generally more stable and less error-prone than the lower body.

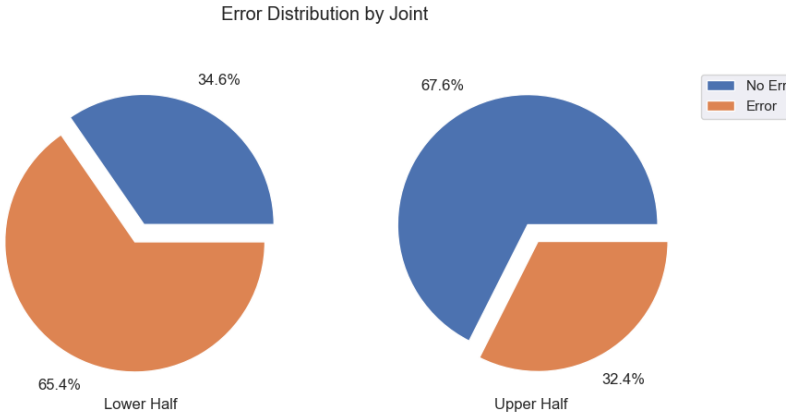


Figure 2-7: The distribution of Errors of the half body problem set.

Body part Error Distribution

The error distribution of each body part is shown in figure 2-8. It can be observed that the errors of the body parts are individually very unbalanced. The left arm is the most error-prone body part with 24.1% of the joints being faulty. The torso is the least error-prone body part with 10.3% of the joints being faulty. This again underlines the assumption that the upper body is more stable than the lower body.

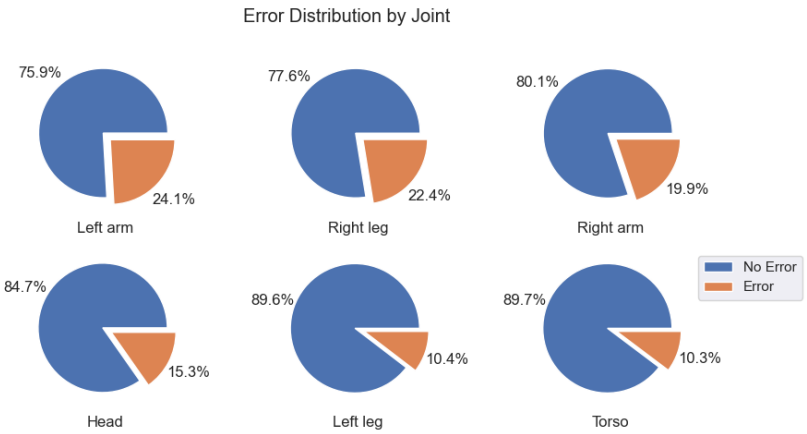


Figure 2-8: The distribution of errors of the body part problem set.

Joint Error Distribution

Figure 2-9 shows that the major part of the errors that occur are errors with the label 2, i.e. the joint is detected in the wrong place. The second most common error is label 1, i.e. the joint is not detected at all. The least common error is label 3, i.e. the joint is detected in the approximate position of where another joint should be.

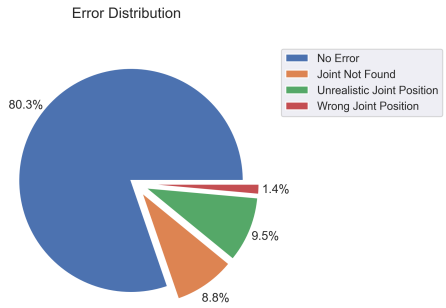


Figure 2-9: The distribution of each error class.

Chapter 3

FESDModel - Model development

While there could be multiple approaches to fault estimation, a deep-learning approach has been chosen. The reason for this is that deep learning has shown to be very successful in many different fields, such as image classification, object detection, and image segmentation. The reason for this is that deep learning can learn the features of the data by itself, without the need for manual feature extraction.

Other possible solutions could be to use rule-based systems, which use inverse kinematics, or use frame-to-frame joint comparison to detect discrepancies, however, these are quite limited and might result in either too many false positives or false negatives. Furthermore, these rules, such as the frame-to-frame joint comparison, are not always applicable to all types of movements, and therefore might not be able to detect all types of faults in all cases.

3.1 Model architecture

To solve the problem of error estimation, two different model architectures are devised. The first model architecture, FESDModelv1, is a deep convolutional neural network that takes the RGB, Depth, and Joint data as input. This model can be seen in figure 3-1.

The second model architecture, FESDModelv2, utilises transfer learning to extract the features of the input data using a pre-trained model. FESDModelv2 can be seen in figure 3-2. Both models are trained to predict the error labels for each joint. The error labels are the same as the error labels used in the data labelling. The error labels are explained in section 2.3.1. The fully connected layers of both networks use ReLU activation functions.

While FESDModelv1 uses the data as it is stored in the dataset, FESDModelv2 merges the data into a single RGB image. This is done using the feature extractor, which is trained on RGB images. The data is merged by assigning each modality to a channel in an RGB image. The RGB image is transformed into greyscale and assigned to the red channel, the depth image is scaled to a value between 0 and 255 and assigned to the green channel, and the joint coordinates are assigned to the blue channel.

In total eight models were developed and trained. Four models were trained using FESDModelv1 and four models were trained using FESDModelv2. Each of the models for each version corresponds to one problem set, i.e. one model for each problem set is trained using FESDModelv1 and one model for each problem set is trained using FESDModelv2. The problem sets are explained in section 2.4.1.

Each of the four different models for each version of FESDModel has a different output vector. The full-body problem set has an output vector of size two. Each of the two values

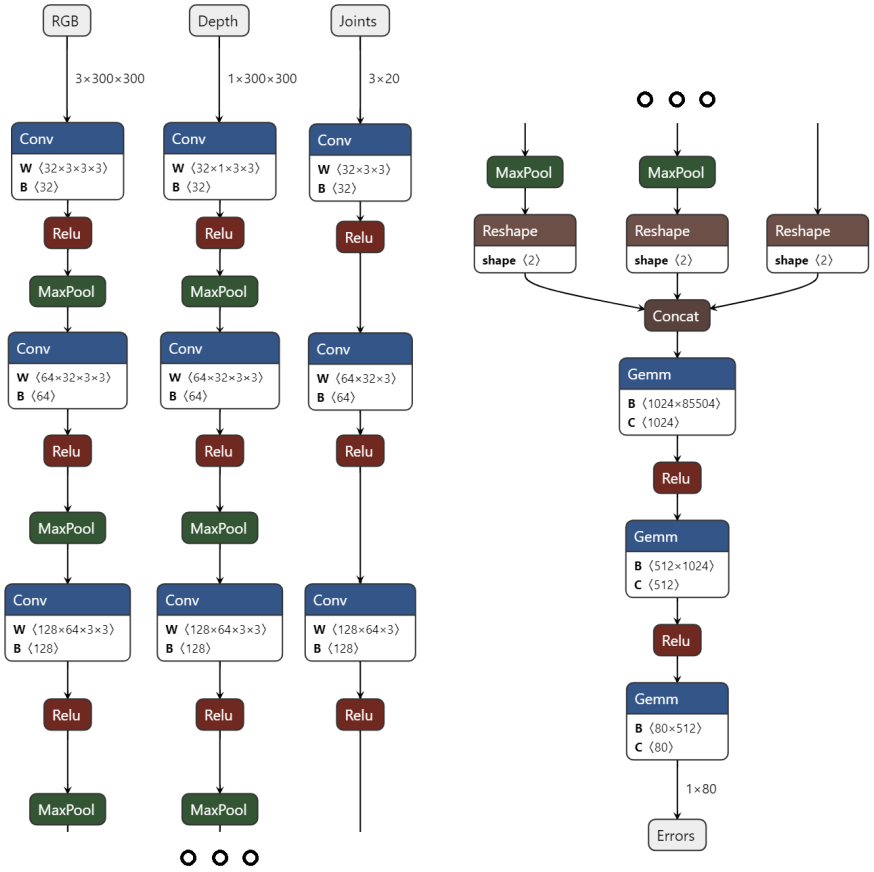


Figure 3-1: Original FESDModel architecture with three different inputs; RGB, Depth and Joint data. After three convolutions the three streams are concatenated to be passed into three fully connected layers with ReLU activation functions. In this example network, the model calculates the joint problem set, therefore the output is a 1D 80 tensor of multi-object, one for each joint, i.e. 20, multi-class, one for each error class, i.e. 4, values.

represents a boolean for "No Error" or "Error". To find if the result is an error or no error, the confidence rating and the most likely result is calculated using soft max(see section 1.5). For every other problem set, the output vector is the number of Error classes, i.e. two, "No Error" and "Error", for the half-body and body part problem set, and four, "No Error", "Joint not Found", "Joint in a Wrong Position", "Joint at a Different Joint Position", for the joint problem set, times the number of objects. For example, the body part problem set has two error classes and six objects or areas, Head, Torso, Left and Right arm, and Left and Right Leg, therefore the output is a vector with 12 values. The output vector is split into the number of objects and the softmax is calculated for each object.

To choose a network that is used by FESDModelv2 as a feature extractor, multiple different networks have been compared, which can be seen in figure 3-3. One of the target applications of the model is to be used in a real-time application so that error handling can be conducted. Consequently, a lightweight model which does not impact the performance much is preferred. Therefore, the models are compared by the number of floating-point operations (FLOPS) to their Accuracy on ImageNet-1K. Table 3.1 shows the top 5 models

according to their accuracy and performance.

