Mendel University in Brno
Faculty of Business and Economics

Automated neural network learning for higher accuracy human skeleton detection under realistic conditions

Master's Thesis

Bc. Damián Sova

Supervisor:
doc. Ing. Oldřich Trenz, Ph.D.

Brno 2024

● MENDELU
● Faculty
● of Business
● and Economics

Department of Informatics
Academic year: 2023/2024

DIPLOMA THESIS TOPIC

Author of thesis:	**Damián Sova**
Study programme:	Open Informatics
Scope:	Scope for Open Informatics
Topic:	**Automated neural network learning for higher accuracy human skeleton detection under realistic conditions**
Length of thesis:	2,5–4 AA

Guides to writing a thesis:

1. The aim of this thesis is to analyze the problem of image detection using neural networks and to design a neural network model for human skeleton detection in real conditions.
2. Analyze the current state of the art in human skeleton detection, i.e., methods, approaches, available datasets.
3. Design a neural network model using available approaches (BodyPoseNet – NVIDIA, MediaPipe PoseNet – Google, ViTPose) for human skeleton detection. As part of the solution to this item, create a reference dataset (image dataset for skeleton detection).
4. Implement the proposed solution using freely available technologies. Use the Apple iPhone 14 Pro, Lidar, as a possible extension in the 3D level.
5. Evaluate your own solution and formulate options for further development.

Selected bibliography:

1. ARLOW, Jim; NEUSTADT, Ila. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky.* 2nd ed. Brno: Computer Press, 2007. 567 p. ISBN 978-80-251-1503-9.
2. PATTON, Ron. *Software Testing.* Indiana: Sams Publishing, 2005. 408 p. ISBN 978-0-672-32798-8.
3. CHOLLET, François; PECINOVSKÝ, Rudolf. *Deep learning v jazyku Python: knihovny Keras, Tensorflow.* 1st ed. Praha: Grada Publishing, 2019. 328 p. Knihovna programátora. ISBN 978-80-247-3100-1.
4. CHOLLET, François. *Deep learning with Python.* Shelter Island: Manning, 2021. 478 p. ISBN 978-1-61729-686-4.
5. C. Patil and V. Gupta (2021, July 15). Human pose estimation using keypoint RCNN in pytorch. LearnOpenCV. https://learnopencv.com/human-pose-estimation-using-keypoint-rcnnin-pytorch/.
6. Rosebrock, A. (2021, April 17). R-CNN object detection with Keras, tensorflow, and Deep Learning. PyImageSearch. https://pyimagesearch.com/2020/07/13/r-cnn-object-detection-withkeras- tensorflow-and-deep-learning/.
7. Z. Tang, D. Wang and Z. Zhang, "Recurrent neural network training with dark knowledge transfer," 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Shanghai, China, 2016, pp. 5900-5904, doi: 10.1109/ICASSP.2016.7472809.

Diploma thesis topic submission date: May 2023

Deadline for submission of Diploma thesis: May 2024

L. S.

Electronic approval: 29. 6. 2023
doc. Ing. Oldřich Trenz, Ph.D.
Thesis supervisor

Electronic approval: 29. 6. 2023
Damián Sova
Author of thesis

Electronic approval: 29. 6. 2023
prof. Ing. Cyril Klimeš, CSc.
Head of Institute

Electronic approval: 29. 6. 2023
doc. Ing. František Dařena, Ph.D.
Study programme supervisor

I express my sincere gratitude to my supervisor, Doc. Ing. Oldřich Trenz, Ph.D., for his invaluable time, unwavering support, and insightful guidance throughout the entirety of this research journey. I am also deeply appreciative of the expert consultations provided by RNDr. Michal Procházka, Ph.D., from visioncraft s.r.o., whose regular insights played a pivotal role in shaping the trajectory of this work. Lastly, heartfelt thanks to my friends and family for their unwavering support, encouragement, and understanding, without which this endeavor would not have been possible.

