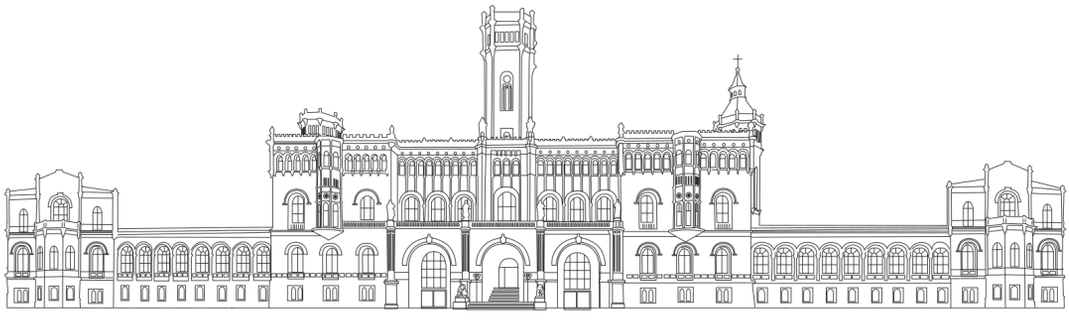
Recognition of Interpretable Gestures

Institut für Verteilte Systeme – Fachgebiet Visual Analytics
Leibniz Universität Hannover

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science (B. Sc.)

vorgelegt von
Chi Thanh Pham
Matrikel-Nr.: 3068560



Erstgutachter: Prof. Ralph Ewerth
Erstgutachter: Prof. Sören Auer
Betreut von: Kader Pustu-Iren

17. Dezember 2020

Signed declaration

"All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author and work. Any illustrations which are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole."

Name: Chi Thanh Pham

Signature:

Date: 17th of December 2020

Abstract

Gestures are the most intuitive form of non-verbal communication that is a vital component to the way humans express their thoughts. Within the past two decades, the growing interest in the study of gesture recognition has greatly improved the interaction between a human and computer. Wearable devices have been a widely used mechanism in the earlier years of gesture recognition however interest has shifted focus towards using methods that provide more ease of use hence aid more natural movement such as camera-based systems. Previous research of gesture recognition has largely been centered on the hands due to its applications in Sign Language Recognition and virtual control. However, gestures usually involve the whole body - e.g. bowing. This paper seeks to propose a method of recognising interpretable gestures from a camera-based system without any special devices to make it practical in real-world applications. Concepts from still-image based recognition, skeletal-based recognition, machine learning and deep learning are explored in this paper which provide the foundations of this project's methodology. Two datasets of 11615 images in total of 49 gestures used in this paper have been acquired from web scraping and manual photography. Experiments on the datasets determine a combination of skeletal data and random forest classification as the most accurate method with an accuracy of 58 %.

Contents

1. Introduction

1.1. Background

Throughout history, gestures have played a key role in the way humans communicate language and thoughts. A gesture can be defined as a physical movement of the hand, arm, body, head, or face that is expressive of an idea, opinion or emotion. Their existence enhances the speaker's point and at times, the point can be already interpreted by the gesture alone without the need for speech. According to Mehrabian [1], both signals and gestures can influence up to 55% of the message. In the situation of one speaking on the telephone, one can even find themselves continuing to perform gestures - this only highlights the extent of how innate gestures are within communication. Gestures have evolved in such a way that more recent languages have been created from gestures alone such as, sign language, aviation, diving and first aid [2].

Gesture recognition is an active and recent field within computer science with the ultimate aim of accurately recognising and interpreting gestures using computer vision, machine learning and mathematical algorithms [3]. Computer vision aims to aid computers to be able to "look" and accurately interpret what digital images within photographs and videos are portraying. Despite the fact that identifying the content of a photograph and/or video may be extremely trivial to the everyday human, it is an extremely difficult problem to teach machines the same problem due to the sheer complexity of human vision and perception. In order to find a solution, machine learning and its learning algorithms provide the mathematical foundations for the computer to train its models in order to be able to deduce its input image correctly and output an accurate inference.

In today's society, gesture recognition is a fast evolving area of research in computer vision and machine learning incorporating the concepts in human-computer interaction (HCI). This discipline provides a more powerful gateway between computers and machines and their understanding of human communication, specifically body language - more powerful than the text or graphical user interfaces that require inputs from the keyboard and mouse showing the evolution of HCI. As a result of gestures being second nature to humans, the process of human-computer interaction is streamlined and made easier. In terms of accessibility, gesture recognition provides advantageous assistance to a larger number of people including children and those that are impaired with a disability such as limited wrist and finger movement. With these additional benefits, its most well known applications today include sign language recognition and translation and virtual environment control. Furthermore gesture recognition can be used to analyse patients in healthcare by monitoring their behaviour [4].

1.2. Motivation

With the rising reliance on technology and demand for increased efficiency, gesture recognition is one of the many fields that are crucial to the growth of HCI. Previous works have mainly focused on specifically hand gestures recognition and emotional facial recognition. Consequently, most of these implementations of gesture recognition only perform an analysis of the hand or face. However, other entities such as the upper body and arms are also heavily involved in the everyday gesture such as waving and bowing. In order to be able to bridge the gap between human behaviour and computer interaction, more is needed to be done to understand the full breadth and depth of innate human movement across the whole body.

1.3. Aims and Objectives

This paper's ultimate aim is to propose a method that is able to identify gestures through image-classification and pose estimation. A key challenge of this project will be acquiring the appropriate number of images of different gestures to be used to train the models that will eventually be tested if they can recognise the content of the image that is being presented to them. Within these images also, the occurrence of image noise and extracting the important information from the image is a prominent issue that the field of computer vision continuously faces and will be faced in this paper. This method will be built based on an exploration and review of different machine learning algorithms and pose estimation libraries which will then be tested with a range of different gestures in order to analyse the accuracy of the proposed approach. From this paper, the findings can be used as an aid for future proposals and research to solve the complex problem of gesture recognition.

1.4. Challenges

Throughout the project, there have been several challenges and obstacles. Firstly, identifying the research papers relevant to the project from a wide scope of literature of literature on the topic. The papers selected for review were of high level of specificity and required considerable time to be processed. Concepts such as Computer Vision and Neural Networks were unlike anything else done before at University, which made reasoning about the content of the research a major challenge. The ideas and themes presented within this project are abstract in nature. The reader has to understand the theoretical concepts in order to be able to follow the practical part of the report. Thus, to enable a high level of comprehension, the ideas and their respective links had to be explained thoroughly. Given the scope of the project and the difficulty of each component, it was a significant challenge to plan in advance so that everything can be finished in time. Since for certain tasks it was difficult to estimate how laborious they were going to be, it was hard to predict positively how much of the project can be finished.

1.5. Thesis structure

This project contains 7 chapters, which progressively show the investigation of related literature, the available resources and techniques, the experimental outcomes and an emphasis on works that could be conducted further.

- **Chapter 2** contains the literature review covering the topic of human action and gesture recognition. It starts with an introduction different still image approaches and highlights directly on the features this project is mainly focusing on.
- **Chapter 3** illustrates the scientific fundamentals necessary for this work, as for computer vision and machine learning. Then it moves on to more technical topics such as the libraries, pre-processing and classification techniques.
- **Chapter 4** defines the requirements for the network, in order to illustrate the attempting goals, but also the boundaries of the work.
- **Chapter 5** discusses about the data regarding the number of training data for each gesture class and the features that are of interest and outlines the two distinct designs implemented in the study which describes the architecture of planned networked used in order to meet the project's aim. It also provides the experimentation plan and a description of the data split for the experiments conducted.
- **Chapter 6** starts with the initial setup of the experiment parameters and the feature set. This chapter is a progression of each experiment conducted.
- **Chapter 7** concludes the findings of the overall experimentation, whether the project's aim was met and a mention of future works of this vast field for further exploration.
- **Chapter 8** summarises the study.

2. Related Works

In this chapter, a literature survey on previous related works was conducted on the relevant surrounding fields of this project. The history of the study of gesture recognition and its methods have evolved greatly which this chapter will first overview. Nevertheless, this chapter will primarily focus on only the state of the art that is relevant to this project - approaches that analyse the body movements and poses through images. Furthermore, whole body gesture recognition research has been extremely limited whilst most gesture recognition research has been entirely focused on specifically sign language and hand gesture recognition which is what most of this literature review will explore. In this section, it first outlines the background of gestures followed by the history of gesture recognition research and then presents the various notable applications, designs and methods explored in order to tackle the problem of image recognition using still-image appearance-based models versus skeletal recognition analysis within the following processes: image processing, feature extraction and classification.

2.1. Gestures

The term "gesture" originates from the Medieval Latin "gestūra" which describes the orientation of the human body to communicate non-verbal information, such as an emotion or a command [5]. Despite the fact that gestures and postures are arguably interchangeable, gestures can also include body language and facial expression hence gestures provide more information compared to postures. Each gesture and their meaning can be influenced by geographical location, culture and/or society - there can be different meanings for the same gestures as well as there being different gestures that convey the same message. For example, bowing is commonly seen as an implication of a manner of respect world-wide, but in East-Asia it is a common way to greet, thank or apologise in a social setting [3].

Depending on the intended individual that the gesture is directed to, gestures can be categorised into two distinct groups: intristic gestures and extrinsic gestures. Intrinsic gestures consist of gestures that are produced without the intention of delivering information to others such as biting the nails and tapping the foot when one is nervous. Although intrinsic gestures are intended for oneself, you can still interpret them as an outsider such as when one is crossing their arms when being defensive. On the other hand, extrinsic gestures are meant to convey information with others. Extrinsic gestures can be divided into two subclasses: emotional gestures and non-emotional gestures. Emotional gestures could be inferred as positive or negative emotions. Furthermore, non-emotional gestures can be split into several sub-categories for example instructing, signalling and responsive gestures [3].

Intrisic and extrinsic gestures can be represented as either a static or a dynamic illustration of the human body [6]. A static gesture can be explained as a stagnant pose of a combination of at least one element of the outer human body as a means of communicating the non-verbal information to others. On the other hand, motion of certain parts of the human body such as waving to greet someone are such examples of dynamic gestures. For

a number of cases such as sign language, they can encompass static and dynamic elements. The meaning of a gesture can also be altered depending on the facial gesture - for example, a fist pump with a happy facial gesture implies victory whereas with an angry facial gesture, aggression would be implied. As the title of this work states, this research covers interpretable gestures, which would include intrinsic as well as extrinsic gestures, since both can be interpreted without the need of verbal communication, but emotional gestures will be excluded to focus on the body language. Below is a gesture classification tree inspired by Konar and Saha [3], which gives a few examples to have an overview about this research's gestures. However, please note that this is illustrative and doesn't cover the whole breadth and depth of gestures.

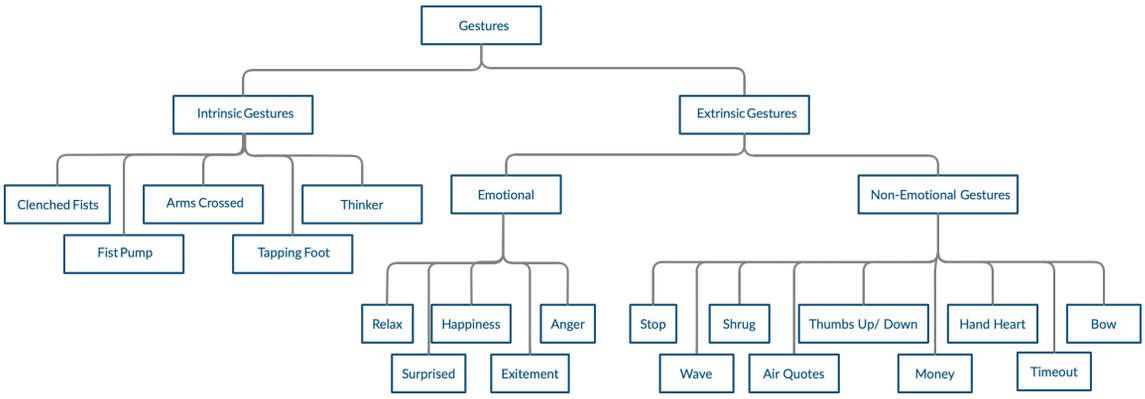


Figure 2.1.: Gesture Classification Tree.

2.2. Gesture Recognition

Gesture recognition can be explained as the procedure where the gestures made by the user are detected and can be inferred accurately by the receiver. It is a multidisciplinary research, encompassing aspects of other fields such as computer vision, image processing, supervised learning, statistical modeling and pattern recognition [6]. Over the course of gesture recognition research, a range of approaches have been used to tackle the complexity of gesture recognition. Traditional approaches have used methods such as the Hidden Markov model (HMM) and the Particle Filter. Additionally, these early approaches needed special equipment and devices, such as gloves and body suits with markers, to track and digitalise information in terms of position, orientation and movement of the body [6]. Even though small and/or fast movements can be detected with these devices, they're often inconvenient for the user as they would have to wear the device for every use and the device is usually connected to other machinery which prevents the user from moving as naturally.

The more recent models incorporate concepts from Computer Vision and can be categorised into two groups: appearance based models and 3D based models [7]. Within 3D based models, skeletal-based algorithms are a simplier and less computationally intensive method than 3D mesh models as it requires less parameters with only requiring joint angles along with segment lengths to represent the body.

Recently, Machine Learning and Deep Learning based approaches have set higher performance results in classification [8]. To achieve satisfactory results, these approaches require a large amount of labelled data.

2.3. Still-image Gesture Recognition

Still-image recognition involves using the image itself and applying image processing, feature extraction and classification onto it in order to detect the contents of the image. The aim is to simplify the image by extracting the important information through tools such as edge detection and object contouring - the outcome of these processes depend on the quality of the image and how easy it is to identify the edges which is made possible through image processing. By extracting the object from the image and removing unneccessary noise, a machine learning algorithm can be applied in order to train the model to detect what the object in the image is. This method has been a popular method amongst scientists for interpreting images due to its simplicity in implementation and fast processing speeds. The approach has been applied to many fields including detecting letters in a word, object detection and medical imaging analysis.

2.3.1. Colour Based Recognition using Skin Colour

In order for the segmentation of the image to be applied successfully and extract the necessary information from the image, the image processing stage is crucial. One of the most popular methods for segmentation is skin colour detection which is achieved with one of either of the following two methods. The first method consists of each pixel being classified into whether it is skin or not (known as pixel based skin detection) whilst for region skin detection, the skin pixels are processed based on intensity and texture [9]. To represent the colour information in an image, mathematical models called colour space are used and are obtained as several formats: red, green, blue (RGB), hue and saturation (HSV, HSI and HSL) and luminance (YIQ, YCbCr and YUV) [9].

In the study of hand gesture recognition for American Sign Language (ASL), [10] converts captured images from RGB into a binary image by using the gray threshold method to pre-process the image before segementation. This in turn produced a 90.19% recognition rate. Perimal et al. [11] includes another process between the conversion between RBG and binary image so the segmentation is more accurate. In order to recognise 14 hand gestures based on finger counting, Perimal et al. [11] first converts the acquired image from RGB to YCbCr. The image is then converted into a binary image using Otsu's method to perform segmentation more easily. External objects are removed by eliminating objects which are less than a certain pixel until the number of objects in the image is 1 (making sure the hand remains in the image) to which finger detection is applied. The recognition accuracy was 100%. This proposal experimented and highlighted the limitations from the effects of noise, light intensity and hand size. Their results showed that the presence of noise at 0.4 noise density and a light intensity of the image below 0.1% each decreased the accuracy of the gesture recognition algorithm by 50%. Another method on hand finger gesture gesture recognition was conducted by [12] where the images were also converted from RGB to YCbCr and then to greyscale. However, this method extracts the hand by searching for the largest object in the image which is assumed to be the hand which then removes all the smaller, external objects. This produced a recognition accuracy of 98%. Their findings also concluded using YCbCr is advantageous with eliminating the effects of illumination however, bright light during the capture of the image reduces the accuracy. This therefore emphasises the importance of acquiring and processing images to have minimal noise and appropriate brightness.