Table 3.1: The top 5 models according to their accuracy and performance. The models are sorted by their GFLOPS and their Top-5 Accuracy[2]. The models are EfficientNet V2 S, ConvNeXt Base, EfficientNet B6, Swin V2 B, and EfficientNet V2 M.

Weight	Acc@1	Acc@5	Params	GFLOPS
EfficientNet V2 S	84.228	96.878	2.15×10^7	8.370
ConvNeXt Base	84.062	96.870	8.86×10^7	15.360
EfficientNet B6	84.008	96.916	4.30×10^7	19.070
Swin V2 B	84.112	96.864	8.79×10^7	20.320
EfficientNet V2 M	85.112	97.156	5.41×10^7	24.580

EfficientNet v2 was chosen since it proved to be the most performant while being the most accurate of the networks that were analysed. In particular, the small variant with 2.15×10^7 Parameters and a Top-1 Accuracy of 84.228%. The model architecture can be seen in figure 3-2.

3.2 Data preparation

To successfully train FESD three steps are taken before training can begin, data augmentation, data merging, and data balancing. The data augmentation is done to ensure that the model is robust to different variations in the data. The data merging is done to combine the different modalities into a single tensor. The data balancing is done to ensure that the model is not biased toward any particular error label.

The finished data preparation pipeline for FESDModelv1 and FESDModelv2 can be seen in figure 3-4.

The images that are stored by *FESDData* are inherently the same size. The joints, however, are stored within a JSON file containing the coordinates of each joint in 2D and 3D. To further process the 2D joint data is drawn on an image that has the same dimensions as the RGB and Depth image for the FESDModelv2. For FESDModelv1 the position of each joint relative to the waist joint is passed into the network.

3.2.1 Data augmentation

Four different augmentations are applied to the data to generalise the data. The first augmentation is flipping the data. The RGB image, the depth image, and the joint image are flipped horizontally. Furthermore, the ground truth data is flipped, as labels refer to the left or right side of the body, which would no longer coincide with the data that is passed into the network.

Additionally, the images are cropped at random while keeping the positions of the joints and a margin around the joints visible. This ensures that the model is robust to different positions of the user in the image.

Finally, at random Gaussian noise is applied to the RGB image and the depth image. This further improves the robustness of irregular data.

The augmentations can be seen in figure 3-4 where they are applied to a sample frame from the dataset.

3.2.2 Data balancing

In the ordinary case, human pose estimation is not meant to produce faulty results. In the selected exercises it is aimed to produce faulty results. However, this does still not produce a balanced dataset. In section 2.4, the statistics of the dataset are shown. Most notably for the problem set *Half* and *Full*, in figures 2-7 and 2-6, where the error label *No Error* is overrepresented it can be seen that the dataset is imbalanced.

To balance the dataset, frames are sampled using a Weighted Random Sampler for each batch of the training. The weights for the samples are calculated based on the occurrence of the error labels in the dataset. While only considering the whole body as a single object, the calculation of the weights is simple. For each frame, the error label is counted and the inverse of the count is used as the weight for the frame. This ensures that the model is not biased toward any particular error label by oversampling the frames which contain an error.

However, for the other problem sets the calculation of the weights is more complex. In the other problem sets each frame contains an error for each area, e.g. when considering the Half-Body problem, the upper and lower body 2 errors. To successfully balance the dataset for each area four weights would need to be created and balanced, i.e. the upper and lower body have an error the upper body is faulty and the lower body is not, etc. This would oversample some frames while undersampling others. In the other problem sets this is far more visible. Therefore, it was decided to consider the sum of erroneous joints per frame as a balancing factor. This means that frames that have the same number of erroneous areas are weighed the same.

3.3 Experimental Setup

To train the models the data has to be passed into the network so that it can predict a value. Based on that value a loss is calculated which is used to adapt the weights of the networks. In the case of FESDModel *cross entropy loss* is used. In cases where the problem set contains more than one problem area, i.e. all problem sets except the full body problem set, the cross entropy loss is calculated for each object or area separately and a summed cross entropy loss is calculated.

Carl F. Sabottke and Bradley M. Spieler showed in their study that a lower resolution of the input images achieves better performances[20]. They found that images that were passed into a CNN which were of a resolution of 300x300 pixels achieved worse results than images with 64x64 pixels. Therefore, the images are scaled to 64x64 pixels.

Both networks are trained using the Adam optimiser, as described by Diederik P. Kingma and Jimmy Ba[13], with an initial learning rate of 0.00005 with learning rate decay.

To improve the performance of the training process Cuda is used. To further speed up the development a batch size of 60 is used.

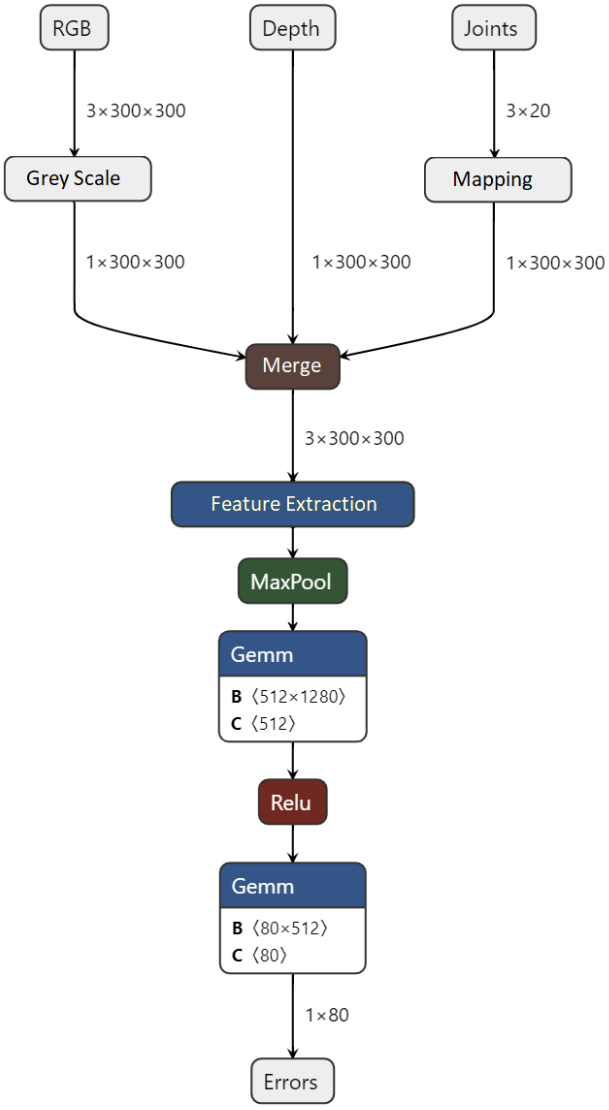


Figure 3-2: FESDModelv2 architecture with transfer learning. The input is merged into a single RGB image and passed into a feature extractor. The feature extractor is a pre-trained EfficientNet v2 S. The output of the feature extractor is passed into two fully connected layers with ReLU activation functions. In this example network, the model calculates the joint problem set, therefore the output is a 1D 80 tensor of multi-object, one for each joint, i.e. 20, multi-class, one for each error class, i.e. 4, values.

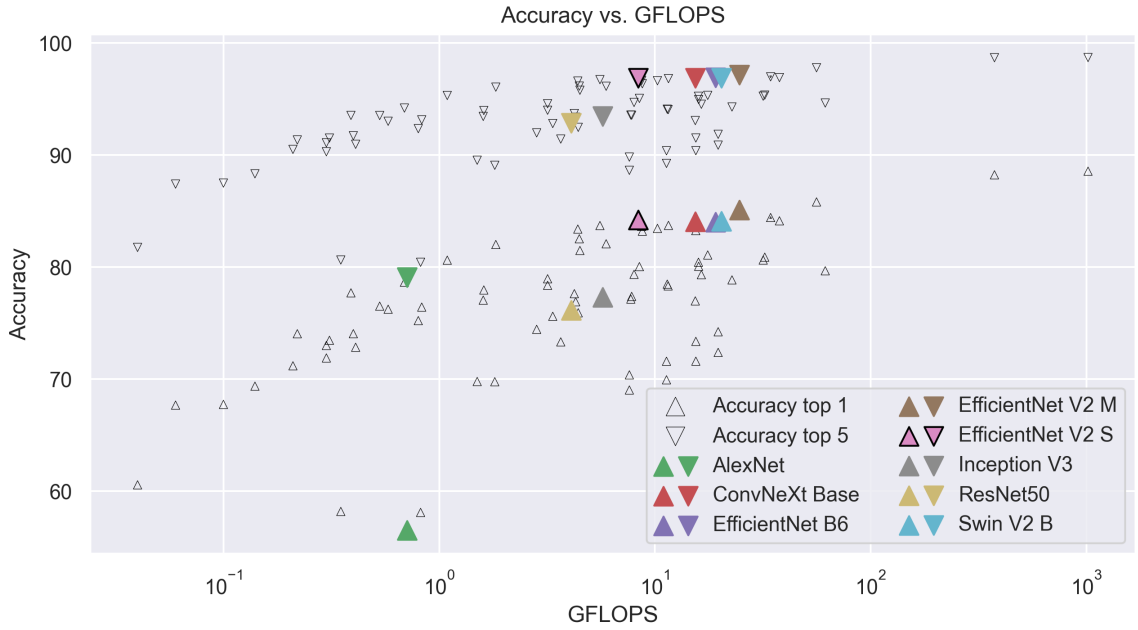


Figure 3-3: The comparison of different networks by their GFLOPS and their Top-5 Accuracy. The models are sorted by their GFLOPS and their Top-5 Accuracy[1]. The models are EfficientNet V2 S, ConvNeXt Base, EfficientNet B6, Swin V2 B, and EfficientNet V2 M. Additionally, AdamNet, ResNet-50 and Inception-v3 are added as a reference.