Declaration

I hereby declare that this thesis entitled *Automated neural network learning for higher accuracy human skeleton detection under realistic conditions* was written and completed by me. I also declare that all the sources and information used to complete the thesis are included in the list of references. I agree that the thesis could be made public in accordance with Article 47b of Act No. 111/1998 Coll., Higher Education Institutions and on Amendments and Supplements to Some Other Acts (the Higher Education Act), and in accordance with the current Directive on publishing of the final thesis. I declare that the printed version of the thesis and electronic version of the thesis published in the application Final Thesis in the University Information System is identical.

I am aware that my thesis is written in accordance to Act No. 121/2000 Coll. (the Copyright Act) and therefore Mendel University in Brno has the right to conclude licence agreements on the utilization of the thesis as a school work in accordance with Article 60 (1) of the Copyright Act.

Before concluding a licence agreement on utilization of the work by another person, I will request a written statement from the university that the licence agreement is not in contradiction to legitimate interests of the university, and I will also pay a prospective fee to cover the cost incurred in creating the work to the full amount of such costs.

Brno April 11, 2024 .....

signature

Abstract

Sova, Damián. *Automated neural network learning for higher accuracy human skeleton detection under realistic conditions.* Master's Thesis. Brno : Mendel University in Brno, 2024.

Key words

image processing, pose estimation, human skeleton detection, data generation

Abstrakt

Sova, Damián. *Automatizované učenie neurónových sietí na presnejšiu detekciu ľudskej kostry v reálnych podmienkach.* Diplomová práca. Brno : Mendelova univerzita v Brně, 2024.

Kľúčové slová

spracovanie obrazu, odhadovanie pózy, detekcia ľudskej kostry, generovanie dát

Contents

1 Introduction **9**

1.1 Motivation and Basic Objectives of the Work 9
1.2 Current State and Problem to Be Addressed 10

2 Theoretical Foundations **11**

2.1 Challenges in Real-World Human Pose Estimation 11
2.2 Neural Network 13
2.2.1 How Neural Network Works 14
2.3 Convolutional Neural Network 15
2.3.1 How Convolutional Layers Work 15
2.3.2 Pooling Layers 16
2.3.3 Fully Connected Layers 16
2.3.4 Training the CNN 17
2.3.5 Example of CNN Usage 17
2.3.6 Limitations of Current Methods 17
2.4 Region-based Convolutional Neural Network 18
2.5 Existing s for Human Pose Estimation 19
2.6 PoseNet 19
2.7 MoveNet 20
2.8 MMPose 22
2.9 Metrics 24
2.9.1 Average Percentage Error 24
2.9.2 Mean Squared Error 27
2.9.2.1 Combining APE and MSE 27
2.10 Chapter Summary 28

3 Practical part **29**

3.1 Overall process introduction 29
3.2 Individual Models detection 29
3.2.1 Detection Format 33
3.3 Created Unified Format 35
3.3.1 Implementation of the Unified Format 36
3.4 Evaluation 36
3.5 Implementation Problems and Technical Limitations 38

4 Conclusion 40

References 41

List of Tables 44

List of Figures 45

List of Abbreviations 46

List of Source Codes 47

APPENDICES

1 Introduction

1.1 Motivation and Basic Objectives of the Work

The field of **computer vision** has witnessed rapid evolution, serving as the foundation for understanding visual information in images and videos (Szeliski, 2010). Within this context, the accurate detection of the **human skeleton** holds immense potential for applications ranging from autonomous systems to healthcare. The motivation driving this master's thesis is to provide an efficient method for **creating specialized datasets** for **human skeleton detection** under realistic conditions through the application of existing **neural networks** (NNs).

The evolution of **Human Pose Estimation** has revolutionized its applicability in real-world scenarios such as smart surveillance, public safety, and medical assistance. However, despite achieving impressive accuracy rates on popular datasets, the translation of these results to real-world settings remains a challenge due to the scarcity of high-quality datasets with **human pose annotations**. This scarcity arises from the costly and time-consuming nature of dataset creation, resulting in real-world applications often being trained on datasets that may not adequately represent the deployment environment (Alinezhad Noghre et al., 2022).