A limitation to skin colour detection is highlighted in the study by [13] where objects in

the background were of similar colours to the skin such as shadows and wood. To mitigate this problem and to extract the hand, the authors took the largest binary linked object (BLOB) so only the region with the biggest linked skin-colour pixels were considered.

Overall, even though images typically come in RGB formats, this format is not optimal to produce accurate segmentation as the format provides the colour and luminance information from the image, mixed together [14]. Therefore, previous literatures have typically converted their images into another colour space before processing the image. Hue and saturation formats and their characteristics perform well with variations in lighting. However, to convert RGB to HSV or HSI is time consuming due to the transformation from Cartesian to polar coordinates hence this colour space is only suitable for colour detection in simple images. Compated to hue and saturation formats, converting RGB to YCbCr is simplier and more appropriate for skin colour detection and therefore better for segmentation [9].

2.3.2. Deep Learning Based Recocognition

Deep learning techniques such as Convoluntional Neural Networks have proven their accurate performance in a range of classification and recognition tasks. For example, Lin et al. [15] uses the concept of skin colour based recognition by extracting the hand object in the image by segmentation by converting their RGB image to YCbCr. The skin model is then trained to classify skin colour and non-skin colour by a Gaussian Mixture model. From this, the image of the hand is calibrated and fed into the CNN - where there are 8 layers - to train and test the network. Their proposal to recognise 7 gestures produced accuracy results of 95.96%. For the future, the study aims to apply a high-level semantic analysis to improve the recognition for more complex gestures and tasks. In order for this to happen, more data is needed. Islam et al. [16] tackles the problem of obtaining an appropriate amount of data for training. To recognise static hand gestures, Islam et al. [16] uses Convoluntional Neural Network (CNN) with data augmentation. In this approach, the background of the images were removed using K-gaussian distribution and then converted to grayscale. A morphological erosion and median filter was also applied in order to reduce the noise in the image. The image is then passed to the CNN for training. The training process composed of two phases: training with the base dataset where the CNN was applied on the images with pre-processing only and training with dataset that was augmented; done in order to increase the number and variation of data by applying changes such as zoom, rotation and flip. In conclusion, the model without augmented data achieved 92.87% whilst the model with augmented data achieved 97.12%. Convolutional Neural Networks and augmentation are also used in the methodology by Tao et al. [17] as a method of American Sign Language recognition. By using the 3D information from depth images, multiview augmentation from virtual cameras was applied in order to generate more images from different perspectives. Their CNN model is composed of a layered feature extraction module and a classification module. For feature extraction, the images were normalised and fed into three 5x5 convolutional layers with each layer followed by a 2x2 max pooling layer. Overall, this method achieved an accuracy between 93 - 100% with public datasets. In the study by Bao et al. [18], the authors omit a segmentation and detection stage - to remove the unnecessary objects from the image - and apply a convolutional network to directly classify the gestures in the images. Similarly to Islam et al. [16] and Tao et al. [17], Bao et al. [18] also incorporates augmentation into their methodology which provided the authors a total of 500,000 data samples for training. Their CNN is composed of 9 convolutional layers, 4 pooling layers and 3 full connected layers. Their design to classify seven types of hand gestures produced

an accuracy of 97.1% with datasets with simple backgrounds and 85.3% with datasets with complex backgrounds.

2.4. Skeleton-based Human Action Recognition

Within the past few years, the study of skeleton-based recognition has significantly improved the advancement of human action recognition allowing the complexities of recognition in videos to be tackled. A typical method to recognise human actions is by analysing body poses and movements with skeleton-based algorithms which extract information about the subject. The most widely used feature that is used to represent the geometric attributes of the object is the joint orientation. This consists of the joint location, the length of space between the joints and the degree of angle between the joints [9]. Skeleton-based recognition proves beneficial in relation to the background and lighting of an image. Regardless of variation in background and lighting, the skeleton and joints are accurately recorded with pose estimation algorithms. As a result, pre-processing the image in ways such as conversion from RGB to a binary image is not necessary which streamlines the process and therefore is less time consuming.

For the recognition of human actions, Vijaya Prasad et al. [19] acquired the raw 3D skeletal keypoints of the joints from the Microsoft Kinect sensor which is used to eliminate the problem of the variation of view on spatio-temporal data. From this, the locations of the joints are converted into a RGB image which is passed through a CNN inspired by VGG net but with 8 layers. Consequently, the results averaged an accuracy of 84.37%. A limitation to this approach however, is that it requires the use of a sensor which is costly and decreases the ease of use for the user. Konstantinidis et al. [20]'s proposal for sign language recognition based on skeletal data does not require any sensor devices and relies purely on the RGB features of the images so pre-processing the image is not needed. Not only does the approach incorporate data from the hands but also from the body too. A pre-trained ImageNet VGG-19 network is used to separately extract the hand and body skeleton detection using a different number of layers for each extraction. To combine these extractions together, the authors remove some leg joints as it is not relevant to sign language. From this, a skeleton classification network is used which is built on the basis of the absolute coordinates of the body and hand joint coordinates and the length of the lines in between each joint. As a result, the network had an accuracy of 98%.

Specialised sensor data recorded by devices presents various obstacles as it is expensive, strenuous and furthermore requires custom software which means the environment to use this is limited. The current state of the art for camera based skeleton-based recognition using images have the the advantage that they are quite fast and reliable and have no requirement for a sensor device like the Microsoft Kinect. Despite this, they lack the ability to properly analyse whole body gestures since they don't extract features that include hands, body and faces all together. One network that is able to do that is OpenPose.

2.4.1. OpenPose

Cao et al. [21]'s system uses a RGB image to generate a set of whole-body human key points for each person detected. In comparison to previous mentioned works the extracted keypoints contain not only information from the head, torso and arms, but also facial features, hands, legs and feet.

Previous methods of keypoint detection only consisted of either a detection system for facial keypoints or body keypoints but not combined. OpenPose is the first real-time multi-person system to be able to jointly detect human body, foot, hand, and facial keypoints (in total 135 keypoints) on single images or video frames.

The user can select an input between images, video, webcam, and IP camera streaming. This implementation is very helpful, because we want to focus on images first to be able to translate those features into different uses like videos. OpenPose consists of three different blocks: body-foot detection, hand detectio, and face detection. The following presents several papers that cover different approaches to classify certain actions, gestures and poses with OpenPose.

Image classification algorithms within human activity recognition generally have difficulty in classifying a set of multiple images from a video into an activity class, because activities usually are performed over a short period of time meaning if a single frame is captured, it is more challenging to identify the activity. Sawant [22] proposed a method to recognise human activities in real time by using Openpose and long short-term memory networks. This system detects activities such as boxing, waving, clapping and jumping (jacks). The keypoints of the skeleton of the body are extracted by OpenPose which takes into account background noise and low illumination. The keypoints are then passed through an Long Short Term Memory (LTSM) Network which is a type an Recurrent Neural Network (RNN) architecture different from standard feedforward Neural Networks as LTSM has feedback connections. Results obtained show an accuracy of 87.12%. Another human activity recognition explored is by Yadav et al. [23] who presents an approach to accurately recognise a number of yoga asanas during real-time videos. A convolutional neural network (CNN) is used to extract the keypoints of the skeletal body of the subject for every frame using OpenPose. Like Sawant [22], long short-term memory is used (LSTM) to provide temporal predictions of the pose. The system was tested with 12 subjects and achieved an accuracy of 98.92%. Both papers use recurrent neural networks (RNNs) which have been proven to be advantageous for sequential information such as speech recognition. Since an activity can be considered as a sequence of actions, RNNs are suitable to process this sequential data. Considering the various RNN architectures, long short-term memory networks (LSTMs) is able to store information for an extended period of time and therefore is widely used. Both approaches eradicates the reliance on sensor devices making it user friendly and hence shows only images or videos as a sequence of images are needed to classify activities.

Despite the good performances of all those works, there is no work that covers a big variety of interpretable gestures. Using the findings from the analysis of the previous literatures that have already approached this problem, how this project is going to approach this problem, will be illustrated in Chapter 5: Design.

3. Fundamentals

The following chapter explores a study of the background research, providing an overview of the fundamental concepts and theoretical knowledge used during the implementation stages of the project. It begins with the definition of gestures and computer vision followed by an introduction to concept of gesture recognition and an overview of machine and deep learning. A background review and investigation of certain classification methods is then covered and compared, as this is the main area of research in this study to achieve the goal of gesture recognition. After this, the various libraries and tools that will be useful to this project will be explored and overviewed.

3.1. Computer Vision

Computer vision is a subfield of research within artificial intelligence that aims to explore and develop methods to help computers in gaining understanding of the content from digital images or videos [24]. A trivial task for humans, recognising and intepreting by vision remains a challenging task for machines due to how complex vision perception is within a continuously changing environment and little understanding of vision in a biological sense [25].

The tasks that are involved in this discipline include acquiring, processing, analysing and understanding of useful information from digital images [26]. The "understanding" aforementioned in this context refers to the process of having an input visual images which then are related to representations of the physical world that are plausible to thought processes and can generate appropriate action [27]. Many applications of computer vision incorporate various recognition methods including:

- Image acquisition: Digital images are obtained by at least one image sensor resulting in image data that is either an 2D or 3D image or sequence of images. Each pixel within an image corresponds to light intensity in a spectral band within grey or colour images, or other physical measures, such the reflectance of electromagnetic waves [28].

- Pre-processing: To satisfy the parameters of the computer vision methods, it is necessary to process the data to make it compatible with the method that will be applied on the data. For example, noise needs to be removed from the image so it doesn't interfere with the inference of the image and constrast and brightness needs to be refined so that it is easier for the system to detect the important information [25].

- Feature extraction: Relevant features within an image with are extracted from the image data. Typical examples of such features involved in gesture recognition include edges, blobs and colour [28].

- Segmentation: A decision is determined about which image points or regions of the image are relevant for further processing. This can be done by extracting the region of interest or removing the irrelevant objects from the image [28].

- High Level Processing and Decision Making: This stage involves a set of data that in inputted in order for the system to make the final decision such as match/no match in recognition [28].
- Pose estimation: The position and/or orientation of a specific object in an image is determined. An example application for this would be tracking the pose and motion of humans for gaming consoles.

3.2. Machine Learning

In 1959, Samuel [29] defined the term "machine learning" as a "field of computer science that gives computers the ability to learn without being explicitly programmed". Deemed as a pioneer in machine learning, he created a machine that was able to beat its opponent in a game of drafts by teaching itself. More recently, [30] presented a definition widely used today: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E." The foundation of this branch of computer science is to provide a machine the ability to learn a solution to a task without having to program it explicitly. The abundance of data now available today provides one of the main mechanisms to appropriately train these machine learning algorithms. Machine Learning can be categorised into four groups: supervised, unsupervised, semi-supervised and reinforcement learning. Supervised will be explored in this chapter as this is the machine learning this project primarily focuses on.

3.2.1. Supervised Machine Learning

The main techniques used in this project revolve around a subcategory of machine learning classification algorithms - supervised machine learning algorithms. Given a dataset with already labelled values, supervised machine learning aims to create a classifier which can approximate a mapping function Y = f(x). The ultimate goal is to continuously refine the mapping function to the point that when you input new data, you can precisely predict the output variable Y. Some of the most popular supervised machine learning algorithms include:

- K-nearest Neighbours
- Decision Trees and Random Forests
- Support Vector Machines
- Naive Bayes
- Neural Networks

To evaluate a model, the simplest method to use is the holdout method. Within this method, the dataset is divided into the training and testing dataset to which the learning algorithm trains the model by using the training data which will produce the target values Y. The performance of a model is measured by how well it predicts the values of Y with unseen data from the test data. The more accurate a model is on unseen data, the higher

generalisation ability of the model for the classification task hence the ratio of correct values from all predictions - including from the test data - is higher.

Within supervised learning, there are a few limitations that can occur. The first issue is the inevitable compromise between bias and variance. The bias is the error from incorrect parametric assumptions made by the learning algorithm whilst the variance is the result of inconsistencies within the training data. The bias is reduced if the model has higher variance however to reduce the variance, the bias has to be increased. It is assumed that each data in the training dataset are statistically independent from each other and the testing data is required to be distinct from the training data to prevent resubstitution validation to minimise optimistic bias on the account of overfitting. Moreover, supervised machine learning requires a substantial amount of training data to be satisfactory and effective and accurately approximate the classifier mapping function [31]. To determine the most suitable algorithm for our specific problem, it is necessary to compare different algorithms and optimisation methods.

Support Vector Machine

An SVM model is a type of supervised learning classifier which performs classification by representing training samples as points in a finite dimensional vector space, where each dimension represents a specific feature of the data [32]. Support vectors are the vectors that define the hyperplane. Each feature will be arranged in the feature space according to their values and then separated by a hyperplane and then the new samples are classified depending on their relative position -which side of the hyperplane they fall on. However, if there are a lot of data points near to the hyperplane, there is a possibility that it could be misclassified as it shares similarities in terms of features. As a result, it is difficult to distinguish between the two features and make a more exact classification. Hence, datasets that contain data with corruption and noise with overlapping classes are not suitable in SVMs.

K-Nearest Neighbour

K-Nearest Neighbour (KNN) is a classification approach that assumes that similar things group together [33]. The method uses measures similarity by the distance between points on a graph - usually by the Euclidean distance. Data is classified by observing the class of the neighbours that occur the most within the region around data point out of all the K neighours around it. As a result, KNN is a non-parametric approach since it makes no assumptions about the distribution shape of the data. The limitations to this method is that KNN is unable to differntiate between relevant and non-relevant observations [34].

Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is an algorithm that is the simplified version of Gradient Descent. SGD is a optimisation method that estimates the gradient iteratively and calculates the loss based on a random value and uses the loss to update the gradient function, instead of computing the gradient exactly [35]. The model is updated during those iterations and therefore doesn't need to memorise past calculations. The adjustments depend on a given learning rate and affects the convergence of the algorithm. If the learning rate is too large, the algorithm takes steps that are too big down the slope and may jump pass the minimum point and therefore missing it [36]. SGD is proposed to be used when

there is a large amount of data, because it is a time-efficient method. For SGD to work effectively, it is suggested to randomly shuffle the data in case it is sorted in classes or any other way.

Naive Bayes

The Naive Bayes method is a set of supervised learning, probabilistic machine learning model that is based on the Bayes' theorem principles:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

It is assumed that the labels of the predicting class are independent from one another even though this is not always the case. In the above equation, the numerators are both normally distributed and divided by a constant probability [37]. These, as a result, create the posterior (also normally distributed) which fits the probability of a new label to be corrected. This probability is then calculated for each correction label and the label with the highest probability is then assigned as the error correction [32]. The method is beneficial in terms of speed in comparison to other methods and also mitigates problems occurring from the curse of dimensionality however it has limitations in terms of estimations. [32]

Decision Tree

A decision tree is a type of supervised learning algorithm based on decision trees which divides the large dataset into smaller subsets. Data points are split into different branches based on the feature class variables with the root being at the top. The best conditional variable values and their order are arranged in such a way that the most similar datasets are formed at the leaves to categorise the data [38]. The data is categorised in separate alike sets based on the decisions from the tree's branches which is then categorised into the item's target value at the tree's leaves. This method has been widely used due to its simplicity in understanding, ease of implementation and its ability to handle both numerical and categorical data. However, decision trees can create trees that are over complex and too specific so the data is not generalised - known as overfitting. Different trees can be created due to small variations in the data and biased trees can be made if certain classes dominate - known as the concepts of variance and bias [38].