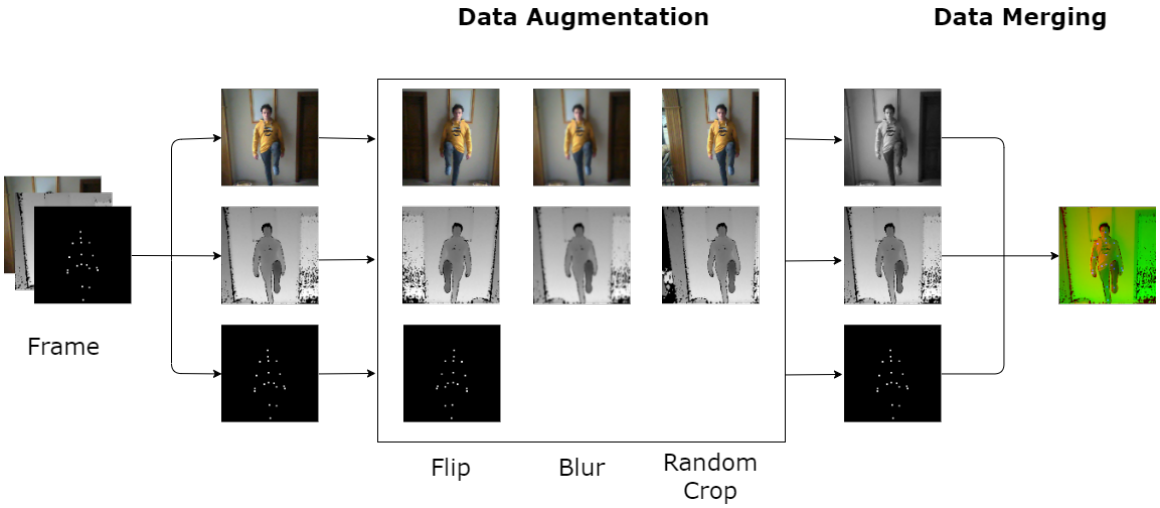


Figure 3-4: The three modalities are separately randomly flipped, Blurred, and Cropped. For FESDModelv1 the separate modalities are passed into the network. For FESDModelv2, the RGB image is transformed into a greyscale image and all three modalities are merged into a single RGB image.

Chapter 4

Results

In this chapter, the results of the experiments are presented. The results for each different problem set is presented seperatly.

4.1 FESDModelv1 Results

The results are better for some stupid reason that I can't explain.

All four models for each training set

4.2 FESDModelv2 Results

4.2.1 Full Body Error Estimation

The first and most general model that was developed was the full-body model. The full body model was trained to give a single error label as an output given an image as an input. The results of the training process of the Full Body model can be seen in figure B-1.

The result of the training is rather disappointing. The testing showed that the model only performs well on the training data. The model does not generalise well to new data. The reason for this is that the model is overfitting. The model is overfitting because the model is too complex for the amount of data that is available. The model has 21.5 million trainable parameters. This is too many parameters for the amount of data that is available. The model is not able to learn the underlying patterns of the data but rather memorises the training data. This is also shown in the training and testing loss. The training loss is decreasing, but the testing loss is increasing. This is a clear sign of overfitting.

Confusion Matrix

The confusion matrix of the full body model is shown in figure 4-1. It can be seen that the model only predicts the error label 0 - No Error. This is because the model is overfitting.

4.2.2 Half Body Error Estimation

The second model that was developed was the half-body model. The half-body model was trained to give two error labels as an output given an image as an input, one for the lower body and one for the upper body. The results of the training process of the Half Body model can be seen in figure B-2.

Figure 4-1: The confusion matrix of the full body error estimation model.

Confusion Matrix

The confusion matrix of the half-body model is shown in figure 4-2. It can be seen that contrary to the full-body model the half-body model varies the error label.

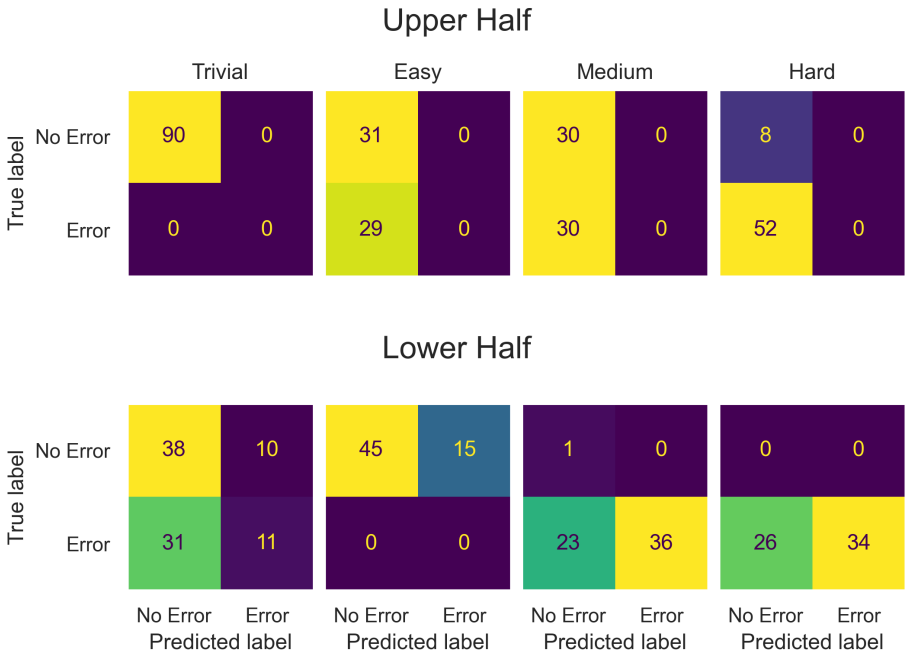


Figure 4-2: The confusion matrix of the half body model.

4.2.3 Body part Error Estimation

The results for the body part error estimation are the same as for the full-body error estimation. The model is overfitting and therefore only predicts the error label 0 - No Error for most cases. The results of the training process of the Body part model can be seen in figure B-4.

Confusion Matrix

The confusion matrix of the body part model is shown in figure 4-3. It can be seen that the model only predicts the error label "No Error" or "Error" for most joints without variation.

This is because the model is overfitting.

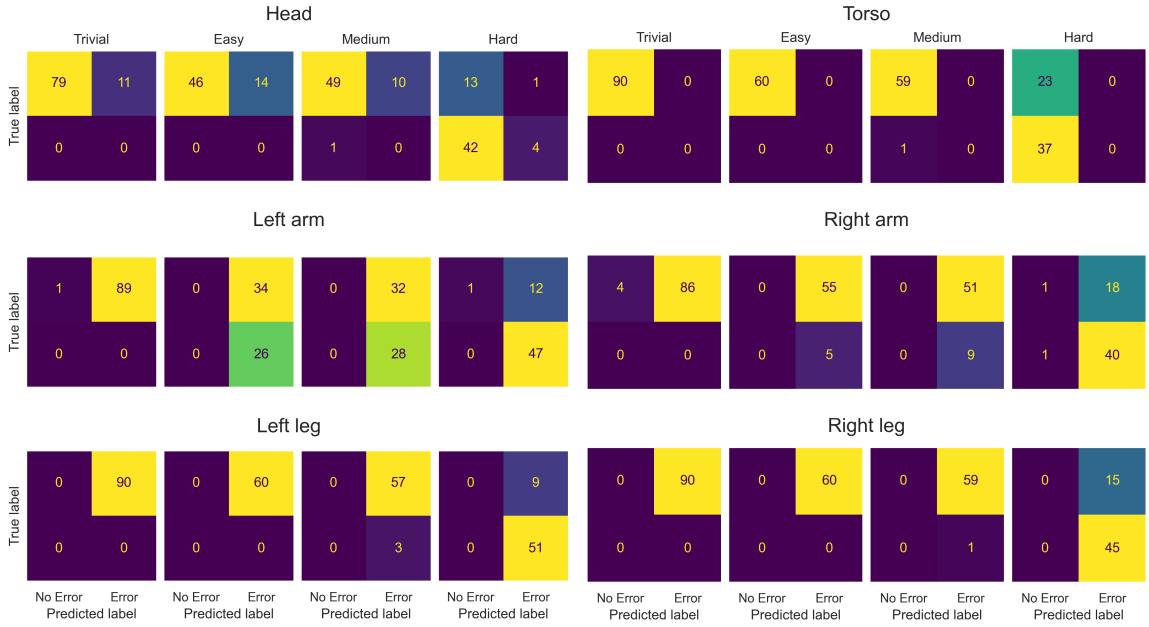


Figure 4-3: The confusion matrix of the body part model.

4.2.4 Joint Error Estimation

The results for the joint error estimation are the same as for the full-body error estimation. The model is overfitting and therefore only predicts the error label 0 - No Error for most joints. The results of the training process of the Joint model can be seen in figure B-5.

Confusion Matrix

The confusion matrix of the joint model is shown in figure 4-4. It can be seen that the model only predicts the error label 0 - No Error for most joints. This is because the model is overfitting.