The disparity between training data and real-world inference data frequently leads to high-accuracy models failing to perform as expected in practical applications, particularly in scenarios involving **crowded scenes**, **heavily occluded individuals**, or subjects positioned at a significant distance from the camera. Existing datasets attempt to address specific challenges individually, but their varying skeletal structures and limited scope make them inadequate for training a unified model.

Training a NN for **human skeleton detection** is inherently challenging. It necessitates the availability of hardware capable of capturing the human body's spatial position through sensors placed on key body points, which are crucial for the detection process. Acquiring sensor data is essential for constructing a comprehensive training dataset for the NN. However, the generation of training data often occurs in controlled "laboratory conditions," using props and actors (Yang, 2018). Consequently, the creation of such a model becomes resource-intensive, requiring significant investments in time, computational resources,

human effort, and hardware. Furthermore, the model's accuracy is constrained by the level of correlation between simulated activities and real-world conditions in the detected scenario.

Existing models for **human skeleton detection** exhibit limited accuracy for specific use cases due to training in artificially created conditions (Toshev et al., 2014). The proposed approach involves leveraging existing NN models and combining their functionalities without intervention or retraining. To construct a training dataset, real-world data, such as videos capturing falls in nursing homes, can be used. Existing NN models will extract information about the skeleton from these data, which can then be utilized to train a new model with the aim of enhancing accuracy in production enviroment.

This thesis presents a cost-effective method for generating datasets specifically designed for real-world applications. The process is adaptable to various model requirements, allowing customization according to the specific needs of the final model. This means that any type of detection model can be employed, facilitating the creation of custom datasets tailored to specific use cases.

1.2 Current State and Problem to Be Addressed

At present, there is a notable gap in tools and methodologies dedicated to training models for **human skeleton detection**, utilizing pre-existing models (Yang et al., 2016). While various tools exist for model optimization, compression, and transfer learning to different models, there is a lack of knowledge regarding approaches that integrate existing NNs for specific **dataset generation**. This thesis aims to bridge this gap by exploring the combination of existing NNs to generate annotations specifically for **human skeleton detection**, addressing the current limitations in accuracy and practicality associated with conventional training methodologies.

2 Theoretical Foundations

This chapter provides an overview of the theoretical foundations of the proposed automated NNs dataset generation approach for human skeleton detection. It explains the difficulties of the human pose estimation in the real-world enviroment. Additionally, it introduces the key concepts of NNs, convolutional neural network (CNN) and region-based convolutional neural network (RCNN). Additionally, it explores existing NNs for human pose estimation, including *PoseNet*, *MoveNet*, and *MMPose*. Finally, the detection performance evaluation metrics are described.

2.1 Challenges in Real-World Human Pose Estimation

Human pose estimation faces numerous challenges in real-world applications, such as smart surveillance. Surveillance cameras are deployed in diverse locations, including shopping malls, stores, hallways, food courts, and parking lots. These locations present varying distances between individuals and cameras, occlusions, and crowded scenes. Three primary challenges were identified:

(1) **Wide Variety of Distances:** This refers to the varying scales of individuals in images, influenced by their distance from the camera and the image resolution.
(2) **Occlusions:** Individuals may be partially obscured by objects or other people in the environment.
(3) **Crowded Scenes:** Pose estimation becomes challenging in highly crowded locations, where occlusions and the presence of many individuals hinder accurate detection.

A significant obstacle in developing models to address these challenges lies in the training data. Popular datasets like MPII(TODO: REFERENCE), AI Challenger(TODO: REFERENCE), and COCO (Tsung-Yi, 2015) mainly feature unoccluded individuals close to the camera in non-crowded scenes. Although specialized datasets like CrowdPose, OCHuman, and Tiny People Pose have been introduced to tackle specific concerns, they each focus on a single issue and present challenges in their annotation styles and validation methods, making it

Figure 2.1
Keypoint annotations from COCO dataset.
Source: (Alinezhad Noghre et al., 2022)

difficult to train a comprehensive model. No single dataset adequately addresses all three main challenges of real-world human pose estimation (Alinezhad Noghre et al., 2022).