Random Forest

Random forest is an ensemble of decision tree, supervised learning algorithms used for classification and regression problems. It is usually trained with the "bagging" method where a combination of learning models increases the overall result [39]. The method builds multiple decision trees and combines them together to produce a more accurate prediction. While the trees are being built, this method adds randomness; instead of finding the most important feature, it searches for the best feature from a random subset of features when splitting a node. As a result, this produces a diverse tree that is able to accurate categorise the data and prevents the issue of overfitting seen in decision trees. A limitation of the random forest is that due to the large number of trees being built, the process may be too time consuming and slow for real time applications [39].

Principle Component Analysis

Principal component analysis (PCA) is a widely used technique in reducing dimensionality of large datasets by processing a larger set of variables into a smaller one that still contains the majority of the information from the previous larger set [40]. This method is used as a small compromise of accuracy for simlicity which makes the algorithm faster [34]. First, a covariance matrix is created to overview how the variables are related to each other and then this matric is analysed with two distinct components: direction (eigenvectors) and magnitude (eigenvalues). The original data is transformed to align with the aforementioned components. The data is then projected into a smaller space by removing the eigenvectors that are least relevant and thus reducing the dimensionality [41]. With PCA, it reduces overfitting and improves visualisation of the data however there is a possibility of relevant information loss as the eigenvectors are removed.

3.2.2. Artifical Neural Networks

Inspired by the concepts of neuroscience and neurons in the brain, Artificial Neural Networks - also known as Neural Networks - are an implementation of machine learning. From a biological perspective, the nervous system comprises of a group of interconnected neurons that transmit signals across the whole body. Complex chemical processes with electrical impulses travel onto neighbouring neurons via synapses which make up the connections between the neurons [42].

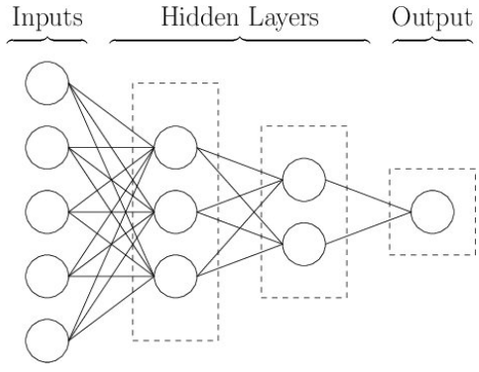


Figure 3.1.: A simple neural network [43].

Within a neural network, a number of neurons are connected together and separated into layers. The data is processed through the input layer, which then passes the data through a number of hidden layers, which then passes the data into the output layer. The network is trained iteratively from a given input to predict the classified output by refining its weights by comparing the correct output with the network output by using a function at each iteration. As this process repeats on a satisfactory number of input samples, the network begins to generalise to new data by taking in patterns in the input data and adjust the strength of each neuron.

The choice of hyperparameters (including the number of neurons in a layer, the number of layers, choice of activation functions etc.) is important as there are many possible combinations that have dramatic effects on the performance of the network. It is possible to use hyperparameter search methods to find the best possible settings, however it is typical use a standard configuration for time efficiency.

Back-propagation uses the output error to change and refine the weights of the network and hence improving the model. A perceptron, a single neuron, with multiple inputs can be described to "fire" if a certain condition is matched and else it does not "fire". However, a single layer Perceptron can only classify linearly solvable problems, with the XOR problem being an example. FeedForward Neural Network (FNN) provides a solution where perceptrons (nodes) with their own weights called hidden layers are in between the inputs and outputs. Generally the output is a vector of probabilities, where the maximum is taken for classifying problems or an average is taken for regression problems. FeedForward refers to the fact that information is only moving forward through the network [44].

3.2.3. Convolutional Neural Networks

Rina Dechter coined the term "Deep Learning" in 1986 which simply refers to a Neural Network with a large number of hidden layers [45]. A Convolutional Neural Network (CNN) is a type of Deep FeedForward Neural Network. Named from the type of hidden layers that make up this network, the hidden layers of a CNN generally is made of convolutional layers, pooling layers, fully connected layers, and normalisation layers [44]. CNN have been an extremely popular method especially in the field of image recognition due to its high accuracy however this method requires a large amount of training data and is computationally expensive. Instead of using the normal activation functions previously mentioned in Neural Networks, CNN uses convolution and pooling functions instead. The architecture of CNN ensures they are relatively spatially invariant, which means that in terms of classifying images for example, transformations of images have a low effect on classifying. This is achievable in CNN as the network tries to avoid connecting each neuron in one layer to every neuron in the next layer - also know as fully-connectedness which makes the data prone to overfitting. The CNN input is arranged into a gridlike data structure which is propagated through each layer of the network while preserving spatial relationships [46].

At each convolutional layer, the input matrix is processed with a pre-specified kernel

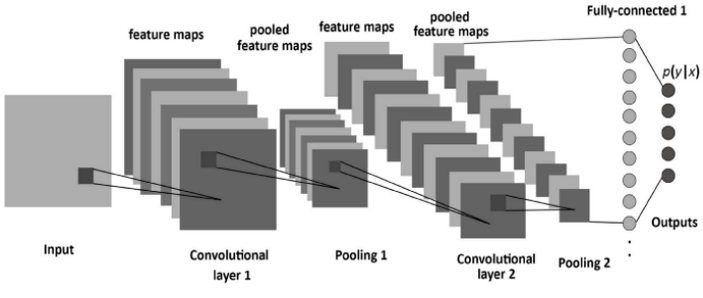


Figure 3.2.: A typical structure of a convolutional neural network [44].

operation and enters into the feature output layer where the standard activation function is applied. The idea is to use layers of kernels (filters) to scan over an input, learning important feature maps for the input, which are then used to generate smaller inputs for another layer of kernels. A kernel is a n-dimension array made up of weights, that are tuned iteratively by the network. For example, a kernel on the first layer of the network may have its weights refined to detect edges of the images, while the deeper kernels will learn other relevant features. The derivative of the error is back-propagated through the network with each pass, tuning these filters, until finally the network outputs a classification or regression [44].

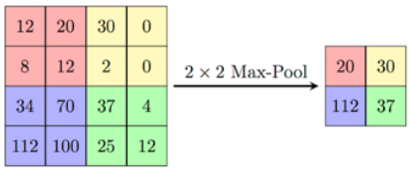


Figure 3.3.: A max pool operation applied on a 4x4 input region [44].

A pooling layer is placed in between CNN layers in order to continuously reduce the dimensionality. As a result, each layer's purpose is to shorten training time and prevent overfitting. Max pooling is the most widely used, which extracts the most relevant value in each region. The aim of this is to prevent over parameterisation while retaining the most valuable information. Alternatively, average pooling calculates the average value for each region [44].

3.3. Useful Libraries

Python has an extensive amount of libraries and tools that provide aid in relation to machine learning for this particular task. The following libraries were used extensively throughout the project.

scikit-learn

Scikit-learn is a Python library used for machine learning purposes. It contains numerous clustering, classification and regression algorithms, but also provides a simple implementation for evaluation. The classification algorithms Scikit-learn features include support vector machines and random forest and also operates well with other libraries such as NumPy and SciPy. The API is uncomplicated to use, fast and robust making the library a widely used one. The library gives thorough documentation in regards to the numerous classification techniques and the set of hyperparameters available. Scikit-learn does not support GPU use, therefore it is doesn't deal with platform specific issues and can be used on many platforms. [32]

Tensorflow/Keras

Like Scikit-learn, Tensorflow is also a library for machine learning purposes however Tensorflow proves advantageous in relation of parallelism, numerial computing and enabling optimal training time for models. Since implementing deep learning models are likely to require a substantial amount of time for training, this makes Tensorflow is a suitable tool for Neural Networks. As well as this, Tensorflow has GPU-support which makes it perform significantly faster than CPU-only systems which makes it superior for Deep Learning [47]. Keras is a high level neural network API that is based on Tensorflow. Known for its user friendliness, the library takes advantage of CPU and GPU like Tensorflow which results in increased computing performance. The principle of Keras is that it is designed to be able to support the process from conceptual ideas to a model in a short amount of time. It is a simpler library than Tensorflow which makes building Deep Learning applications more streamlined. However it is more limited with customisation in comparison to Tensorflow [47].

ResNet 50: Pre-trained CNN for Image Classification

Deep neural networks continuously set new high standards in image classification [48], meaning that there are many to choose from. Residual Networks (ResNets) are neural network implementations that have won competitions for ImageNet detection, ImageNet localisation, COCO detection and COCO segmentation. ResNet 50 (ResNet 50 will be described as ResNet in the following chapters) was the model that won the ImageNet challenge in 2015. AlexNet, the winner of ImageNet 2012 had 8 convolutional layers, the VGG network had 19 and GoogleNet had 22 layers and 152 layers were present in ResNet 152. In theory, it is known that deeper Neural networks perform better, but from a certain threshold, their performance decreases again due to the vanishing gradient problem. Despite this, ResNet overcomes this obstacle by using "Shortcut Connections" which allow them to venture into deeper neural networks [49].

OpenPose Alternatives

Most of the current state of the art refer to the original OpenPose, but with the setup being incompatiable with our current systems, it is necessary to work with alternatives. The two unofficial OpenPose networks that were compatible with our systems and easy to implement and to work with were a Tensorflow version [50] and a PyTorch version [51]. The felixchenfy/Realtime-Action-Recognition implementation [52] with Tensorflow pose-estimation [50] had a network for training classifiers for videos already. Since the chosen actions in [50] require hand-pose-estimation, which are quite important for gestures, the network structure needs to be altered and adjusted in order to meet the requirements for this project. The PyTorch OpenPose (later referred to as OpenPose) network is only made for single images and folders with images which can be additionally implemented for videos as well, but more importantly, it includes a model for body pose-estimation as well as a hand pose-estimation which is based on Simon et al. [53]. The network does not provide the coordinates of the key points thus this will have to be manually coded. Connecting the hands to the body will be a challenge, because they are estimated separately by independent models.

4. Requirements and Analysis

4.1. Aims and Objectives

The project goal is to implement a supervised machine learning classifier to recognise gestures for interpretation from an image input. In this project, we focus on images because creating a dataset based on videos is out of scope for this project, but it should be possible to classify a video based on images by breaking it down into frames and then treat it like single images. Logically this feature would not be able to consider the aspect of time for dynamic gestures, but only the estimated pose in the frame. Furthermore, the study will examine how the network accuracy varies with changing certain gestures to understand if there is a pattern for which gestures can be predicted easily and which ones not easily, as well as examining the effect technical indicators have on the network accuracy.

4.2. Testing Solutions

Testing is an imperative part of this project; any errors or mistakes with the algorithm need to be identified and corrected. If errors remain until the results stage, results will be significantly hampered, or false conclusions may be drawn. Not only should errors be evaluated throughout, but the performance and accuracy too. The recognition will not be restricted to single gestures, therefore a total of 49 gestures will be used in testing. To avoid overfitting, the ideal network will be able to generalise from all datasets with similar features. Furthermore, by choosing parameters that work well with a variety of gestures, the project becomes much more precise.

4.3. Evaluation of Solutions

Evaluation is similar to testing in the regard that it assesses the running of the project. However, evaluation will be used as an indication of the completeness of the final project. Detailed evaluation will be useful for future work that either builds on the work done here (fixing mistakes, optimising). To determine the project's success, there will be two metrics, one will focus on requirements and the other one will focus on the networks performance and recognition quality. As with all projects, the first and most simple check will be with the requirements table (Table: 4.1), such as if all of the high priority requirements have been met and an explanation of how the requirement were matched. Furthermore, and arguably more importantly, by doing this, it can be clarified why a certain requirement could not be met, what problems prevented it to be matched and how can it be avoided in the future. The second metric will be the scikit-learn's internal metrics to evaluate the performance of the network. There are many useful model measures already included within the libraries. Within the design stage, the specific metrics will be looked at in more detail. Pre-packaged metrics are a fundamental part of the scikit-learn library. Therefore, they are very easy to implement by calling simple functions and reliably measure the networks performance.

4.4. Limiting Factors

Due to the fact that a completely new dataset had to be created for numerous classes, the quantity of data might not be enough, but also the quality in terms of noise is uncertain. If there is a lot of noise in the data, it has a bigger impact on a smaller dataset, which can make the classifier inconsistent and manually cleaning the dataset is very expensive. Another limiting factor is time. There is a significant amount of theoretical further knowledge that must be acquired concerning concepts outlined in the literature review. Furthermore, due to the fact that executing jobs must be queued on the University's High performance computer (HPC) which has a maximum amount of resources, it may not be possible to conduct as many experiments as planned, in the time frame. The solution to this, is to prioritise the most necessary experiments. Moreover, the planned framework uses Python's Keras, PyTorch and scikit-learn, which was not used before. However, documentation exists for the libraries as well as a number of online tutorials, so it is a challenge that can be overcome.

Table 4.1.: Project requirements with priority ranking.

Requirements	**Description**	**Priority**
Algorithm creation	The chosen algorithm will be able to recognise interpretable gestures of people in a image.	Essential
Algorithm improvement	The chosen algorithm will be improved by becoming multi variant allowing the input of many technical indicators.	Essential
Technical indicators	Optimise the number of chosen technical indicators to increase accuracy and returns.	Essential
Divversity of gestures	Be able to recognise a variety of interpretable gestures.	Essential
Algorithm comparison	Compare various machine learning algorithms to determine the improvement actually performed.	Essential
Evaluation of the algorithm	Compute how well the algorithm has performed on a given dataset using a collection of metrics.	Essential
Executional Program	Write a program, that executes the model on given input and returns results.	Essential
Demonstrational visualisation	Create an output for the program that visually shows the estimated skeleton, but also shows recognised gesture with its interpretations.	Desirable
Input videos	Extend the functionality of the program with the feature to input videos that can be analysed and return the prediction and the video.	Desirable

5. Design

The following section provides an explanation of the datasets used and the tools required to approach the problem of recognising intepretable gestures. Two proposed methodologies will be discussed in 6 and will explain exactly how the networks were technically implemented. A properly thought out design chapter will explore all of the options available - settling on the best one. It is important to understand where weaknesses lie in the tools available and the limitations they impose. The first part examines the structure of the project and dataset available, before choosing a network and the evaluating the possible tools to build it. As a consequence of the lack of a fitting dataset, it is decided to create a dataset from scratch. Unfortunately, because of poor performance of the first dataset acquired, there was a change of plans necessary and it was decided that there will be a additional dataset to figure out what needed to be done to fix the network. Significant decisions will be made in this chapter that will direct the remainder of the project.