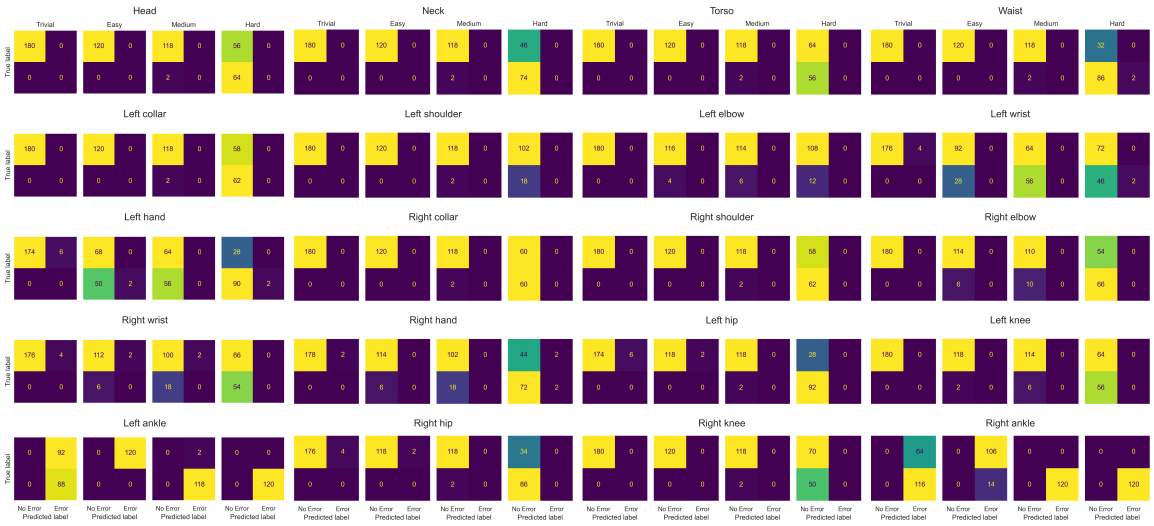


Figure 4-4: The confusion matrix of the joint model.

Chapter 5

Conclusion

In the scope of this thesis, I analysed common problems of human pose estimation and derived exercises which cause issues in a controlled manner. Using these exercises I captured and labelled FESDDataset using a custom tool for multi-modal stream capture and error labelling for human pose estimation. To investigate different problem areas I defined four different problem sets that encompass different abstractions of the problem.

Using the dataset I trained eight models, FESDModel, two for each problem set, version one of the model, FESDModelv1, uses a custom feature extractor, which extracts the features of each modality individually and then combines all features into a single feature vector where it is passed into a fully connected layer. Version two of the model, FESDModelv2, uses a pre-trained EfficientNetv2 to extract the features from a single RGB image. The features are then passed into a fully connected layer.

Two of the eight trained models achieve arguably good results. With an accuracy of 0.89 and an F1-score of 0.64, the best-performing model is FESDModelv2 trained on the joint problem set. The second best performing model is FESDModelv2 trained on the half-body problem set with an accuracy of 0.65 and an F1-score of 0.49. FESDModelv2 is overfitting on each of the other problem sets and only predicts that there is no error.

The code of this thesis is available on GitHub[1]. The repository is divided into two major parts, FESDData, which contains the C++ implementation of FESDData recorded, FESDModel, which contains the implementation of the model, FESDModelv1 and FESDModelv2, as well as the Jupyter notebooks that were used to evaluate the dataset, to train and evaluate the model.

5.1 Future work

To further improve the dataset and the model, FESDData and FESDDataset could be expanded to include the accurate position of the joints in the image. This would allow for the use of the dataset for error correction.

To improve the quality of the FESDModel, more data needs to be collected and labelled in different settings. The current dataset is limited to a single room with a single camera. To improve the model, the dataset needs to be expanded to include different scenes, different pieces of clothing and different camera angles. These additions might prevent the model from overfitting. Additionally, the model could be improved by using a different backbone,

[1] `https://github.com/LeonardoPohl/FESD`

which is not as performant but more accurate, such as EfficientNet, or by using a different loss function, such as focal loss, which.

5.1.1 Possible applications

FESD might find several different areas of application in the future. Firstly, the trained model can be used to assist in developing games and other applications that utilise Human Pose estimation. In its simplest application, it may be used to warn users of possible errors when the skeleton is not detected correctly. In more advanced cases the information provided by the model could be used to attempt to fix joints through joint position interpolation and prediction rather than using the faulty joint. Moreover, multiple human pose estimators could be considered resulting in an overall more robust human pose estimation.

Furthermore, if the model proves to have high accuracy for a specific use-case, it could be used to train a better pose detector in the same way as it is proposed by João Carreira et al. in [5].

The dataset and the dataset recorder may also be used to further the development of FESDModel and it can also find application in other areas such as recording datasets for action recognition. The dataset in and of itself can be used for action detection. The exercises are predefined and can be recorded and automatically labelled by FESDData, thereby making it easy to record large amounts of data without requiring manual labelling.

References

[1] Sizhe An, Yin Li, and Umit Ogras. mRI: Multi-modal 3D Human Pose Estimation Dataset using mmWave, RGB-D, and Inertial Sensors. *CVPR*, 2022.

[2] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2D Human Pose Estimation: New Benchmark and State of the Art Analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[3] 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.

[4] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. In *CVPR*, 2017.

[5] Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. Human Pose Estimation with Iterative Error Feedback. *CVPR*, 2015.

[6] Kenny Chen, Paolo Gabriel, Abdulwahab Alasfour, Chenghao Gong, Werner K. Doyle, Orrin Devinsky, Daniel Friedman, Patricia Dugan, Lucia Melloni, Thomas Thesen, David Gonda, Shifteh Sattar, Sonya Wang, and Vikash Gilja. Patient-Specific Pose Estimation in Clinical Environments. *IEEE Journal of Translational Engineering in Health and Medicine*, 6:1–11, 2018.

[7] Yucheng Chen, Yingli Tian, and Mingyi He. Monocular human pose estimation: A survey of deep learning-based methods. *Computer Vision and Image Understanding*, 192:102897, 2020.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[9] Martin Fisch and Ronald Clark. Orientation Keypoints for 6D Human Pose Estimation. *CVPR*, 2020.

[10] Varun Ganapathi, Christian Plagemann, Daphne Koller, and Sebastian Thrun. Real time motion capture using a single time-of-flight camera. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 755–762, 2010.

[11] Varun Ganapathi, Christian Plagemann, Daphne Koller, and Sebastian Thrun. Real-Time Human Pose Tracking from Range Data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012.

[12] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, jul 2014.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. 2017.

[14] Fabian Kröger. *Automated Driving in Its Social, Historical and Cultural Contexts*, pages 41–68. Springer, 05 2016.

[15] Laxman Kumarapu and Prerana Mukherjee. AnimePose: Multi-person 3D pose estimation and animation. *Graphics*, 2020.

[16] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context, 2014.

[17] Yunheng Liu. Contour Model and Robust Segmentation based Human Pose Estimation in Images and Videos. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 8:1–10, 03 2015.

[18] Mary McHugh. Interrater reliability: The kappa statistic. *Biochemia medica : časopis Hrvatskoga društva medicinskih biokemičara / HDMB*, 22:276–82, 10 2012.

[19] Bernhard Preim and Monique Meuschke. A survey of medical animations. *Computers and Graphics*, 107, 09 2022.

[20] Carl F. Sabottke and Bradley M. Spieler. The Effect of Image Resolution on Deep Learning in Radiography. *Radiology: Artificial Intelligence*, 2(1):e190015, 2020. PMID: 33937810.

[21] Leonid Sigal, Alexandru Balan, and Michael Black. HumanEva: Synchronized Video and Motion Capture Dataset and Baseline Algorithm for Evaluation of Articulated Human Motion. *International Journal of Computer Vision*, 87:4–27, 03 2010.

[22] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. Hand Keypoint Detection in Single Images using Multiview Bootstrapping. In *CVPR*, 2017.

[23] Michal Tölgyessy, Martin Dekan, and Lubos Chovanec. Skeleton Tracking Accuracy and Precision Evaluation of Kinect V1, Kinect V2, and the Azure Kinect. *Applied Sciences*, 11:5756, 06 2021.

[24] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *CVPR*, 2016.

Appendix A

FESDData Appendix

This appendix contains graphs which were not included in the main body of the thesis.

A.1 Distribution of the number of joints with an error

Each problem set, except the joint problem set, is an abstraction from the base data which combines different areas of the pose into separate objects. A threshold is used to determine whether an area is considered as faulty based on the number of joints that are faulty in the pose. To determine this threshold the distribution of joints with errors is calculated for each of the problem areas over all the data and the 50th percentile is picked as the threshold.

Full Body problem set The results of the full body problem set can be seen in Figure A-1. The threshold is at two errors, i.e. if more than two joints in the whole body are faulty, the whole body is considered as faulty.