In **Figure 2.1**, it is evident that individuals who are distant from the camera or in crowded scenes are not annotated. In the upper left image, people riding elephants are not labeled, and in the bottom right image, most of the crowd is also unlabeled. Similarly, individuals distant from the camera in the other images are not annotated, even though they are visible. Hand annotating all these unmarked individuals would be challenging and time-consuming, which explains their absence. The COCO dataset's annotation files contain null keypoint annotations corresponding to individuals who may be present in the image but are not annotated. During validation, if additional skeletons lacking annotations are identified, the count of null key points is deducted. Moreover, COCO automatically disregards all but the 20 skeletons with the highest confidence to prevent undue penalization of networks for estimating skeletons of unlabeled individuals.

These limitations disproportionately affect bottom-up approaches, favored for real-world applications due to their lower computational complexities and better real-time execution capabilities. Unlike top-down approaches, bottom-up methods aim to detect individuals independently. However, the lack of labels in real-world scenarios affects both training and validation, hindering the detection of distant individuals and potentially leading to false positives without proper penalization.

In human pose estimation, there are two primary approaches: **bottom-up** and **top-down**.

(1) **Bottom-up approach:** In this approach, the algorithm first detects individual body parts, or keypoints, such as joints like elbows, knees, and wrists, in the image. These keypoints are then grouped together to form complete human poses. Bottom-up methods are often preferred for their efficiency and scalability, particularly in crowded scenes with multiple individuals.
(2) **Top-down approach:** Conversely, the top-down approach involves first detecting the entire human body in the image and then estimating the positions of individual keypoints. This method usually involves using a person detector to locate individuals and then applying a separate model to estimate their poses. While top-down approaches may offer better accuracy for isolated individuals, they can be computationally more intensive, especially in crowded environments.

These two approaches each have their advantages and limitations, and the choice between them depends on factors such as the complexity of the scene, computational resources available, and the desired balance between accuracy and efficiency.

2.2 Neural Network

We will now temporarily set aside the challenges inherent in human pose estimation and delve into the mechanics employed by existing detection models. This exploration will afford us a deeper understanding of the underlying processes driving detection methodologies.

NNs, inspired by the structure and function of the *human brain*, are computational models comprising *interconnected* layers of artificial *neurons* responsible for processing and transforming information. Demonstrating remarkable capabilities, NNs have proven effective in diverse tasks, including image recognition, natural language processing, and machine translation. A schematic representation of a simple NN is presented in **Figure 2.2**, illustrating individual layers of neurons interconnected with their neighbours. The initial layer is commonly referred to as the *input layer*, followed by *hidden layers*, and concluding with the *output layer*. In practical usage, data, such as an image in the form of a vector where values represent individual pixels, is input into the initial layer for analysis. The NN processes this information, ultimately yielding a result in the form of a single value or vector, dependent on the nature of the problem—be it