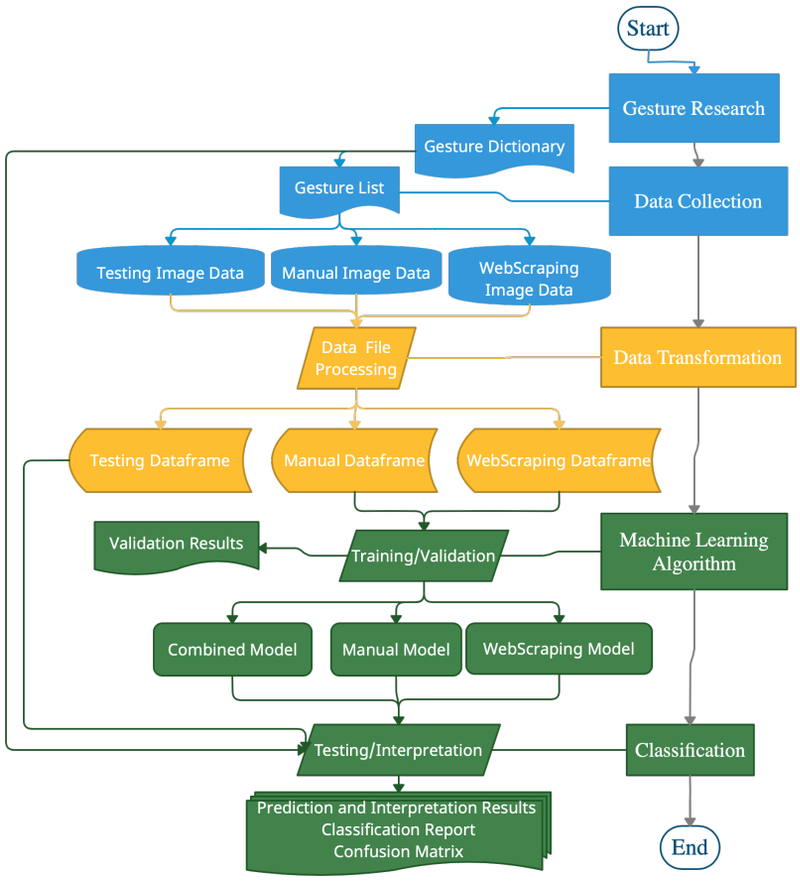


Figure 5.1.: Flow chart of the network.

First, there needs to be a foundation for this project to build on. For this, we constructed a Gesture Dictionary 5.1 for interpretable gestures. Using the information from the dictionary, the classes can be defined which are going to be the target values that are aimed to be classified. Afterwards, we can generate our two datasets: one using web scraping and one by manually collecting the image data with a camera. From this point on, the project will be split into two experiments that follow the approaches reviewed in 2: feature extraction using a pre-trained CNN for object detection and feature extraction using a pre-trained CNN for pose-estimation. Both approaches return features that need to be processed and fed into a learning algorithm that generates a model with the intention to recognise interpretable gestures from other images.

5.1. Gesture Dictionary

Due to the lack of a proper list of all interpretable gestures, it was necessary to do research on our own and create a dictionary with all the gestures and their meanings which vary in different local areas, cultures and societies. Each entry in the dictionary contains a gesture label, gesture name and gesture interpretations with descriptions, types and area-used. This dictionary will be used as the source for the classes that are planned to be classified and will be used later to return the interpretable meanings of classified gestures. For the case that there is no gesture visible, there are two additional classes: standing straight and sitting. During the training and the testing where we know that all observations belong to a class. However, this is not given in real-world situations, so there will be a threshold in the executional program, which returns "gesture not recognised" for predictions below that gesture.

Example:

```
{
  "gesture_label":"gesture_1",
  "gesture_name":"bow",
  "gesture_interpretations":
    [
      {
        "description":"respect",
        "type":"social",
        "area_used":"world-wide"
      },
      {
        "description":"greeting,thanks, apology",
        "type":"social",
        "area_used":"East-Asia"
      }
    ]
}
```

5.2. Data Format

It is important that the data is structured and sectioned. A simple way is to have directory with subdirectories with the name of the class and contain data for each class. It

gives the opportunity to label the data this way during the process, iterating through the subdirectories.

5.3. Dataset

Despite the big coverage of gesture recognition in computer vision, unfortunately there was no dataset available that covers interpretable gestures in terms of body language and human-human-communication. Most datasets were limited to hand gestures aiming to be used as a input tool for human computer interaction.

Originally only web scraping was planned, but because the performance was too poor to make satisfactory conclusions, the experiment was extended with a separate dataset. From this point, we pursue two approaches to create a new dataset from scratch: web scraping and manually creating a dataset from a camera.

5.4. Data File Processing

Data pre-processing is an essential part of machine learning as well as transforming real world data into a usable format. Real world data is usually incomplete or contains errors which data pre-processing overcomes these pitfalls. The most evident problems are: inconsistency, noise and incompleteness. If this corrupted data is passed into the model unmodified, the network will produce noisy inconsistent results.

Using image-processing is a necessary method to transform an image into a numpy array which can be fed into a CNN. Using this method while iterating through the image dataset and embedding the transformed files into a dataframe, prepares the data for the classification process.
For the approach using pose-estimation to extract body-keypoints from images, the model only needs to learn the body-keypoints instead of the whole image file. The first step is iterating through the whole image dataset and run OpenPose on it. This step returns an image with the estimated skeleton drawn on it and the coordinates of every detected joint are saved as JSON files. The exported images are irrelevant for creating the model, so creating those images could be skipped to save space, but they are a good indicator to analyse how good OpenPose actually estimated by visually comparing if the joints are estimated at all and if they are, then if they are estimated properly. The script exports a JSON file for each person, so skeletons could be removed if they are not wanted. To collect all the information from the single file, there is a dedicated script.

5.5. Network Choice

Out of all the algorithms examined in the literature review and fundamentals, ResNet50 trained on "ImageNet" and OpenPose will be used to extract features. Those approaches provide the most promising results as demonstrated by existing studies as seen in 2 - the most prominent being network performance. For the learning algorithms, all algorithms will be tested out because they are easy to implement with scikit-learn, which give the flexibility to facilitate the variations in variables for the experiments. Depending on their performances from relatively basic hyperparameters, the most accurate models will be used for further experiments. Finally, it is a unique network that provides plenty of opportunity

for improvements. Scikit-learn will provide the ideal tools to easily implement, customise, optimise and evaluate the networks.

5.5.1. Training

Training is used to ensure the network recognises patterns in the data. Traditionally, training data is 80% of the original data. Because we have two different datasets and two different networks, the ratio varies: the amount of images in the datasets are 3440 (web scraping) and 8175 (manual). The amount is different for OpenPose, because on the one hand, one image can have multiple people whose skeleton is computed and on the other hand, skeletons with missing essential joints will be removed and some images don't include people to estimate. So the datasets for OpenPose have 2110 (web scraping) and 9582 (manual) samples for training. Additionally, aiming to illustrate how those datasets influence each other, they can be combined (11615 images and 11692 skeletons). The training process passes the training data into the network and the weights and bias are updated for each model.

5.5.2. Validation

Validation is used to determine bias and variance by using data for evaluation that was not seen in the training process. Bias determines how accurate the model is expected to perform compared to how it actually performs. Variance describes how sensitive a network is by how much the peformance of a model fluctuates with repeating the learning process with minor controlled changes in the training set. However, validation data is not used for training. This project has no need for a dedicated validation set, because of K-Fold Cross Validation 5.5.3.

5.5.3. Validation with K-Fold Cross Validation

One of the most popular and common techniques to evaluate and select a model in machine learning is k-fold cross validation. The concept behind cross-validation is to make it possible for every sample in the dataset to be tested. K-fold cross validation is a specific type of cross validation that iterates k times over a dataset. The data will be split into k parts and for every iteration one of those parts will be used for validation once while the remaining k-1 parts are fed into the learning algorithm. With this cross validation method, k different models will be fitted which was validated on non-overlapping validation sets and return different performance values. The cross validation performance is measured by the arithmetic mean of the performance values.

5.5.4. Testing

Testing is used to assess the performance of the model. Testing is traditionally 20% of the total data. In this project, the total amount of testing data is 24 images per gesture: 1176. All images show exactly one person performing the gesture, but due to false pose-estimation the total amount in OpenPose is 1193. The testing data is separate to training data. It is used to validate the performance of the network. The data is only used when the whole model has been trained with all the available training data. Testing data must be different to the validation data to ensure false results do not occur.

5.6. Overfitting/underfitting

As explored in the fundamentals, the network is learning a target function from training data. Generalisation is a measure of how well the network learned the trends in the data. The network should then be able to perform well on an unseen set of data in the same problem domain. However, overfitting is the largest problem in machine learning [54]. Overfitting is when the train data has been learnt by the network too well, meaning also learning the noise and specific details. So the network rather memorises than tries to learn the general patterns. If the network learns noise, it will incorrectly make future predictions, but overfitting can be solved by having more data examples in the training set and ensuring the network is not over complex. In contrast to overfitting, underfitting can happen if the classifier is not able to recognise patterns from the fed data.

5.7. Programming Environment

The project will be written in Python 3.8.3 Anaconda custom. The editor by choice is Atom and codes will be run through Command Lines on Terminal (Mac OS). Atom allows the user to write code in a human-friendly way. As this project may become complex very quickly, every decision should be made to increase code readability. The university's HPCs' Devboxes from exchange@research.tib.eu serves for high computational tasks and offer GPU-usage.

5.8. Metrics

Metrics are measurement tools that evaluate a classifier's performance. Each metric evaluate different characteristics of the model trained by the classification algorithm. [55] This section looks at the most relevant metrics to this project. All metrics can be accessed with scikit-learn functions from the library sklearn.metrics. A clearly arranged overview can be shown with a classification report which contains all the following metrics for each class and the classifier's average. To get a more detailed insight, normalised confusion matrices can be reviewed.

Accuracy

Accuracy is ratio of the correctly predicted observations to the total predicted observations for a test. y represents the true values while $\hat{y}$ represents the predicted values:

$$accuracy(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

Precision

Precision is a ratio of correctly predicted positive (true positives TP) observations to the total predicted positive observations (true positives TP and false positives FP) in a class. Higher precision rates suggest lower false positives.

$$precision = \frac{TP}{TP + FP}$$

Recall

Recall is a measure of sensitivity. It is a ratio of correctly predicted positive observations (true positives TP) to all the observations (true positives TP and false negatives FN)in a class. It shows how many true predictions the system did recognise for a class.

$$recall = \frac{TP}{TP + FN}$$

F-score

F-score is a measure that take into account precision and recall. It is the geometrical mean of precision and recall. Because the value precision and recall is equal in this project, F1-score is used.

$$F_1\text{-}score = 2 * \frac{precision * recall}{precision + recall}$$

6. Implementation

This chapter explains how the final model was built and how the image dataset was prepared. Using the classic image-classification and pose-estimation, a prediction was produced for gestures in images. This is an important chapter to appreciate the methods used as it allows reproduction and refinement for future projects.

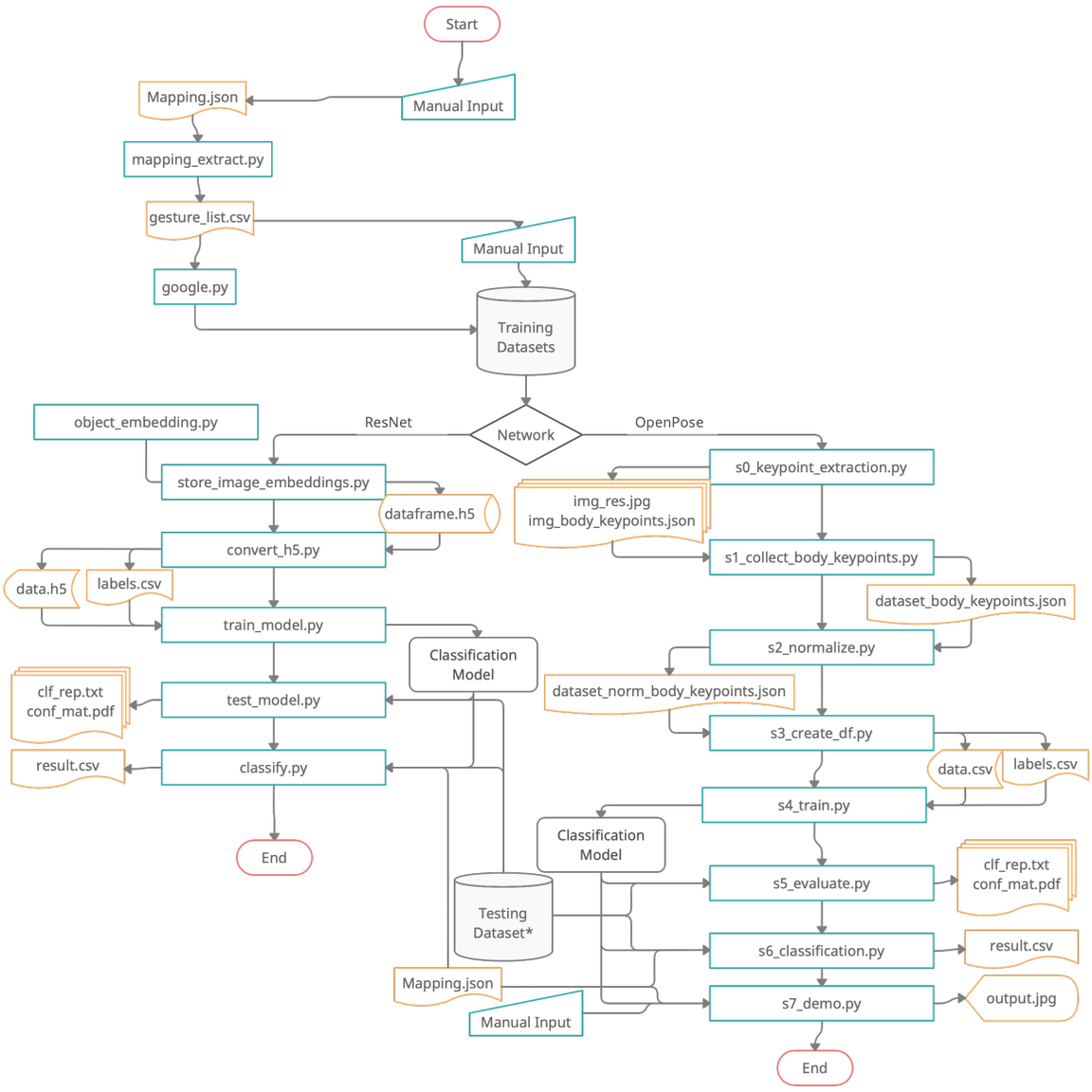


Figure 6.1.: Flow chart of the implemented networks.
*Testing Data is already processed for simplicity.

6.1. Data Collection

In order to obtain a substantial amount of data enough for the networks to produce satisfactory and reliable results, the following two methods were conducted:

Data Collection with Web Scraping

Aiming to get a wide variety of examples of gestures, Web Scraping is a method to support this approach. The term "web scraping" (synonyms: Screen Scraping, Web Data Extraction, and Web Harvesting [56]) might be confused with "web crawling". Even though both procedures are similar, there is a significant difference: web scraping means automatically extracting specific web data of interest, for example, for retrieval or analysis, while web crawling means automatically searching for more information and content, for example, used by search engines [57]. Consequently, it saves the time and effort to manually search through the internet and download single images for this project. Using web scraping images of each gesture can be retrieved. A gesture list serves as input for the script which uses Google Search [58] as the base URL.

For the first procedure a list "gesture_list.csv" needed to be generated to extract the "gesture_name" from the 5.1 stored in "Mapping.json". Using that list, data was generated from web scraping with a combination of icrawler, BeautifulSoup and Python libraries and sorted in subfolders containing the gestures, which will be use as labels later. Because the usage of the gesture name alone returns a lot of noise, for example "wave" gives us mostly water waves from the sea, the search needed to be more specific. Therefore, for the search query, "full body gesture" was added. In addition to that are representations of the neutral classes "standing straight" and "sitting". Logically, humans can perform gestures while standing straight or sitting, but for this project we limit the feature to only try to recognise one gesture per person/image and we assume that when a person is not standing straight or sitting, they are performing a gesture. The maximum amount of each picture to be downloaded is set to 500, but this limit is likely to not reach. Each class has between 50 to 95 image examples. Even after limiting the search entries, the results still contain a lot of noise, which will likely worsen the network's performance.