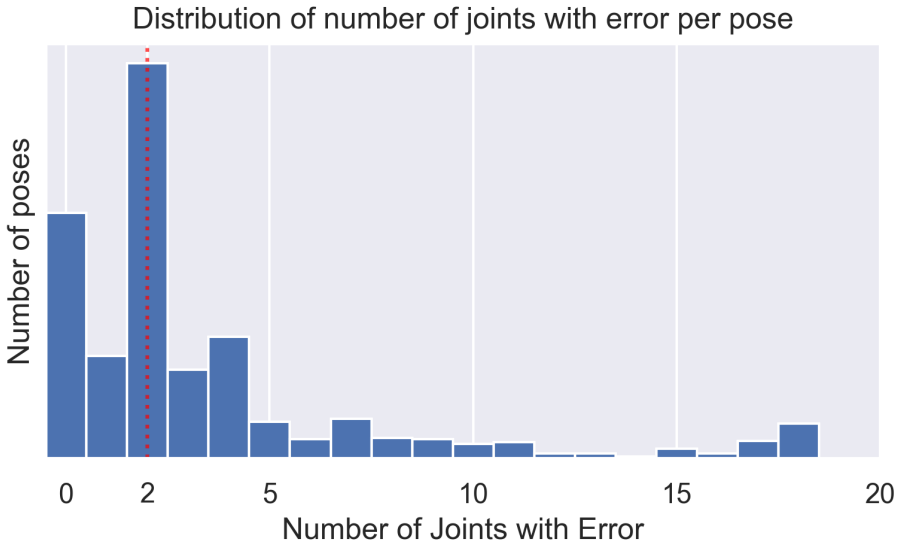


Figure A-1: The distribution of the number of joints with errors in each frame.

Half Body problem set The distribution of joints with errors for both the lower and upper body can be seen in figure A-2. If one joint in the upper body or more than two joints in the lower body are considered as faulty, the upper or lower body is considered as faulty respectively.

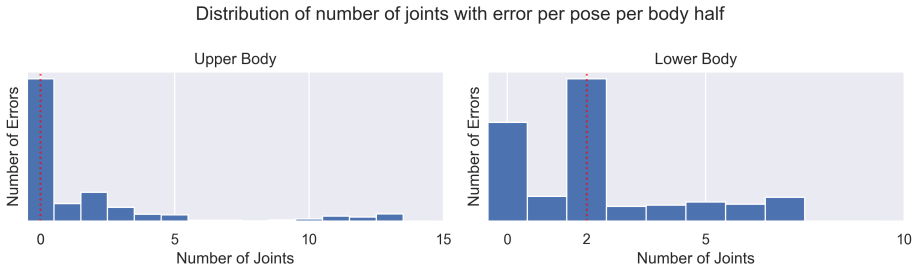


Figure A-2: The distribution of the number of joints with errors in each frame per body half.

Body parts problem set The left and right Legs require two or more faulty joints. Whereas all other body parts require only one or more errors to be considered as faulty.

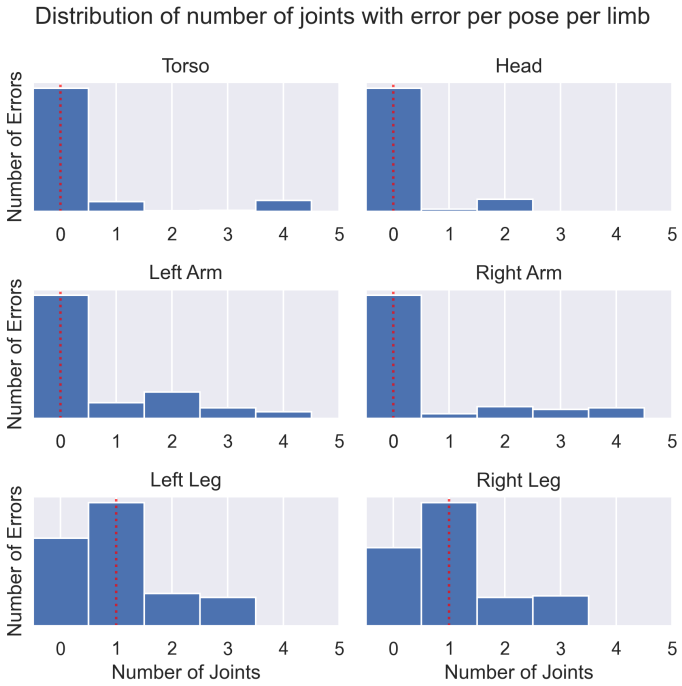


Figure A-3: The distribution of the number of joints with errors in each frame per body part.

A.2 Distribution of Errors

This section contains the different distributions of errors for each of the problem sets.

Full Body Figure A-4 gives an overview of the error distribution by the difficulty of the exercise. The figure indicates that the difficulty of the exercise directly influences the percentage of errors that occur.

Figure A-4: The distribution of Errors of the full body problem set by difficulty.

Half Body The distribution of errors by difficulty for the half body problem set can be seen in figure A-5. Easy exercises seem to be less error-prone when grouping the joints into body halves. This might be caused by the error threshold that was set before.

Body parts The distribution of errors by difficulty can be seen in figure A-6. A less distinct distribution of errors can be seen when grouping the joints into body parts. Only a very small amount of errors occur during trivial exercises. This might be caused by the error threshold that was set before. The majority of errors occurring during trivial exercises are caused by a single faulty joint in the ankle. In Figure A-3 it is shown that the legs have an error threshold of two faulty joints. This means that the legs are considered faulty if there are more than two faulty joints. The ankle is the only joint that is faulty in trivial exercises. This means that the legs are not considered faulty.

Joints In medium exercises, the majority of errors that occur for the joints problem set, occur due to missing joints, rather than joints which are in the wrong position. This is shown in figure A-7.

A clear distinction between error-prone and stable joints can be seen in figure A-8. While in the majority of cases (72.3%) the right ankle is faulty only 4.9% of the left shoulder is faulty. Furthermore, the left and right ankles have a significant number of missing joints,

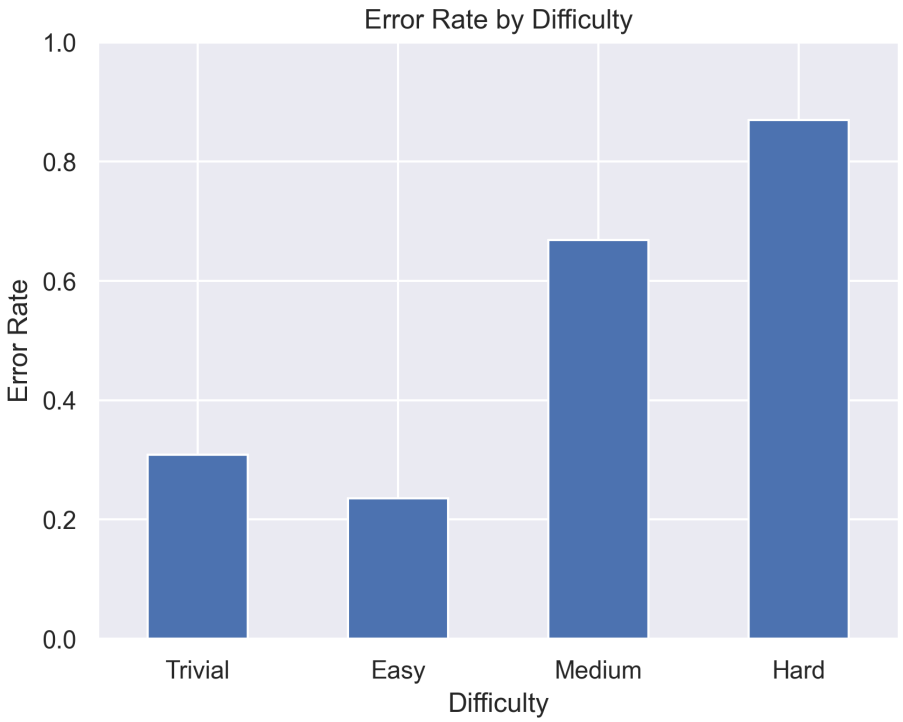


Figure A-5: The distribution of Errors of the half body problem set by difficulty.

contrary to the other joints, which have a more balanced distribution of error classes. This might be caused by occlusion or a cutoff from the image or a confidence which is too low and therefore discarded by Nuitrack.

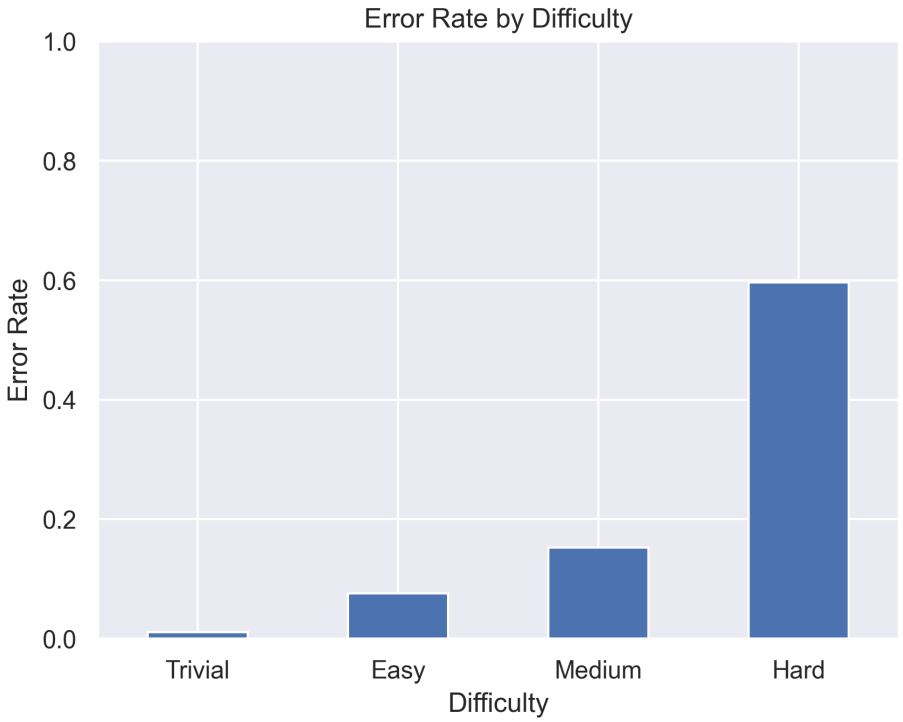


Figure A-6: The distribution of Errors of the half-body problem set by difficulty.