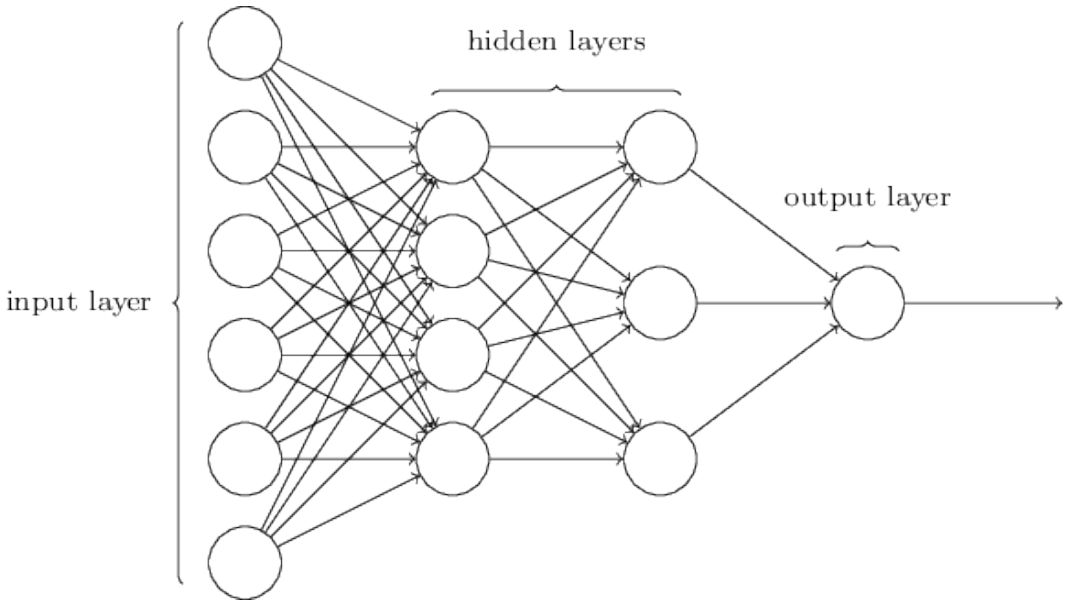

Figure 2.2
Example neural network schema. A very simple structure introduces the input layer with 6 dimensions followed by the 2 hidden layers. The first has 4 dimensions and the second with 3 dimensions. Finally, the output layer has only 1 dimension. This means, that the multidimensional input given to the NN is generalised and expressed just by one number. This is the key concept for classification models. Source: (Nielsen, 2015)

a classification or regression task. Across various fields, NNs have consistently demonstrated their robustness, excelling in tasks such as classification, prediction, filtering, optimization, pattern recognition, and function approximation (Simoneau et al., 1998).

2.2.1 How Neural Network Works

A NN inspired by the human brain, is a computational system organized into layers of artificial neurons (Nielsen, 2015). Each connection between neurons has a *weight*, representing the strength of influence (Goodfellow et al., 2016). The network learns by adjusting these weights during training, where it processes input data through layers, utilizes *activation functions* to determine neuron 'firing', and iteratively adjusts weights based on the difference between predicted and actual outcomes (Nielsen, 2015; Goodfellow et al., 2016; Mazur, 2015). The forward pass involves making predictions, while the backward pass compares predictions to actual results, adjusting weights to minimize *errors*

(Mazur, 2015). This learning process enables the neural network to recognize patterns and make accurate decisions in tasks like *image recognition* or *language processing* (Goodfellow et al., 2016).

2.3 Convolutional Neural Network

CNNs are a type of NN architecture that excels at processing and analyzing visual data, such as images and videos. They are particularly well-suited for skeleton detection due to their ability to *extract* local features from the input data. CNNs typically consist of a series of *convolutional layers*, each of which applies a *filter* or *kernel* to the input data to extract *features*. The filters are learned during the training process, allowing the CNN to learn the patterns and relationships that are important for skeleton detection (Singh, 2019). For a better understanding of the CNN architecture see example **Figure 2.3**.

CNNs have several advantages for skeleton detection (Ce et al., 2020):

- **Translation Invariance:** CNNs are invariant to small translations in the input data. This is important for skeleton detection, as the human body can be in *different positions* in an image or video.
- **Feature Learning:** CNNs can learn *complex features* from the input data, which is essential for accurate skeleton detection.
- **Parameter Sharing:** CNNs share *weights* across different positions in the input data. This reduces the number of parameters in the network, making it more efficient and easier to train.

CNNs have become the dominant architecture for skeleton detection, and they have significantly improved the accuracy of this task (Singh, 2019 Ce et al., 2020).

2.3.1 How Convolutional Layers Work

Each convolutional layer in a CNN takes an input image and applies a filter to it to extract features. The filter is a small matrix of weights that slides across the input image, producing a feature map at each position. The feature map is a representation of the input image that highlights the patterns that are relevant to the task at hand (Agarwal et al., 2019).