Manual Data Collection in controlled environment

In order to collect a dataset of images for each gesture, 5 people of different height, sex, skin-color and body-type, were willing to perform gestures as subjects for this project. Instead of taking single still-images it was more efficient to take videos, which can be broken down to single frames later. To reduce computational effort, the record settings were set to the lowest resolution possible on the camera: 720p (1280*720 pixels). The only available frame rate was 60 frames per second (fps). To reduce motion blur which can make pose estimation harder, it is recommended to use high shutter speed. For this set up it was 1/100 s. It is important to differntiate between frame rate and shutter speed, because frame rate only means how my frames the video has per second, but not necessarily how fast the image will be captured. Shutter speed means for how long the camera is letting light in before it closes the shutter. The shutter speed could have been set faster to remove motion blur visibly completely, but because of the poor light situation the lights needed to be turned on and caused flickering at faster shutter speed. To avoid this issue, shooting outside with good window light or with professional lighting is recommended. The lens that was used

was a professional zoom lens so it is easier to switch between a full-body and a close-up shot. Every gesture was recorded around 10 seconds and the subject performed the gesture in different directions in the radius of 180° (from side to side and never facing the back) and continuously to make sure we have different angles and no frames without performing the gesture. Another additional aspect that could have been considered is tilting the angle with low and high angle and rotation, which might lower accuracy for unusual framing. In order to have more variations in the data, the whole process was shot handheld and with slight camera movement, zoom in and out.

6.2. Experiment 1: ResNet

It has been decided that for this project, a pre-trained CNN for Object Detection with a ImageNet-model trained by ResNet 50 will be used. By returning the "avg_pool"-layer we can extract the image features.

Image Pre-Processing for ResNet

Using ResNet50 the data needs image pre-processing which Keras provides libraries for. Therefore, first loading and resizing the image to 224x224 pixels is necessary for pipeline processes which require images of the same size. After expanding the array, the image can be pre-processed, returning a encoded tensor/numpy-array representation of the image ready for the model to be processed.

Feature Extraction from Average Pool Layer

The main reason why this network doesn't use a CNN with fully connected layers is because they are more likely to be affected by overfitting. That means to achieve generalisation for the network a large amount of data is needed, which is not available in this project. Although the fully connected layers of the CNN are not helpful, it can be taken advantage of by the global average pooling of the CNN which has proven to improve the generalisation ability and prevents more overfitting [59]. That means instead of generating on a feature map for each corresponding category of the classification from the convolutional layers, it can take the average of each feature map and be used in another machine-learning algorithm to classify. How it is realised in Keras is by initialising the ResNet 50 model, but only used as a base for a model which outputs the average pool layer, which returns the 2048 features which will be embedded into a data frame for further processing.

6.3. Experiment 2: OpenPose

This section has a detailed description how the second experiment: classifying by using pose-estimation for feature extraction will be conducted.

Image Pre-Processing for OpenPose

As for Experiment 1, this experiment needs resizing too, but for maintaining the width/height-ratio, it is scaled only so that the height of the image is resized to a height of 368 pixels and then image padding from the bottom right corner fills the image up to a 368x368 image, which gives the kernel of the CNN more space to cover the image and results in a more

accurate analysis of the image. Also in this network, the numpy-array will be pre-processed, returning a encoded tensor/numpy-array representation of the image ready for the model to be processed.

Pose Estimation: Body Keypoint Extraction

For the pose estimation, this network uses a OpenPose Model for body detection. Because Cao et al. [21] OpenPose is run on Caffe, the model needed to be converted for PyTorch by using caffemodel2pytorch [60]. It returns an output from two convolutional layers containing heatmaps and part-affinity fields. By using this, each joint will be computed and can be drawn as a skeleton in the picture to see the result. For extracting the data, the coordinates of each joint's coordinate will be saved. Due to the separate models for body and hand estimation, it is necessary to allocate a hand to the correct wrist of a body. The program only looks for a hand if shoulder, elbow and wrist were estimated for either the right or left side of the body. Using this information, the position of allocated hand can be limited by a rectangle calculated with the coordinates of shoulders, elbow and hands. This ensures that hands that are connected to the wrist will be correctly allocated. The model for hand pose estimation will be used in the same way to the body, but only on the calculated excerpt. A problem that is needed to keep in mind is that it might be difficult to separate hands, which can be together or close to each other.

Figure 6.2.: Examples failed hand pose estimation when close together: "air kiss" and "kung fu salute"

Finally estimated, the 60 coordinates (18 for the body and 21 for each hand) are brought together for the joints of the body and both hands yield the 120 extracted features.

Feature Selection

In order to fix incompleteness of the skeletal body, every estimated body without a neck and at least one hip gets rectified as we need these joints for normalisation. To make the data simpler, unimportant features like face keypoints (nose, eyes and ears) and leg keypoints (knees and ankles) will be removed in order to achieve better generalisation ability. The face itself has no significant meaning in this project's gestures, but with legs we risk that individual gestures, for example "curtsey" might lose accuracy. This loss will be taken into account. This step reduces the data to 102 variables.

Normalisation

The image data was already normalised during preprocessing for the model, but to bring body and hands together and allocate them to a person, it is easier to do it from the

output of the pose estimation, where the coordinates are represented as positions in the original image in pixels. Moreover, the results is more understandable with this method. Therefore, this output needs to be normalised again manually. Due to the fact that the body keypoints are locally relative to the first pixel of an image, it's necessary to center each body; in this project the centroid is the neck. To achieve that, it is only needed for the subtraction of every body keypoint's coordinate with the neck's coordinate. If there is no neck nor any hips estimated, that body has been already removed from the dataset in previous step. The hips are necessary in the next step: For some cases, there might be missing keypoints as a consequence of the framing of the picture or poor estimation, but to avoid incompleteness due zero-values, we generate artificial keypoints, by estimating relatively to the keypoints that are correctly generated. For example, the left shoulder is supposed to be on the opposite side of the neck. As a default pose for the arms which is used as a reference, we define a straight standing person. For a missing hip we assume that there is no major twists so knowing that it's mostly parallel to the shoulders, we can estimate the missing hip. Furthermore, centering the body is not enough for generalising, because it only takes into account 2-dimensional space and ignores distance. (There are also the aspects of lens distortion and perspective distortion, but their influence is very special and small enough that we can ignore it.) Because of that, the data has to be scaled so they are easier to compare. Therefore, we use the distance between the neck and the hip (middle between right hip and left hip) as the scaling reference and scale the whole body. This normalised data is stored into a data frame which will be used for the classification process.

6.4. Chosen Algorithm

As previously discussed, all mentioned machine learning algorithms will be used for the supervised learning task (Chapter 3.2.1). These algorithms will be sourced from the Python library scikit-learn. All those algorithms can be used with a wide variety of potential hyperparameters, but for the first tests, it will be mostly default settings.

6.4.1. PCA

To determine whether PCA is useful for the datasets, PCA will be applied before feeding the data into the classifier. To limit the amount of tests, only reducing the amount of features by 50 % will be tested and the best result is going to be compared against the initial set up.

6.4.2. Hyperparameter Optimisation: Grid Search

Grid search is a traditional, but also popular search strategy that is used in this project to optimise the hyperparameters for the classifier models [32]. Hyperparameters are the parameters that control how the machine learning algorithm is behaving. Aiming to determine the most optimal set of hyperparameters, Grid Seach demands defining a subset of various possible learning rates or different constraints that influences each classifier. Since this process is a brute-force algorithm, it runs through all provided combinations of hyperparameters, creates the models and compares them to decide the set with the best scores. Compared to random search, it's computationally less efficient, but since the search space is not too large, it guarantees the best result. In addition to that, it is possible to use

k-fold cross validation for model selection, instead only for model evaluation, by combining that method with grid search. Grid search can apply k-fold cross validation on the training set. Using grid search takes a multiple amount of computing time for the repeated training process, because it has to train each model k times first, before it can return the optimal set. [61]

6.5. Executional Program

In order to execute the trained model on chosen data, a program will be coded which applies the model on given dataset and returns as a csv-file of each image or skeleton of each class with their respective prediction probability scores. Additionally, a program was made only for OpenPose that is able to run all steps from image data to classification with visualisation of the pose-estimation and classification information with gesture interpretations.

7. Results and Discussion

The following section presents the findings of the data collection, the network creation and the recognition performance from the experiments. The recognition will be assessed using the technical metrics 5.8. A variety of gestures were used to test the systems performance which are listed in the Appendix A with a description of their meaning. All the training data acquired from web scraping was available data from the year 2020. All of the training data was passed into each of the networks tested and the average accuracy of the test data and training data was taken. Afterwards, the results and findings from the experiments will be discussed concluding what further work can be done for this research and outline what problems and limitations were encountered and how to avoid them.

The primary goal of the experiment was to see how the accuracy of the recognition varied with each gesture and the accuracy of the predictions made by each network. Each of the 49 gestures were used as training data on each network to which the results from this were recorded. Aiming to illustrate the performance of the network, the recognition accuracy of both datasets on each network will be compared. Subsequently, a comparison of how the networks perform if the two datasets are combined to determine what impact this approach has hence understand how the experiments can be improved in the future.

7.1. Data Collection

Because the manual dataset was created in a controlled environment, there were no issues. The only thing noted later is that the dataset was not balanced perfectly, but due to the amount of samples available for every class, this can be ignored.
In contrast to the manual dataset created, the web scraping dataset caused many problems: finding the right keywords to get optimal result is not an easy task and if one finds them, they might not return many samples. The issue with not finding the right keyword is essential, because this determines the amount of noise that will end up in the dataset. In this project, the dataset was full of noise, like images of other gestures and unrelated images. This issue could not be solved in this project but to see if the networks can deal with the noise, it will still be used.

7.2. Final Networks

Due to the time constraints of this project, the final networks that the datasets will be passed through were determined before executing the experiments. From the experiments, an optimal algorithm will be chosen which will then go through a number of attempts of optimisation.

Chosen Algorithm

To determine which classifier provides the best accuracy to be used for the baseline model, all the learning algorithms outlined below were applied on the combined dataset, since this

dataset contained the largest available number of data. Therefore, the results would be more explicit.

Table 7.1.: Classifier accuracy results.

Network	**5-NN**	**SGD**	**Linear SVM**	**RBF SVM**	**Decision Tree**	**Random Forest**	**Neural Network**	**Naive Bayes**
ResNet	7 %	10 %	14 %	**17 %**	5 %	11 %	14 %	13 %
OpenPose	49 %	37 %	35 %	35 %	19 %	**56 %**	54 %	33 %

From Table 7.1, the results show that for all classifiers, OpenPose accuracy percentage levels are at least twice the accuracy percentage levels of ResNet with the biggest difference in 5-NN with OpenPose being 7 times more accurate than ResNet. For ResNet, the least accurate performing classifier was the Decision Tree at 5% and the highest was the RBF SVM at 17%. For OpenPose, the Random Forest classifer performed best at 56% whilst the worst classifier was the Decision Tree at 19%. From this result the algorithms for further optimisation and experiments will be chosen: RBF SVM for ResNet and Random Forest for OpenPose. It might be possible that optimising other algorithms will produce accuracy results larger than RBF SVM and Random Forest, but to test all possible combinations would not be possible with the time constraints of this project.

Baseline

In this experiment, RBF SVM and Random Forest were used as the baseline classifier for their respective networks which trained the model with default hyperparameters and no PCA with the web scraping, manual and combined dataset. The default settings for RBF SVM are (C=1.0, kernel='rbf') and for Random Forest (max_depth=None, n_estimators=100). The overall accuracy scores for the experiments on all 3 dataset variants are shown below:

Table 7.2.: Baseline accuracy results.

Network	**Classifier**	**Web Scraping**	**Manual**	**Combined**
ResNet	RBF SVM	5 %	16 %	**17 %**
OpenPose	Random Forest	24 %	54 %	**56 %**

From the above table, the findings suggest that accuracy results were the highest for both recognition networks when both the web scraping and manual dataset were combined, performing at 15% with ResNet and 56% with OpenPose. Using the accuracy of the baseline results gives a reference value to determine how effective each optimisation step is.

PCA

The highest performing classifiers for each of the networks were used to perform a PCA on the data sets in order to identify the hyperparameters, which improve the scores further. In both networks, reducing the dimensions by half gave the best result using PCA.
The scores are shown below:

Table 7.3.: PCA accuracy results.

Network	**Classifier**	**Dimensions**	**Accuracy (no PCA)**	**Accuracy (PCA)**
ResNet	RBF SVM	2048	**17 %**	14 %
OpenPose	Random Forest	102	**58 %**	53 %

Surprisingly, there is no improvement; instead, there is a slightly decrease in accuracy. It's clear that initial setups without PCA are the best ones to use.

Grid Search

The highest performing classifiers for each of the networks were used to perform a grid search in order to identify the hyperparameters, which improve the scores further.
The scores are shown below:

Table 7.4.: Grid search accuracy results.

Network	**Classifier**	**Accuracy (Default)**	**Accuracy (Optimised)**
ResNet	RBF SVM	**17 %**	15 %
OpenPose	Random Forest	56 %	**58 %**

Comparing the networks both return different results. While optimisation with grid search worsens the accuarcy for ResNet by 2 %, it improves OpenPose by 2 %. In other hyperparameter choices, the performance of the classifiers drops by small amounts. The parameters we initially started with are usually not the best ones to use, so it's proposed to try hyperparameter optimisation, but for ResNet we will keep the initial settings. Furthermore, for networks like ResNet who take long for training the classifier, it is suggested to use random search instead to improve efficiency.

Evaluation with K-Fold Cross-validation

Aiming to evaluate the robustness of the networks, K-Fold cross-validation was applied on the optimal configurations. For this process, the training data was randomly split into 5 folds using shuffle split. The scores are shown below:

Table 7.5.: K-Fold cross-validation results.

Network	Classifier	Accuracy	CV-Accuracy	Variance	Bias
ResNet	RBF SVM	17 %	59 %	±1%	42 %
OpenPose	Random Forest	58 %	70 %	±1%	12 %

Although both networks return similar results from the cross-validation, they must be interpreted differently. Firstly, a look at the variance gives us information about the networks' sensitivity towards the data: both networks have a low variance of 1 % which gives the conclusion that the sensitivity towards the data is low. Comparing the accuracy of the cross-validation and the actual accuracy from the test, we can determine the optimism bias by the difference between them. It is expected that there is a certain degree of optimism bias when the testing set is independent from the training set, however, results differ distinctively: 42 % bias for ResNet while OpenPose has 12 % bias. Not only is the difference in bias large, but the low accuracy of ResNet is more than double, while for OpenPose it is less than a quarter. Therefore, ResNet fails to generalise it's classifier leading in overfitted predictions. OpenPose on the other hand is more capable of dealing with unseen data.

7.3. Analysis

This section analyses how well each model performs the classification task with a dedicated testing set independent from the training data. The full testing results can be found as classification reports in Appendix A. For better visualisation, the rows are coloured: grey for not predicted classes, light grey for classes with a F-score under 10 % and apricot for classes with an F-score under 20 % . All other classes are considered useful and more colours were added for those mentioned in the analysis: green for optimal scores, lime for good scores, cyan for high precision and low recall and yellow for the opposite.
Giving an overview, the table below shows all settings of the experiments and the overall accuracy:

Table 7.6.: Final accuracy results.