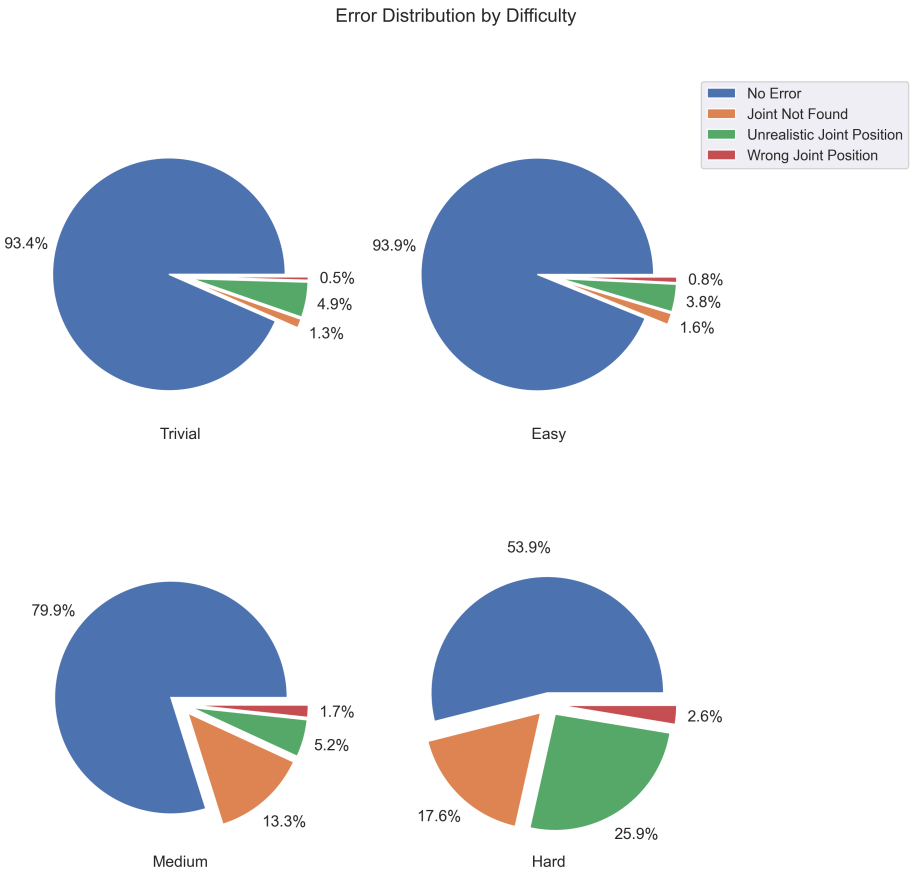


Figure A-7: The distribution of each error class grouped by difficulty.

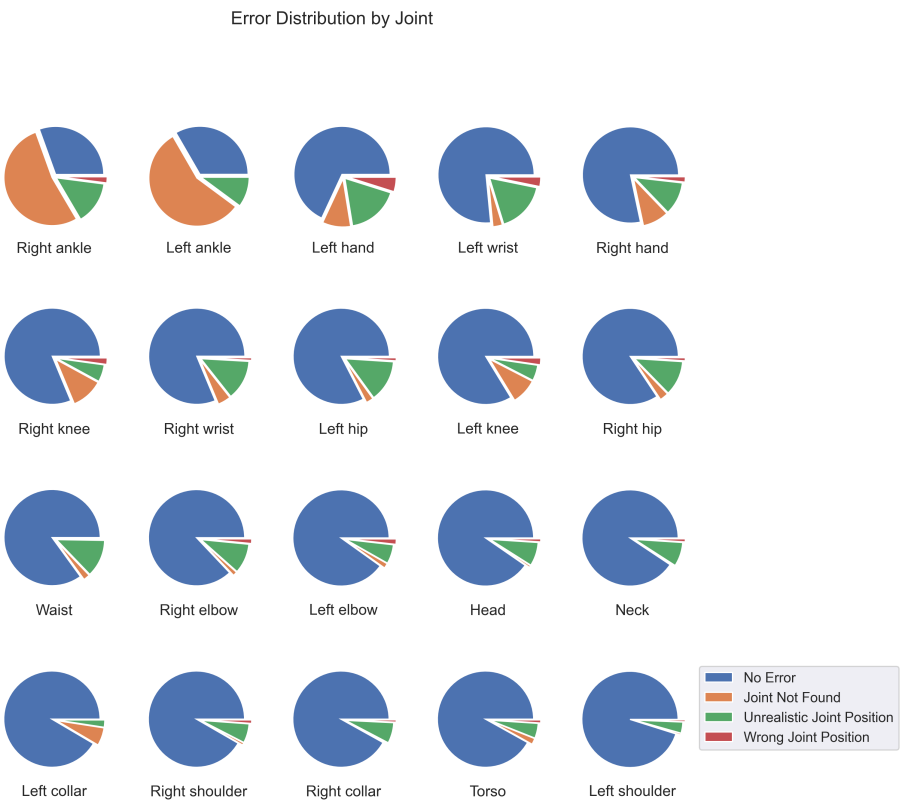


Figure A-8: The distribution of each error class grouped by joint.

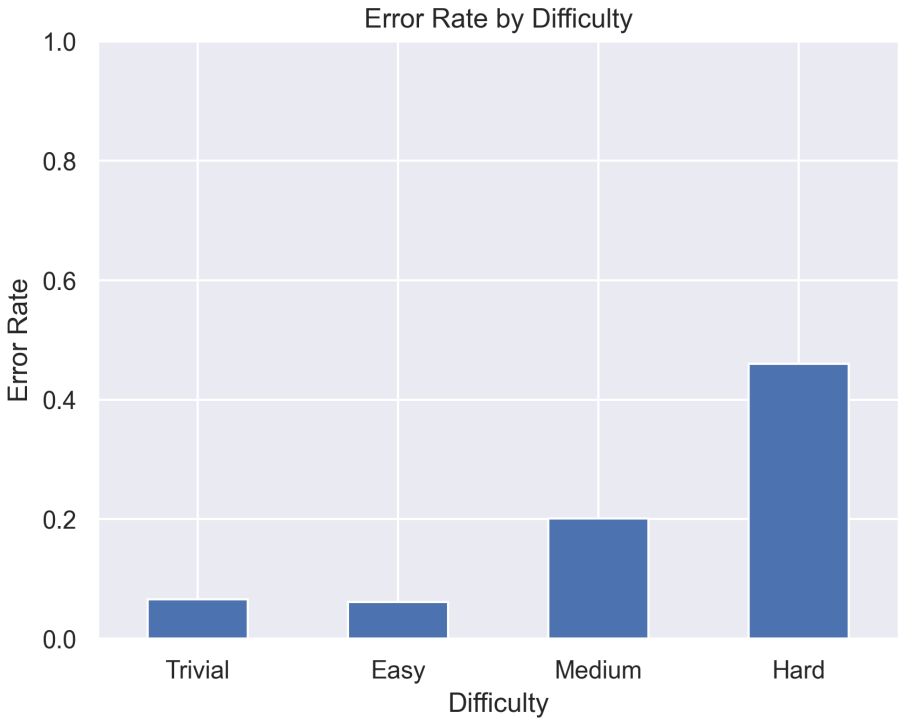
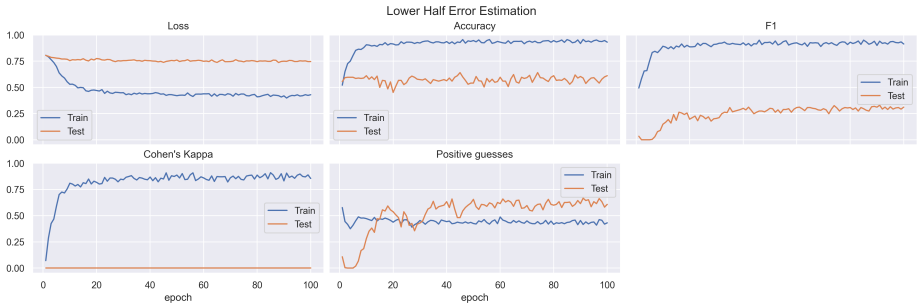


Figure A-9: The distribution of Errors of the joint problem set by difficulty.

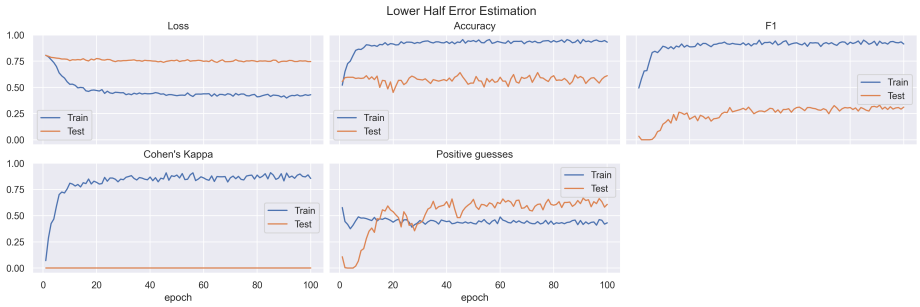
Appendix B

Results

Figure B-1: The training results of the full body model.