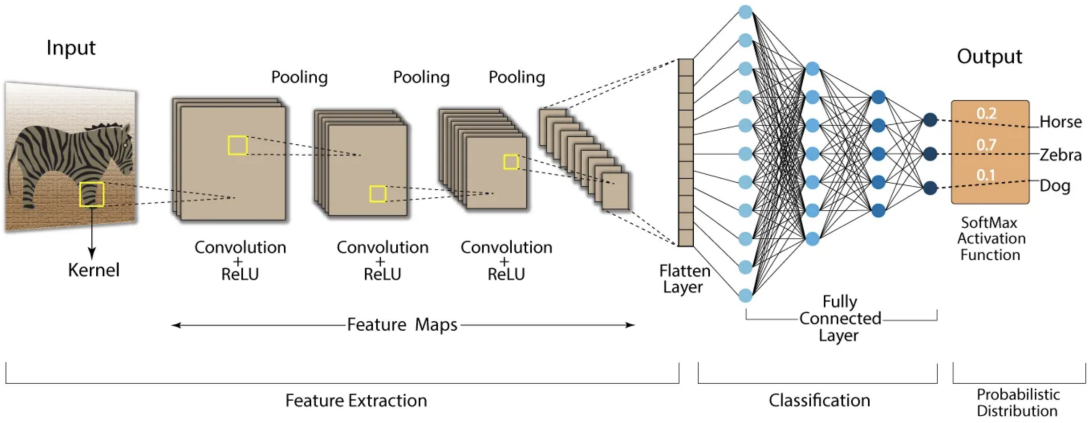

Figure 2.3
A simple classification architecture by CNN. Source: (Koushik, 2023)

For example, in the case of human skeleton detection, a filter might be used to extract features that are indicative of human joints, such as the elbows, knees, and wrists. The feature map produced by this filter would highlight the locations of these joints in the input image.

2.3.2 Pooling Layers

After the convolutional layers extract features, pooling layers are often used to reduce the dimensionality of the feature maps. This helps to reduce the computational cost of the network and also helps to make the network more invariant to small changes in the input data.

Pooling layers work by dividing the feature map into smaller regions and then taking the maximum or average value of each region. This produces a smaller feature map that still contains the most important features from the original image (Agarwal et al., 2019).

2.3.3 Fully Connected Layers

Once the feature maps have been extracted and pooled, they are passed through a series of fully connected layers. These layers are similar to the artificial neurons that are found in traditional neural networks. They take an input vector and produce an output vector.

In the case of human skeleton detection, the fully connected layers are used to classify the detected features as either human joints or backgrounds. The output vector from the final fully connected layer is a probability distribution over the possible classes (Agarwal et al., 2019).

2.3.4 Training the CNN

The CNN is trained using a process called *supervised learning* (Liu, 2012). This involves providing the network with a dataset of labelled images, where each image is labelled with the positions of the human joints. The network then learns to associate the features extracted from the images with the corresponding labels.

The training process involves adjusting the weights of the filters and connections in the network. This is done using an algorithm called backpropagation (Mazur, 2015), which iteratively updates the weights to minimize the error between the network's predictions and the ground truth labels (Agarwal et al., 2019).

2.3.5 Example of CNN Usage

To illustrate how a CNN is used for human skeleton detection, consider a scenario where a CNN is tasked with detecting human skeletons in a video stream. The CNN would first extract features from each frame of the video using its convolutional layers. Then, it would use these features to predict the positions of the human joints in the frame. This prediction can be used for various analyses of the human body movements in the video.

2.3.6 Limitations of Current Methods

While CNNs have achieved significant success in human skeleton detection, there are still some limitations to these methods. One limitation is that CNNs can be *computationally expensive*, especially when dealing with *high-resolution* images or videos. Additionally, CNNs can be sensitive to *noise* and *occlusions*, which can make it difficult to accurately detect skeletons in real-world scenarios.