Network	Classifier	Web Scraping	Manual	Combined
ResNet	RBF SVM	5 %	16 %	**17 %**
OpenPose	Random Forest[1]	24 %	55 %	**58 %**

1 optimised with proposed hyperparameters from grid search

7.3.1. Experiment 1: ResNet

ResNet worked best with initial setup for a SVM using RBF-kernel, with no PCA and no hyperparameter optimisation. Thus, it can be assumed that this setup was optimal from the beginning and didn't needed any tuning. It's needed to be mentioned that the SVM training process with those 2048 features took the longest of all learning algorithms, which needed to be considered when replicating the experiment.

Web Scraping Data

The overall accuracy for the Web Scraping Data Set is 5 % and failed to classify 36 out of 49, while 7 classes have a F-score under 10 % . For reasons of convenience this threshold will be used to declare when a class was predicted, but very poor. The classes with the highest scores are "sitting" (32 % precision, 38 % recall, 35 % F-score) and "shrug" (18 % precision, 46 % recall, 26 % F-score). Although the scores are low, "sitting" has balance between precision and recall, which indicates a relatively equal distribition of false positives and false negatives. "Shrug" on the other hand, has a high recall compared the its precision, which shows that the class was predicted with more false positives. It means that the classifier is to optimistic classifying classes with a higher recall.
Overall this classifier performs poorly and even it's best predicted classes are not accurate.

Manual Data

The classification accuracy for the Manual Data Set is 16 % the model was not able to classify 12 gestures. Furthermore, 16 classes have very low F-scores. The classes with the highest precision are "money" (100 % precision, 4 % recall, 8 % F-score), "big" (91 % precision, 42 % recall, 57 % F-score) and "curtsey" (86 % precision, 25 % recall, 39 % F-score). "Money" has a perfect precision score, but the low recall means that the amount of false negatives is very high, while there are no false positives. This relation of high precision and low recall is still an indicator for a classifier being too careful to predict this gesture. Compared to "money", "big" still has good recall and "curtsey" has mediocre recall. "Open palms" (32 % precision, 71 % recall, 44 % F-score) which is an example for the opposite. "Sitting" (50 % precision, 54 % recall, 52 % F-score) has good but also balanced scores and it is predicted better than previous experiment.
In conclusion, this model is also performing poorly, but clearly better than previous model.

Combined Data

This model's accuracy score trained on the Combined Data Set is 17 %. 15 classes were not predicted at all while 8 of the rest has low F-scores (under 10 %). The classes with the highest scores are "sitting" (70 % precision, 58 % recall, 64 % F-score), "big" (63 % precision, 50 % recall, 56 % F-score) and "TT" (48 % precision, 50 % recall and 49 % F-score). This relation of precision and recall are quite balanced for those classes. Even though the total scores are slightly better for this model, compared the model trained on the manual data, but since it didn't predicted 3 classes more it can be argued if it is really better.
As a result of all three variations, none of those classifiers is declared useful, due to poor ability doing the classification task.

7.3.2. Experiment 2: OpenPose

For OpenPose the best performance is achieved learning with Random Forest. The model was trained with a maximum depth of 40 and 700 estimations while set on "auto" for maximum features. Due to hyperparameter optimisation with grid search, the improvements in the metrics were around 0-2 %, so it is an method with small impact, but should not be skipped. Because the training process with data extracted from OpenPose grid search

is less complex, it was definitely more time-efficient than in Experiment 1 7.3.1 and gave better improvements.

Web Scraping Data

This experiment's model failed to classify 15 classes and the overall accuracy is 24 % and 6 classes had an F-score below 10 %. When it comes to accuracy this model outperformed all models from Experiment 1 using ResNet and it predicted more classes than the ResNet model trained on the same data set. The best predicted class with balanced scores was "air quotes" (67 % precision, 75 % recall and 71 % F-score), which means the amount of true positives was quite high while false positives and negatives were kind of equally low. From Experiment 1 also no class was predicted that well. Other high scoring classes were "thumb up" (46 % precision, 79 % recall, 58 % F-score), "big" (82 % precision, 58 % recall, 68 % F-score) and "bow" (37 % precision, 79 % recall, 50 % F-score). In a direct comparison, this model outperforms the ResNet model clearly on the same data set.
Overall, it still doesn't predict 31 % of the classes.

Manual Data

The model performed with a overall accuracy of 55 %. The average metrics are 55 % precision, 55 % recall and 54 % F-score. Every class was predicted by the model trained on the manual data set. Furthermore there is only one class with a F-score below 10 %: "blah-blah" (20 % precision, 4 % recall, 7 % F-score). The best predicted classes are "cupped hands" (96 % precision, 100 % recall, 98 % F-score), "air quotes" (77 % precision, 100 % recall, 87 % F-score) and "surrender" (84 % precision, 88 % recall, 86 % F-score). Except for mentioned classes there F-score ranges from 17 % - 83 %.
In conclusion, the model from this experiment outperformed every previous by scores, amount of predicted classes an amount of low score classes by far. It proved itself more useful for given classfication task, but with some classes as weaknesses.

Combined Data

The model performed with a overall accuracy of 58 %. The average metrics are 58 % precision, 58 % recall and 57 % F-score, which shows that combining two dataset can improve overall accuracy scores, although there is a big quality gap between those data sets if one compares the experiments where they are seperate. Every class was predicted by the model trained on the combined data set and there is not only one class with a F-score below 10 %, which we determined as a low accuracy class. The lowest F-score were predicted by "blah-blah" (18 % precision, 8 % recall, 11 % F-score). The best predicted classes are "cupped hands" (100 % precision, 100 % recall, 100 % F-score), "air quotes" (77 % precision, 100 % recall, 87 % F-score) and "surrender" (88 % precision, 92 % recall, 90 % F-score). So the lower end and the higher end stay the same classes. Except for mentioned classes there F-score ranges from 23 % - 84 %.
In conclusion ,the model from this experiment is even more useful than the others and knowing that this experiment is our final network, it is worth it to take a look at the confusion matrix A.1 which can give a better insight into how the model behaves. Looking at the main diagonal (bottom-left to top-right) which represent the predicted true positives shows the recall of the model predicting each classes. The darker the colouring the better the recall. This information was already shown looking at the classification report, so the

interesting deductions can be made from the false predictions, which lie outside the main diagonal. The highest false predictions were made for "kung fu salute" as "air kiss" and "money" as "L" with 42 %. The amount of false predictions between "kung fu salute" and "air kiss" illustrate how the model faces difficulties differentiating them from each other, but only in one direction, because with the other way around, this behaviour was not shown. This could be the case, because in some cases both gestures can have similar hand positions. On the other hand, there is not such a relationship found between "pointing index finger" and "OK", except they are both hand gestures. Another interesting observation is that "hands in pockets" was falsely predicted as "hands on hips" with 32 %, which might be due similar wrist position. Other than mentioned observations, there are no clear relationships visible.

7.4. Discussion

Both of the proposed designs ResNet and OpenPose were implemented and tested. Although they don't reach state of the art results, once all design proposals were combined, at least OpenPose produced usable results for gesture recognition in a practical environment. Unfortunately, ResNet performed significantly worse due to being unable to cope with global transformations, so that none of the 3 models are actually useful, but it showed how more training data and especially more training data in a controlled environment can make significant improvements. The same is shown in Experiment 2, where the model trained on manual data has more than double the accuracy than the model trained on web data. It's impressive how a small data set, with just 5 subjects, is enough to make usable predictions for 49 classes, which also have many similar gestures. So knowing that training on the Manual Data Set is way more effective, the interesting question is how the model behaves when you combine those two data sets. Because they are not equally distributed, it's hard to make hard assumptions, but there were two outcomes expected: either the accuracy will decrease because the classifier gets interfered by the noise of the web data or the web data contributes enough of good data to the classifier ending up enhancing its performance sufficient to outweigh the noise. Reviewing the findings, it can be said that combining these two data sets resulted in improving the classifier.

7.5. Visualisation

As a result of researching the interpretable gestures, collecting the the data, processing the data and create a machine learning model, the results can be brought together in a program that shows the results visually:

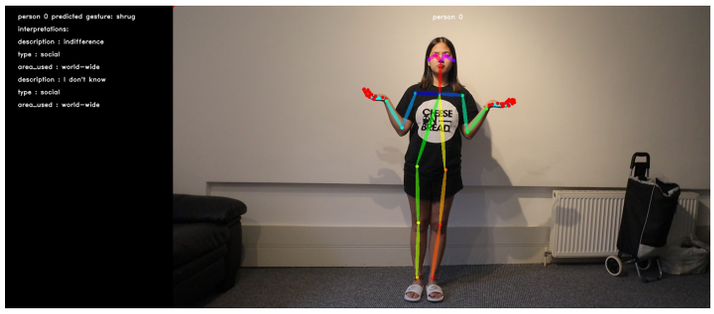


Figure 7.1.: Example of visualisation output.

7.6. Requirements Achieved

This section takes a look back through requirements defined in table 4.1, and evaluates the extent to which they were achieved.

- **Algorithm creation (Priority: Essential)** was successfully completed with two different approaches implemented: ResNet and OpenPose. They train with labeled images and at least one of them achieved acceptable accuracy for recognising gestures in images.
- **Algorithm improvement (Priority: Essential)** due to chosen pre-processing methods like normalisation and feature selection, the data was prepared for more successful model creation.
- **Technical indicators (Priority: Essential)** using were optimised by taken advantage of PCA dimension reduction and grid search and cross-validation for hyperparameter optimisation aiming for maximum accuracy.
- **Diversity of Gestures (Priority: Essential)** From a total of 49 gestures the best performing model is able to recognise 35 with Precision higher than 50 %. While most of the previous work found achieved high accuracy, none of them attempted to classify numerous gestures to this extent.
- **Algorithm comparison (Priority: Essential)** was discussed for all supervised machine learning algorithms in 3.2.1 after being applied to the algorithm. Random Forest performed the best in all experiments, which was then used as chosen algorithm for evaluation and analysis.
- **Evaluation of the algorithm (Priority: Essential)** using sklearn.metrics classification_report shows the metrics precision, recall, f1-count for each class and the average and accuracy for the whole testing set (the classification report for each experiment are in A). This is a clearly arranged presentation. A visually more detailed insight give us the confusion matrices, highlighting the behaviour of the algorithm by showing the relationships between predicted classes and true classes.
- **Executional Program (Priority: Essential)** gives us exact information about the classification for every inputted data sample it shows the predictions score for every class so one can determine how confident the model is. Additionally the interpretations from the gesture dictionary will be mapped to the gestures, so with the recognition the interpretations are also returned.

- **Demonstrational visualisation (Priority: Desirable)** is implemented for images that can be dropped and called in one program, without a need to go through every single processing step like the dedicated data sets. The script only need a input and a output and classifies the image returning a image with a skeleton and recognised gesture drawn.
- **Input videos (Priority: Desirable)** was used for creating the data set by breaking down the frames, but a dedicated function for video classification was not implemented, because higher priority task require the time.

- **The overall system** was successfully build in general completing all essential high priority goals. Despite missing one of the two additional features for reason relating to project constraints, this system is still usable and has potential for further research. The following section illustrates the areas that can be improved, but also further areas and aspects that were not dealt with in this project.

7.7. Further Work

From defining objectives through to designing and implementing the project, the practicalities of time, hardware and expertise required to experiment with image classification became increasingly evident. This introduces an abundance of possible further research, given more resources; whether it be time or hardware capacity. There was a variety of indicators available and there was not enough time to test all of the combinations. Further work would need to identify the combinations and test all of them to find the ones that maximise the accuracy. This section outlines some clear areas for expansion of the project.

Datasets

The project could be further developed by changing and preparing a qualitatively better and quantitatively bigger data set for the network. This could be improved by, for example, optimising search results for web scraping, which generates more pictures and is exactly what is searched for. Additionally, web scraping does not has to be limited to Google Search, but can be extended to other search engines or stock photo data bases. Stock photos have high quality and for gestures they often have a neutral background and perform only that on gesture which makes it optimal for a gesture dataset. Even if the photos that are available online have water signs, OpenPose faces no issues handling those images. A big issue with web scraping is the amount of noise in the data that is retrieved. A proposal would be using a object detector to recognise people and crop the excerpts of each person. This way, everything not human will be excluded already. Outlier detection could be another approach to sort out data that deviate from the majority. Furthermore, to expand the data set with useful data is data augmentation. Data augmentation is a method that helps a network achieve higher generalisation abilities, by generating additional data samples through changing available samples from the original data. Islam et al. [16] applied this method on static hand gesture recognition to improve his model's accuracy. A more experimental approach for data augmentation would be using DeepFake [62] to create artificial faces on available images. Such a method could be useful if the amount of people in a data set is small like in this project's manual data set. Such a data set can

be improved by having more people and capture gestures from all possible angles, by using multiple cameras. This way collecting data for a video data set for a RNN could extend the functionality to properly recognising dynamic gestures.

ResNet

An alternative neural network could be implemented to see whether a on ImageNet pre-trained ResNet is the best choice as well as comparing to the alternative pre-trained models like COCO or even other networks like VGG. This project's approach with image classification doesn't recognise each person seperately in a picture yet. This can be solved by separating people in images like in 7.7. However, this approach would require a good method for noise removal, because object detection can detect all people, which means also random people, that might get labeled falsely. Other than that, the results from this experiment didn't give much insight for improvement, but with a good enough data set and implementing all accuracy-increasing features, this network should work well too, like many previous projects in this field.

OpenPose

Since the original OpenPose by Cao et al. [21] was not executable in this project, other works like Schneider et al. [8] were still able to make use of it. Additionally OpenPose can estimate faces too which opens up opportunities to extend the variety of classes with facial gestures that carry interpretable meanings. However, compared to OpenPose, PyTorch OpenPose [51] is really slow and doesn't make good use of the GPU, therefore it is not useful for applications like live video with a camera. For CPU-only applications on old laptops like the one used, it takes around 240 seconds for one image with one person to be estimated. On the HPC when capacity utilisation is low it usually takes 2-30 seconds per image. So hopefully, this research can be continued with faster performing libraries.

A uncomplex way to apply data augmentation with OpenPose is instead of altering the images altering the skeletons by mirroring, rotating and using random changes of single joints, to mimic different body shapes and multiply variations of pose examples.

In this state, the network is able to recognise one gesture for multiple people but a person is not limited to one gesture at a time, so it would be interesting to see if OpenPose is able to recognise multiple gestures like "sitting, face palm, thumb down".

7.8. Problems Encountered

In comparison to the pre-trained ResNet Model which was supported by a lot of relatively overviewable, understanding and designing, the PyTorch OpenPose network system in Python presented a large problem, because it only had the functions to estimate poses and draw the skeletons. Furthermore, how the original code was implemented were bodies and hands were drawn separately made it difficult to understand due to poor documentation and furthermore, it was unusable. After trying other implementations, the PyTorch version was still the only one coming into consideration, because of the ability to estimate hands. Finally, changing the script structure made it possible to successfully detect bodies and hands together in a way that body keypoints and hand keypoints can be allocated to one person. The majority of the project time was spent programming the body-keypoint extraction, the learning algorithms and network setup. However, after preparing the data

for training, the second major problem occured when the network was set up, but the performance was so poor that it was hard to figure out the origin of the problem. It was obvious that noise was a big problem, but also that the images were not made for pose-estimation: from 3440 images, the script retrieved only 2110 samples, that were useful by the network. A good side-effect was that it removed noise. So to ensure that at least the network works in theory, it was decided to create a self-made data set. Another difficulty that was faced was training time with the data extracted from ResNet 50: It was very slow which made evaluation and optimisation processes with cross-validation and grid search took too much time for low performance results. Furthermore, because this project is executed during the world-wide restrictions due to the COVID-19 pandemic, finding subjects for the data set was limited to people from the same household. There are an increasing number of research papers containing gesture recognition networks used in various capacities, but none discuss recognition with many gestures, especially similar gestures.