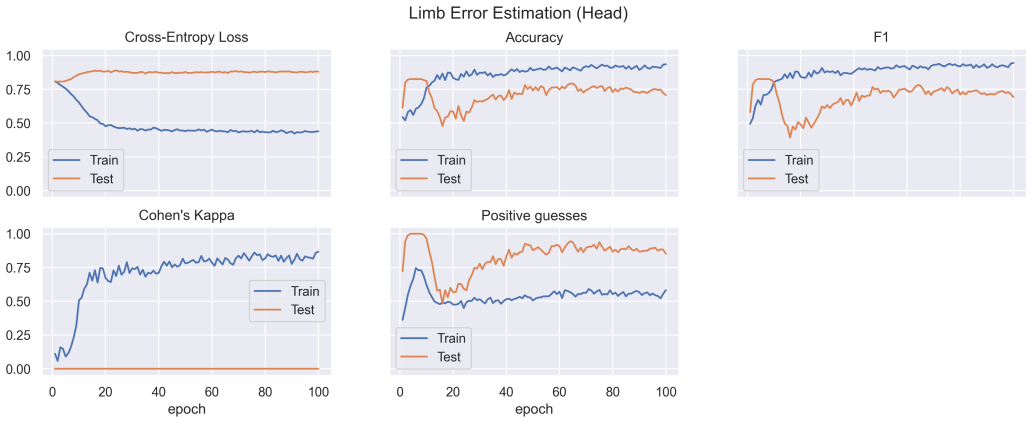


(a) Upper Body Error Estimation

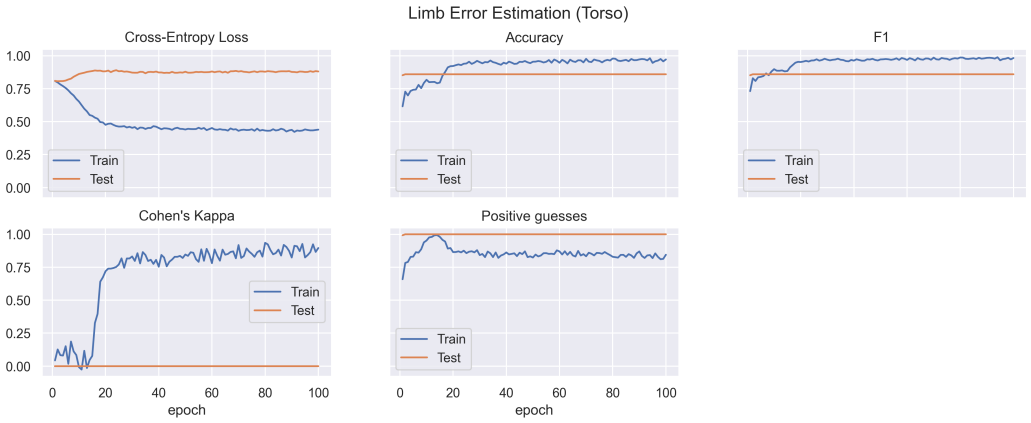


(b) Lower Body Error Estimation

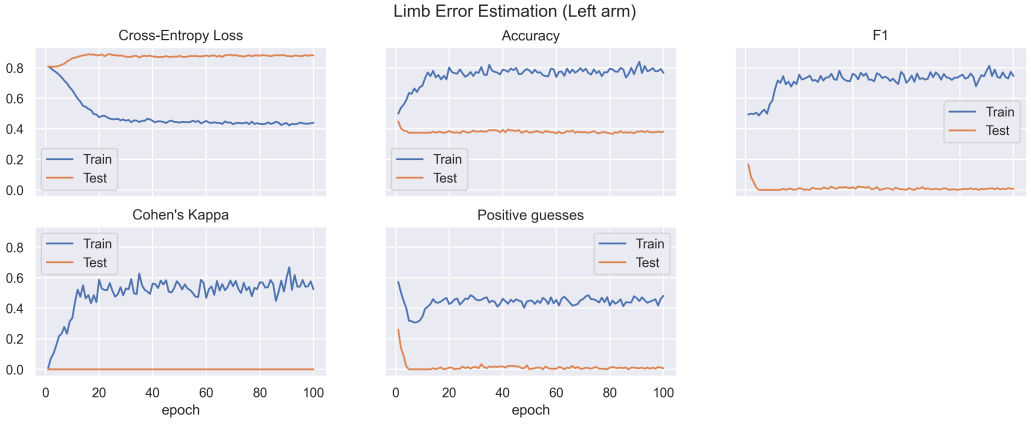
Figure B-2: The training results of the half body error estimation model.



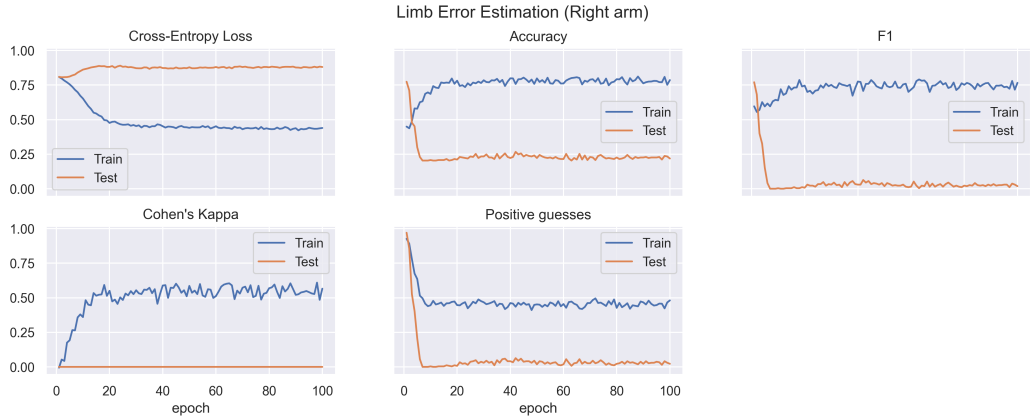
(a) Head Error Estimation



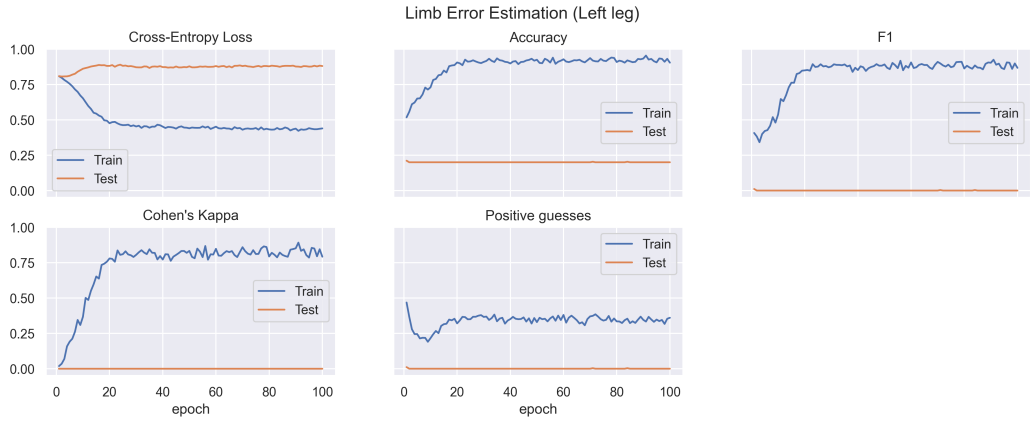
(b) Head Error Estimation



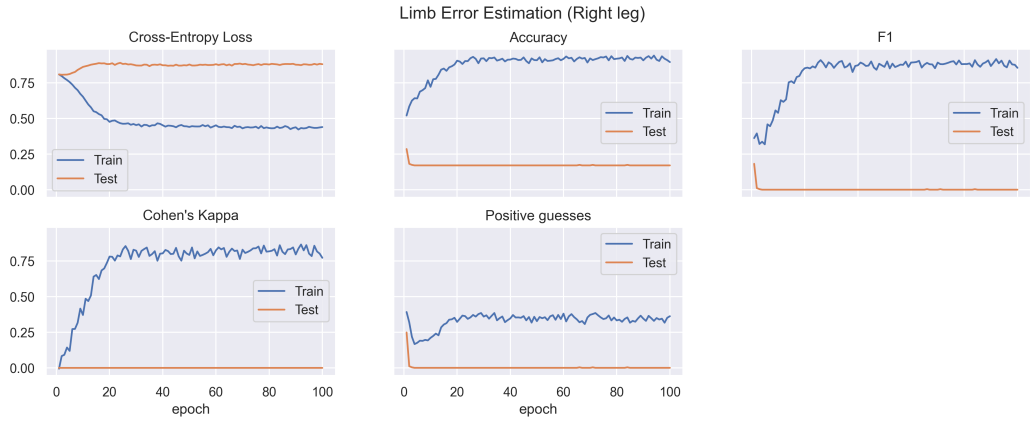
(c) Left Arm Error Estimation



(a) Right Arm Error Estimation

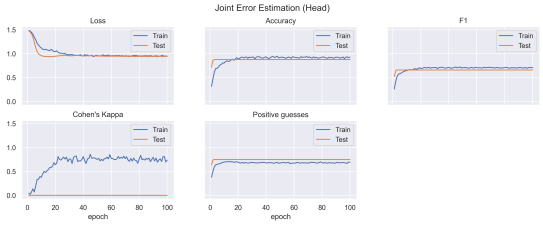


(b) Left leg Error Estimation

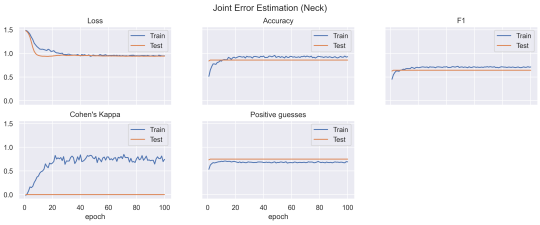


(c) Right leg Error Estimation

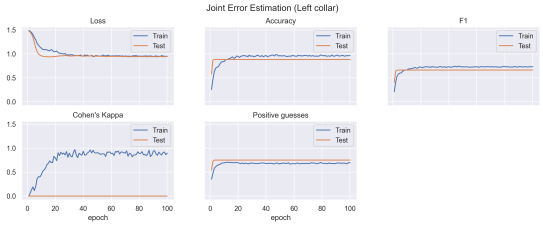
Figure B-4: The training results of the body part error estimation model.