8. Conclusion

This paper presents several public papers that cover different approaches to classify certain action, gestures and poses with image classification methods and skeletal-based approaches.

Artificial intelligence methods can be used to analyse a digital image and either interpret the image with pretrained features or accurately estimate the pose of a human and interpret its gesture. CNN models are able to successfully classify images and recognise gestures. This project has examined the reduction of digital images with human bodies to simple skeletons. It was estimated that the results would improve as generalisation increased. Using this project alone as a strategy for identifying gestures is feasible. Our networks show the ability to recognise a persons gesture. One problem is that the data available on the web wasn't abundant and accurate. If the desired gesture is not the same as what the network requires then more errors will arise. While the manually created dataset put forward is not considered high quality due its limited amount of subjects and variation, it was still enough to classify a large number of gestures correctly. This project presented a method that opens up the opportunity to research automated gesture analysis in a broader dimension than just a small number of gestures for human-computer interaction, even with a degree of underfitting taken into consideration. Indicators provided a minimal boost to the network accuracy. Ultimately, more advanced recognition algorithms could be used in conjunction with an OpenPose model to further improve success prediction, but the work done here establishes methods for future projects, which can profit from the advantages of OpenPose like its robustness against image noise, poor lighting conditions and estimating multiple subjects just by using RGB images to explore various real-world applications thanks to the broad availability of cameras and smartphones. Those devices are not ready for executing heavy machine learning tasks yet, but with the rapid development and evolution of technology, this makes complex applications available for daily life in the near future.

Bibliography

[1] A. Mehrabian, *Silent Messages: Implicit Communication of Emotions and Attitudes*, 1981.

[2] J. Galván-Ruiz, C. M. Travieso-González, A. Tejera-Fettmilch, A. Pinan-Roescher, L. Esteban-Hernández, and L. Domínguez-Quintana, "Perspective and evolution of gesture recognition for sign language: A review," *Sensors (Switzerland)*, vol. 20, no. 12, pp. 1–31, 2020.

[3] A. Konar and S. Saha, *Gesture Recognition*. Springer, Cham, 2018. [Online]. Available: https://www.springer.com/de/book/9783319622101

[4] S. Saha, S. Datta, A. Konar, B. Banerjee, and A. K. Nagar, "A novel gesture recognition system based on fuzzy logic for healthcare applications," *2016 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2016*, pp. 634–641, 2016.

[5] Merriam-Webster, ""Gesture."," 2020. [Online]. Available: https://www.merriam-webster.com/dictionary/gesture

[6] S. Mitra and T. Acharya, "Gesture recognition: A survey," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 37, no. 3, pp. 311–324, 2007.

[7] V. I. Pavlovic, R. Sharma, and T. S. Huang, "Visual interpretation of hand gestures for human-computer interaction: A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, 1997.

[8] P. Schneider, R. Memmesheimer, I. Kramer, and D. Paulus, "Gesture Recognition in RGB Videos Using Human Body Keypoints and Dynamic Time Warping," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11531 LNAI, pp. 281–293, 2019.

[9] M. Oudah, A. Al-Naji, and J. Chahl, "Hand Gesture Recognition Based on Computer Vision: A Review of Techniques," 2020.

[10] J. R. Pansare, S. H. Gawande, and M. Ingle, "Real-Time Static Hand Gesture Recognition for American Sign Language (ASL) in Complex Background," *Journal of Signal and Information Processing*, vol. 03, no. 03, 2012.

[11] M. Perimal, S. N. Basah, M. J. A. Safar, and H. Yazid, "Hand-Gesture Recognition-Algorithm based on Finger Counting."

[12] A. A. Sulyman, Z. T. Sharef, K. H. A. Faraj, Z. A. Aljawaryy, and F. L. Malallah, "Real-time numerical 0-5 counting based on hand-finger gestures recognition," *Journal of Theoretical and Applied Information Technology*, vol. 95, no. 13, pp. 3105–3115, 2017.

[13] M. Karabasi, Z. Bhatti, and A. Shah, "A model for real-time recognition and textual representation of malaysian sign language through image processing," in *Proceedings - 2013 International Conference on Advanced Computer Science Applications and Technologies, ACSAT 2013*, 2013.

[14] K. B. Shaik, P. Ganesan, V. Kalist, B. S. Sathish, and J. M. M. Jenitha, "Comparative Study of Skin Color Detection and Segmentation in HSV and YCbCr Color Space," in *Procedia Computer Science*, vol. 57, 2015.

[15] H. I. Lin, M. H. Hsu, and W. K. Chen, "Human hand gesture recognition using a convolution neural network," in *IEEE International Conference on Automation Science and Engineering*, vol. 2014-Janua, 2014.

[16] M. Z. Islam, M. S. Hossain, R. U. Islam, and K. Andersson, "Static Hand Gesture Recognition using Convolutional Neural Network with Data Augmentation," Tech. Rep.

[17] W. Tao, M. C. Leu, and Z. Yin, "American Sign Language alphabet recognition using Convolutional Neural Networks with multiview augmentation and inference fusion," *Engineering Applications of Artificial Intelligence*, vol. 76, 2018.

[18] P. Bao, A. I. Maqueda, C. R. Del-Blanco, and N. Garciá, "Tiny hand gesture recognition without localization via a deep convolutional network," *IEEE Transactions on Consumer Electronics*, vol. 63, no. 3, 2017.

[19] K. Vijaya Prasad, P. V. Kishore, and O. Srinivasa Rao, "Skeleton based view invariant human action recognition using convolutional neural networks," *International Journal of Recent Technology and Engineering*, vol. 8, no. 2, pp. 4860–4867, jul 2019.

[20] D. Konstantinidis, K. Dimitropoulos, and P. Daras, "Sign language recognition based on hand and body skeletal data," in *3DTV-Conference*, vol. 2018-June, 2018.

[21] Z. Cao, T. Simon, S. E. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, no. Xxx, pp. 1302–1310, 2017.

[22] C. Sawant, "Human activity recognition with openpose and Long Short-Term Memory on real time images," *EasyChair Preprint*, vol. 2297, 2020. [Online]. Available: https://github.com/chinmayembedded/Human-Activity-Recognition

[23] S. K. Yadav, A. Singh, A. Gupta, and J. L. Raheja, "Real-time Yoga recognition using deep learning," *Neural Computing and Applications*, vol. 31, no. 12, pp. 9349–9361, 2019. [Online]. Available: https://doi.org/10.1007/s00521-019-04232-7

[24] X. Feng, Y. Jiang, X. Yang, M. Du, and X. Li, "Computer vision algorithms and hardware implementations: A survey," 2019.

[25] J. Brownlee, "A Gentle Introduction to Computer Vision," 2019.

[26] R. Klette, *Concise Computer Vision: An Introduction into Theory and Algorithms*, 2014.

[27] J. Ponce and D. Forsyth, *Computer vision: a modern approach*, 2012.

[28] E. R. Davies, *Computer and Machine Vision*, 2012.

[29] A. L. Samuel, "Some Studies in Machine Learning," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, 1959. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp={&}arnumber=5392560

[30] T. M. Mitchell, "Machine Learning and Data Mining," *Communications of the ACM*, vol. 42, no. 11, 1999.

[31] S. Raschka, "Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning," Tech. Rep., 2018.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," pp. 2825—-2830, 2011. [Online]. Available: https://scikit-learn.org/stable/

[33] O. Harrison, "Machine Learning Basics with the K-Nearest Neighbors Algorithm," 2018.

[34] G. James, D. Witten, T. Hastie, and R. Tibshirani, *Springer Texts in Statistics An Introduction to Statistical Learning - with Applications in R*, 2013.

[35] L. Bottou, "Stochastic Gradient Descent Tricks." Springer, Berlin, Heidelberg, 2012, pp. 421–436. [Online]. Available: http://leon.bottou.org

[36] A. V. Srinivasan, "Stochastic Gradient Descent — Clearly Explained !!" 2019.

[37] J. Brownlee, "A Gentle Introduction to Bayes Theorem for Machine Learning," 2019.

[38] Prashant Gupta, "Decision Trees in Machine Learning – Towards Data Science," 2017.

[39] "The Random Forest Algorithm: A Complete Guide — Built In." [Online]. Available: https://builtin.com/data-science/random-forest-algorithm

[40] "Principal Component Analysis explained visually." [Online]. Available: https://setosa.io/ev/principal-component-analysis/

[41] M. Brems, "A One-Stop Shop for Principal Component Analysis - Towards Data Science," *Towards Data Science 2018-Apr-18*, 2017.

[42] Y. Y. Chen, Y. H. Lin, C. C. Kung, M. H. Chung, and I. H. Yen, "Design and implementation of cloud analytics-assisted smart power meters considering advanced artificial intelligence as edge analytics in demand-side management for smart homes," *Sensors (Switzerland)*, vol. 19, no. 9, 2019.

[43] L. Cinelli, G. Chaves, and M. Lima, "Vessel Classification through Convolutional Neural Networks using Passive Sonar Spectrogram Images," 2018.

[44] V. Nigam, "Understanding Neural Networks. From neuron to RNN, CNN, and Deep Learning," 2018.

[45] R. Dechter, "Learning While Searching in Constraint-Satisfaction-Problems," in *Aaai*, 1986.

[46] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, 2018.

[47] Sayantini, "Keras vs TensorFlow vs PyTorch — Deep Learning Frameworks — Edureka," 2019.

[48] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem. IEEE Computer Society, dec 2016, pp. 770–778. [Online]. Available: http://image-net.org/challenges/LSVRC/2015/

[49] K. Dy, J. Ligan, and M. Cabatuan, "Understanding and Coding a ResNet in Keras," 2018.

[50] "GitHub - ildoonet/tf-pose-estimation: Deep Pose Estimation implemented using Tensorflow with Custom Architectures for fast inference." [Online]. Available: https://github.com/ildoonet/tf-pose-estimation

[51] "GitHub - Hzzone/pytorch-openpose: pytorch implementation of openpose including Hand and Body Pose Estimation." [Online]. Available: https://github.com/Hzzone/pytorch-openpose

[52] "GitHub - felixchenfy/Realtime-Action-Recognition: Apply ML to the skeletons from OpenPose; 9 actions; multiple people." [Online]. Available: https://github.com/felixchenfy/Realtime-Action-Recognition

[53] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, "Hand keypoint detection in single images using multiview bootstrapping," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017.

[54] X. Ying, "An Overview of Overfitting and its Solutions," in *Journal of Physics: Conference Series*, vol. 1168, no. 2. Institute of Physics Publishing, mar 2019.

[55] H. M and S. M.N, "A Review on Evaluation Metrics for Data Classification Evaluations," *International Journal of Data Mining & Knowledge Management Process*, vol. 5, no. 2, pp. 01–11, mar 2015.

[56] V. Singrodia, A. Mitra, and S. Paul, "A Review on Web Scrapping and its Applications," *2019 International Conference on Computer Communication and Informatics, ICCCI 2019*, 2019.

[57] "What is the differences between web crawling and web scraping? - Parsers." [Online]. Available: https://parsers.me/what-is-the-differences-between-web-crawling-and-web-scraping/

[58] "Google." [Online]. Available: https://www.google.com/search?

[59] M. Lin, Q. Chen, and S. Yan, "Network In Network," Tech. Rep.

[60] "GitHub - vadimkantorov/caffemodel2pytorch: Convert Caffe models to PyTorch." [Online]. Available: https://github.com/vadimkantorov/caffemodel2pytorch

[61] P. Liashchynskyi and P. Liashchynskyi, "Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS," *arXiv*, no. 2017, pp. 1–11, 2019.

[62] T. T. Nguyen, C. M. Nguyen, D. T. Nguyen, T. Nguyen, and S. Nahavandi, "Deep Learning for Deepfakes Creation and Detection: A Survey," Tech. Rep. [Online]. Available: https://github.com/StromWine/DeepFake

A. Appendix

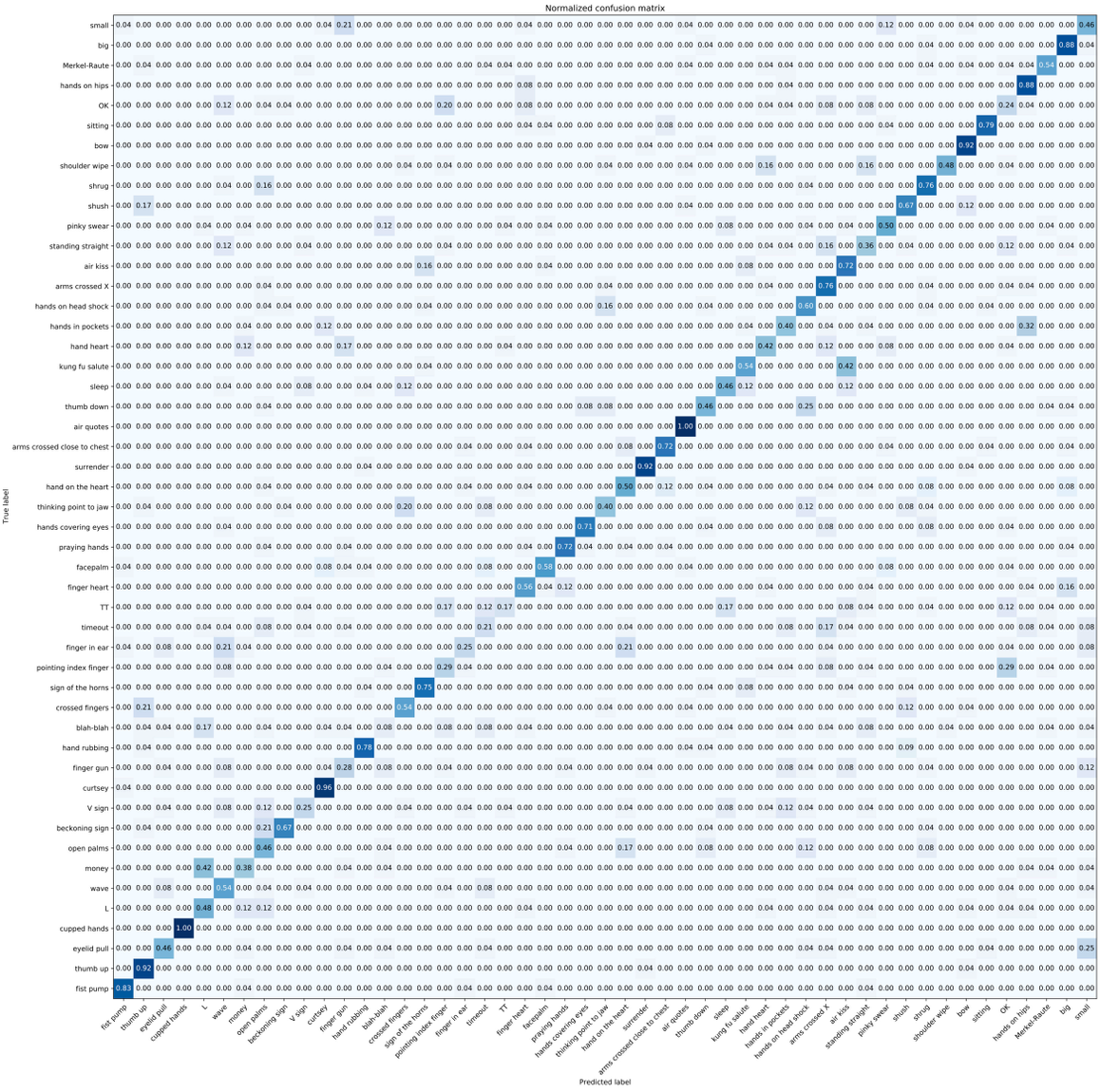


Figure A.1.: Confusion Matrix of OpenPose trained on the combined data set.