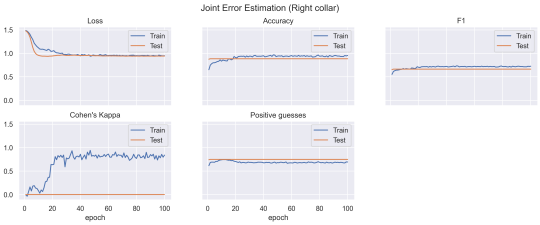
(a) Head Error Estimation



(b) Neck Error Estimation

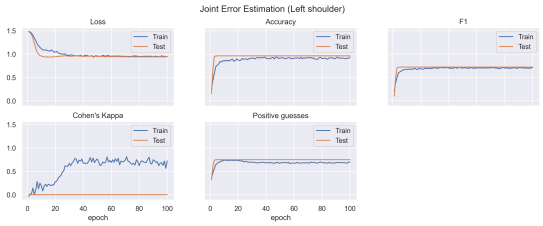


(c) Left Collar Error Estimation

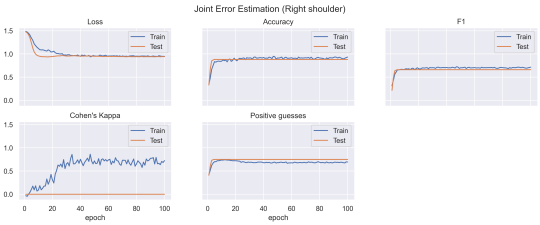


(d) Right Collar Error Estimation

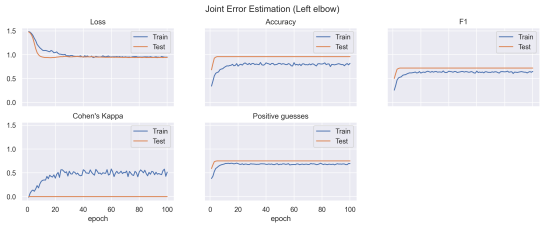
Figure B-5: The training results of the Joint error estimation model.



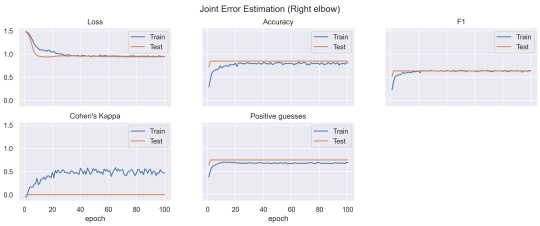
(a) Left Shoulder Error Estimation



(b) Right Shoulder Error Estimation

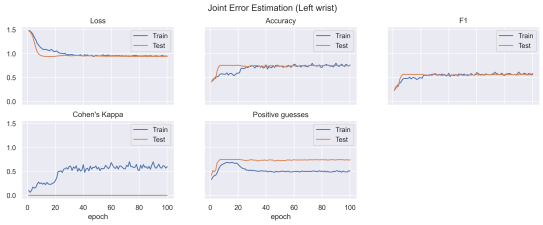


(c) Left Elbow Error Estimation

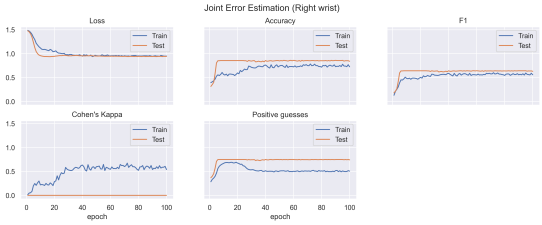


(d) Right Elbow Error Estimation

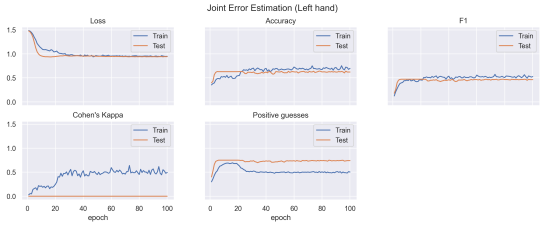
Figure B-6: The training results of the Joint error estimation model. (cont. 1)



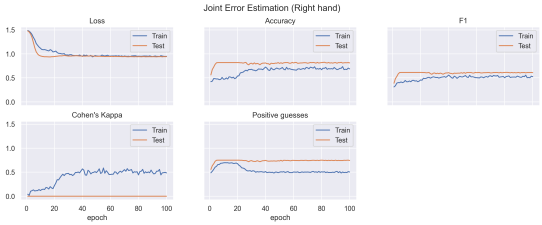
(a) Left Wrist Error Estimation



(b) Right Wrist Error Estimation

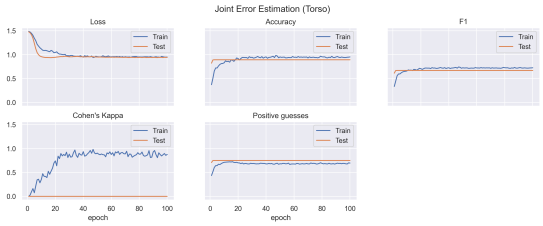


(c) Left Hand Error Estimation

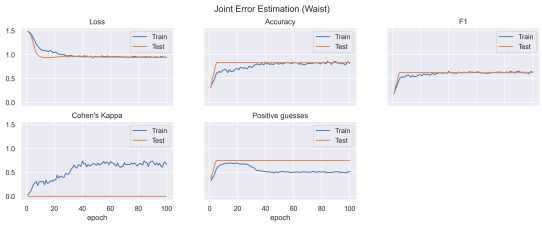


(d) Left Arm Error Estimation

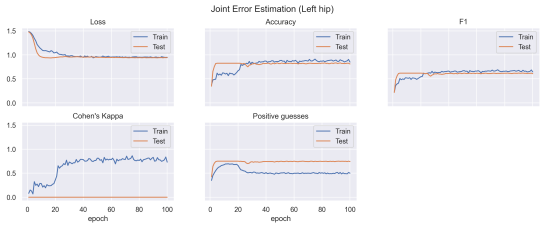
Figure B-7: The training results of the Joint error estimation model. (cont. 2)



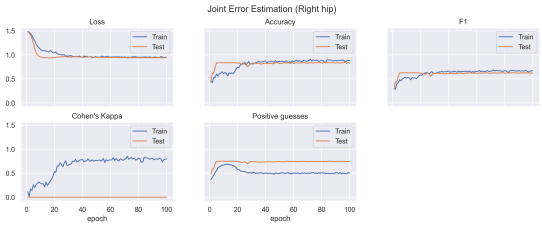
(a) Torso Error Estimation



(b) Waist Error Estimation

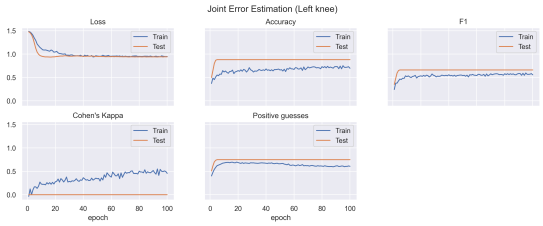


(c) Left Hip Error Estimation

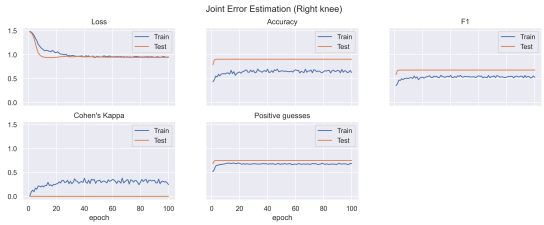


(d) Right hip Error Estimation

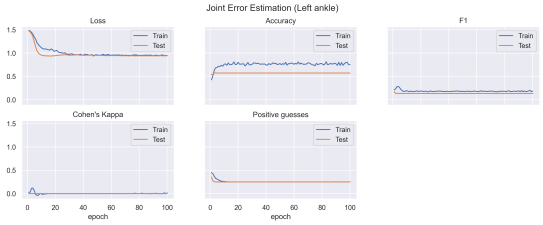
Figure B-8: The training results of the Joint error estimation model. (cont. 3)



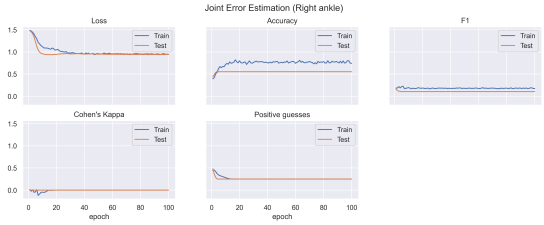
(a) left Knee Error Estimation



(b) Right Knee Error Estimation



(c) Left Ankle Error Estimation



(d) Right Ankle Error Estimation

Figure B-9: The training results of the Joint error estimation model. (cont. 4)