Table A.1.: Classification report: ResNet, Web Scraping Data Set.

Class	Precision	Recall	F-score	Support
L	0.0	0.0	0.0	24
Merkel-Raute	0.0	0.0	0.0	24
OK	0.0	0.0	0.0	24
TT	0.0	0.0	0.0	24
V sign	0.0	0.0	0.0	24
air kiss	0.0	0.0	0.0	24
air quotes	0.0	0.0	0.0	24
arms crossed X	0.02	0.25	0.04	24
arms crossed close to chest	0.0	0.0	0.0	24
beckoning sign	0.0	0.0	0.0	24
big	0.0	0.0	0.0	24
blah-blah	0.08	0.04	0.05	24
bow	0.0	0.0	0.0	24
crossed fingers	0.0	0.0	0.0	24
cupped hands	0.0	0.0	0.0	24
curtsey	0.2	0.08	0.12	24
eyelid pull	0.0	0.0	0.0	24
facepalm	0.06	0.33	0.11	24
finger gun	0.0	0.0	0.0	24
finger heart	0.0	0.0	0.0	24
finger in ear	0.0	0.0	0.0	24
fist pump	0.0	0.0	0.0	24
hand heart	0.0	0.0	0.0	24
hand on the heart	0.0	0.0	0.0	24
hand rubbing	0.0	0.0	0.0	24
hands covering eyes	0.16	0.21	0.18	24
hands in pockets	0.0	0.0	0.0	24
hands on head shock	0.0	0.0	0.0	24
hands on hips	0.0	0.0	0.0	24
kung fu salute	0.0	0.0	0.0	24
money	0.02	0.29	0.04	24
open palms	0.0	0.0	0.0	24
pinky swear	0.0	0.0	0.0	24
pointing index finger	0.03	0.04	0.04	24
praying hands	0.0	0.0	0.0	24
shoulder wipe	0.0	0.0	0.0	24
shrug	0.18	0.46	0.26	24
shush	0.0	0.0	0.0	24
sign of the horns	0.0	0.0	0.0	24
sitting	0.32	0.38	0.35	24
sleep	0.11	0.04	0.06	24
small	0.0	0.0	0.0	24
standing straight	0.04	0.25	0.07	24
surrender	0.23	0.12	0.16	24
thinking point to jaw	0.0	0.0	0.0	24
thumb down	0.0	0.0	0.0	24
thumb up	0.0	0.0	0.0	24
timeout	0.5	0.04	0.08	24
wave	0.0	0.0	0.0	24
average	0.04	0.05	0.03	1176
accuracy			0.05	1176

Table A.2.: Classification report: ResNet, Manual Data Set.

Class	Precision	Recall	F-score	Support
L	0.06	0.04	0.05	24
Merkel-Raute	0.25	0.33	0.29	24
OK	0.0	0.0	0.0	24
TT	0.19	0.71	0.3	24
V sign	0.1	0.08	0.09	24
air kiss	0.0	0.0	0.0	24
air quotes	0.4	0.08	0.14	24
arms crossed X	0.25	0.12	0.17	24
arms crossed close to chest	0.25	0.04	0.07	24
beckoning sign	0.0	0.0	0.0	24
big	0.91	0.42	0.57	24
blah-blah	0.0	0.0	0.0	24
bow	0.23	0.58	0.33	24
crossed fingers	0.0	0.0	0.0	24
cupped hands	0.13	0.17	0.15	24
curtsey	0.86	0.25	0.39	24
eyelid pull	0.5	0.04	0.08	24
facepalm	0.0	0.0	0.0	24
finger gun	0.0	0.0	0.0	24
finger heart	0.0	0.0	0.0	24
finger in ear	0.0	0.0	0.0	24
fist pump	0.32	0.38	0.35	24
hand heart	0.2	0.04	0.07	24
hand on the heart	0.0	0.0	0.0	24
hand rubbing	0.17	0.04	0.07	24
hands covering eyes	0.04	0.08	0.05	24
hands in pockets	0.0	0.0	0.0	24
hands on head shock	0.09	0.04	0.06	24
hands on hips	0.33	0.21	0.26	24
kung fu salute	0.2	0.17	0.18	24
money	1.0	0.04	0.08	24
open palms	0.32	0.71	0.44	24
pinky swear	0.06	0.04	0.05	24
pointing index finger	0.08	0.04	0.06	24
praying hands	0.04	0.04	0.04	24
shoulder wipe	0.14	0.17	0.15	24
shrug	0.15	0.12	0.14	24
shush	0.14	0.25	0.18	24
sign of the horns	0.21	0.46	0.29	24
sitting	0.5	0.54	0.52	24
sleep	0.3	0.12	0.18	24
small	0.05	0.25	0.08	24
standing straight	0.13	0.46	0.2	24
surrender	0.25	0.08	0.12	24
thinking point to jaw	0.06	0.04	0.05	24
thumb down	0.04	0.21	0.07	24
thumb up	0.0	0.0	0.0	24
timeout	0.15	0.21	0.17	24
wave	0.02	0.04	0.02	24
average	0.19	0.16	0.13	1176
accuracy			0.16	1176

Table A.3.: Classification report: ResNet, Combined Data Set.

Class	Precision	Recall	F-score	Support
L	0.0	0.0	0.0	24
Merkel-Raute	0.42	0.33	0.37	24
OK	0.0	0.0	0.0	24
TT	0.48	0.5	0.49	24
V sign	0.08	0.04	0.06	24
air kiss	0.0	0.0	0.0	24
air quotes	0.67	0.08	0.15	24
arms crossed X	0.15	0.21	0.17	24
arms crossed close to chest	0.33	0.08	0.13	24
beckoning sign	0.0	0.0	0.0	24
big	0.63	0.5	0.56	24
blah-blah	0.0	0.0	0.0	24
bow	0.31	0.62	0.41	24
crossed fingers	0.0	0.0	0.0	24
cupped hands	0.2	0.21	0.2	24
curtsey	0.57	0.17	0.26	24
eyelid pull	0.0	0.0	0.0	24
facepalm	0.0	0.0	0.0	24
finger gun	0.0	0.0	0.0	24
finger heart	0.0	0.0	0.0	24
finger in ear	0.0	0.0	0.0	24
fist pump	0.46	0.25	0.32	24
hand heart	0.33	0.04	0.07	24
hand on the heart	0.33	0.04	0.07	24
hand rubbing	0.25	0.04	0.07	24
hands covering eyes	0.08	0.12	0.1	24
hands in pockets	0.16	0.12	0.14	24
hands on head shock	0.11	0.08	0.1	24
hands on hips	0.0	0.0	0.0	24
kung fu salute	0.18	0.25	0.21	24
money	0.13	0.17	0.15	24
open palms	0.5	0.33	0.4	24
pinky swear	0.0	0.0	0.0	24
pointing index finger	0.12	0.08	0.1	24
praying hands	0.05	0.04	0.04	24
shoulder wipe	0.06	0.38	0.1	24
shrug	0.28	0.42	0.33	24
shush	0.23	0.42	0.3	24
sign of the horns	0.23	0.42	0.3	24
sitting	0.7	0.58	0.64	24
sleep	0.16	0.21	0.18	24
small	0.05	0.25	0.08	24
standing straight	0.21	0.5	0.3	24
surrender	0.15	0.08	0.11	24
thinking point to jaw	0.0	0.0	0.0	24
thumb down	0.06	0.25	0.09	24
thumb up	0.0	0.0	0.0	24
timeout	0.16	0.29	0.2	24
wave	0.06	0.08	0.07	24
average	0.18	0.17	0.15	1176
accuracy			0.17	1176

Table A.4.: Classification report: OpenPose, Web Scraping Data Set.

Class	Precision	Recall	F-score	Support
fist pump	0.0	0.0	0.0	24
thumb up	0.46	0.79	0.58	24
eyelid pull	0.1	0.38	0.16	24
cupped hands	0.0	0.0	0.0	24
L	0.0	0.0	0.0	25
wave	0.17	0.21	0.19	24
money	0.5	0.04	0.08	24
open palms	0.14	0.08	0.11	24
beckoning sign	0.41	0.54	0.46	24
V sign	0.0	0.0	0.0	24
curtsey	0.0	0.0	0.0	24
finger gun	0.0	0.0	0.0	25
hand rubbing	0.15	0.09	0.11	23
blah-blah	0.0	0.0	0.0	24
crossed fingers	0.6	0.12	0.21	24
sign of the horns	0.09	0.17	0.12	24
pointing index finger	0.08	0.04	0.05	24
finger in ear	0.14	0.21	0.16	24
timeout	0.12	0.17	0.14	24
TT	0.0	0.0	0.0	24
finger heart	0.33	0.48	0.39	25
facepalm	0.11	0.04	0.06	26
praying hands	1.0	0.08	0.15	25
hands covering eyes	0.2	0.04	0.07	24
thinking point to jaw	0.0	0.0	0.0	25
hand on the heart	0.23	0.5	0.32	26
surrender	0.37	0.46	0.41	24
arms crossed close to chest	0.31	0.64	0.42	25
air quotes	0.67	0.75	0.71	24
thumb down	0.0	0.0	0.0	24
sleep	0.09	0.21	0.12	24
kung fu salute	0.0	0.0	0.0	24
hand heart	0.0	0.0	0.0	24
hands in pocket	0.0	0.0	0.0	25
hands on head shock	0.82	0.36	0.5	25
arms crossed X	0.17	0.04	0.06	25
air kiss	0.24	0.68	0.35	25
standing straight	0.27	0.52	0.35	25
pinky swear	0.0	0.0	0.0	24
shush	0.35	0.54	0.43	24
shrug	0.39	0.52	0.45	25
shoulder wipe	0.0	0.0	0.0	25
bow	0.37	0.79	0.5	24
sitting	0.27	0.25	0.26	24
OK	0.25	0.04	0.07	25
hands on hip	0.23	0.58	0.33	24
Merkel-Raute	0.22	0.46	0.3	24
big	0.82	0.58	0.68	24
small	0.12	0.12	0.12	24
average	0.22	0.24	0.19	1193
accuracy			0.24	1193

Table A.5.: Classification report: OpenPose, Manual Data Set.

Class	Precision	Recall	F-score	Support
fist pump	0.74	0.83	0.78	24
thumb up	0.72	0.88	0.79	24
eyelid pull	0.5	0.25	0.33	24
cupped hands	0.96	1.0	0.98	24
L	0.37	0.52	0.43	25
wave	0.35	0.38	0.36	24
money	0.35	0.33	0.34	24
open palms	0.38	0.62	0.48	24
beckoning sign	0.75	0.62	0.68	24
V sign	0.64	0.29	0.4	24
curtsey	0.72	0.96	0.82	24
finger gun	0.33	0.32	0.33	25
hand rubbing	0.73	0.83	0.78	23
blah-blah	0.2	0.04	0.07	24
crossed fingers	0.57	0.5	0.53	24
sign of the horns	0.73	0.79	0.76	24
pointing index finger	0.32	0.29	0.3	24
finger in ear	0.56	0.21	0.3	24
timeout	0.18	0.17	0.17	24
TT	0.44	0.17	0.24	24
finger heart	0.55	0.48	0.51	25
facepalm	0.67	0.54	0.6	26
praying hands	0.67	0.8	0.73	25
hands covering eyes	0.68	0.71	0.69	24
thinking point to jaw	0.42	0.44	0.43	25
hand on the heart	0.48	0.46	0.47	26
surrender	0.84	0.88	0.86	24
arms crossed close to chest	0.74	0.68	0.71	25
air quotes	0.77	1.0	0.87	24
thumb down	0.65	0.46	0.54	24
sleep	0.62	0.42	0.5	24
kung fu salute	0.59	0.67	0.63	24
hand heart	0.44	0.46	0.45	24
hands in pockets	0.31	0.32	0.31	25
hands on head shock	0.44	0.56	0.49	25
arms crossed X	0.35	0.72	0.47	25
air kiss	0.39	0.48	0.43	25
standing straight	0.33	0.2	0.25	25
pinky swear	0.57	0.54	0.55	24
shush	0.58	0.62	0.6	24
shrug	0.57	0.68	0.62	25
shoulder wipe	0.86	0.48	0.62	25
bow	0.62	0.75	0.68	24
sitting	0.86	0.79	0.83	24
OK	0.23	0.24	0.24	25
hands on hips	0.44	0.58	0.5	24
Merkel-Raute	0.75	0.5	0.6	24
big	0.69	0.92	0.79	24
small	0.38	0.5	0.43	24
average	0.55	0.55	0.54	1193
accuracy			0.55	1193

Table A.6.: Classification report: OpenPose, Combined Data Set.

Class	Precision	Recall	F-score	Support
fist pump	0.83	0.83	0.83	24
thumb up	0.61	0.92	0.73	24
eyelid pull	0.61	0.46	0.52	24
cupped hands	1.0	1.0	1.0	24
L	0.43	0.48	0.45	25
wave	0.38	0.54	0.45	24
money	0.45	0.38	0.41	24
open palms	0.31	0.46	0.37	24
beckoning sign	0.84	0.67	0.74	24
V sign	0.46	0.25	0.32	24
curtsey	0.74	0.96	0.84	24
finger gun	0.32	0.28	0.3	25
hand rubbing	0.82	0.78	0.8	23
blah-blah	0.18	0.08	0.11	24
crossed fingers	0.57	0.54	0.55	24
sign of the horns	0.75	0.75	0.75	24
pointing index finger	0.32	0.29	0.3	24
finger in ear	0.55	0.25	0.34	24
timeout	0.28	0.21	0.24	24
TT	0.5	0.17	0.25	24
finger heart	0.56	0.56	0.56	25
facepalm	0.75	0.58	0.65	26
praying hands	0.78	0.72	0.75	25
hands covering eyes	0.85	0.71	0.77	24
thinking point to jaw	0.53	0.4	0.45	25
hand on the heart	0.48	0.5	0.49	26
surrender	0.88	0.92	0.9	24
arms crossed close to chest	0.75	0.72	0.73	25
air quotes	0.77	1.0	0.87	24
thumb down	0.52	0.46	0.49	24
sleep	0.55	0.46	0.5	24
kung fu salute	0.62	0.54	0.58	24
hand heart	0.45	0.42	0.43	24
hands in pockets	0.43	0.4	0.42	25
hands on head shock	0.47	0.6	0.53	25
arms crossed X	0.44	0.76	0.56	25
air kiss	0.46	0.72	0.56	25
standing straight	0.36	0.36	0.36	25
pinky swear	0.57	0.5	0.53	24
shush	0.64	0.67	0.65	24
shrug	0.58	0.76	0.66	25
shoulder wipe	0.92	0.48	0.63	25
bow	0.71	0.92	0.8	24
sitting	0.86	0.79	0.83	24
OK	0.21	0.24	0.23	25
hands on hips	0.57	0.88	0.69	24
Merkel-Raute	0.65	0.54	0.59	24
big	0.68	0.88	0.76	24
small	0.39	0.46	0.42	24
average	0.58	0.58	0.57	1193
accuracy			0.58	1193