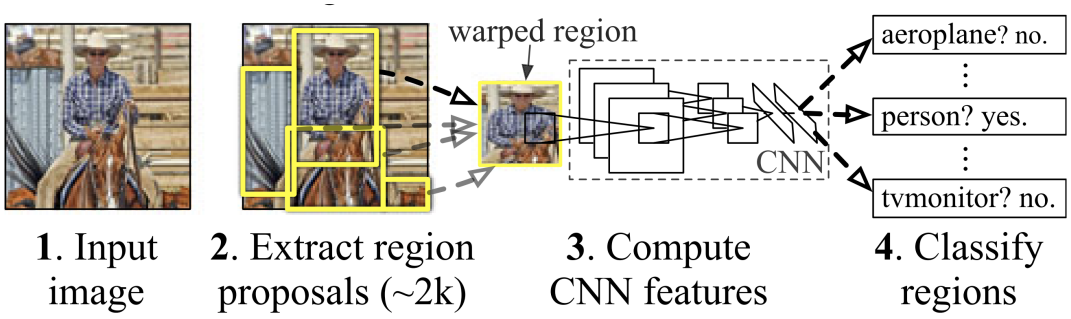

Figure 2.4
RCNN stages. Source: (Girshick, 2016)

Researchers are continuing to develop new methods to improve the accuracy and efficiency of CNNs for human skeleton detection. These methods include using deeper networks, exploring new architectures, and developing more efficient training algorithms (Agarwal et al., 2019).

2.4 Region-based Convolutional Neural Network

RCNNs are a class of deep CNNs that have been widely used for object detection and localization. They are typically characterized by a *two-stage* pipeline that involves *region proposal* and *region classification* (Ren et al., 2015). In the Figure 2.4 is displayed possible detection scenario of the RCNN.

- **Region Proposal:** The first stage of an RCNN involves generating a set of region proposals, which are candidate *bounding boxes* for objects in the input image. These proposals are typically generated using a *selective search algorithm* (He et al., 2015) that identifies regions that are likely to contain objects based on their visual saliency and spatial context (Girshick et al., 2016).
- **Feature Extraction and Classification:** The second stage of an RCNN involves classifying each region proposal as either *containing* the object or *not* (Ren et al., 2015). This is accomplished by using a CNN to extract feature vectors from each proposal and then applying a classifier to determine whether the features are indicative of the object (Girshick et al., 2016).

The original RCNN architecture has been criticized for its computational *inefficiency*, as it involves two separate stages of processing (Ren et al., 2015). To address this issue, researchers developed *Faster R-CNN*, which integrates the region proposal and region classification stages into a *single network* (Ren et al., 2015). This significantly reduces the computational cost and improves the overall performance of the system (He et al., 2015).

2.5 Existing NNs for Human Pose Estimation

Several NN architectures have been developed for skeleton detection. This thesis explores three notable examples, each with a dedicated section in this chapter:

(1) **PoseNet:** Lightweight and efficient CNN for real-time single-person detection.
(2) **MoveNet:** Family of lightweight models for real-time human pose estimation on mobile devices. Used the *lightning* version for single-person detection.
(3) **MMPose:** Library uses a CNN for multiple human pose estimation.

2.6 PoseNet

Pose_landmark (PoseNet) is a single-person detection model from the MediaPipe family that is used to detect keypoints or pose landmarks on the human body in images and videos. It is a CNN-based model that uses a *two-stage* pipeline to first detect person *bounding box* and then refine the detection by *estimating* the positions of **33** *keypoints* on detected person (Posenet, 2024). The output structure of the *PoseNet* model can be found in **Figure 2.5**.

The first stage of the pipeline, the person detection stage, uses a Single Shot MultiBox Detector (SSD) to generate a bounding box around the person in the input image. The SSD is a lightweight and efficient CNN architecture that is well-suited for real-time applications (PoseNet, 2024).

The second stage of the pipeline, the pose estimation stage, uses a CNN to refine the person detections by estimating the positions of 33 keypoints on the detected person. The keypoints are typically located on the joints of the human body, such as the elbows, knees, and wrists (PoseNet, 2024).

The PoseNet model is trained on a large COCO dataset with images and videos of people performing a variety of actions. This training data helps the model to learn to identify the keypoints on human bodies in a variety of poses and orientations. In the Table below can be found some of the key features of the PoseNet model.

Table 2.1 PoseNet model features

Feature	**Description**
Input	RGB image or video frame
Output	Pose landmarks for a person detected in the input
Landmarks	33 keypoints
Accuracy	Up to 83% accuracy on the COCO dataset
Speed	10 - 20 FPS

2.7 MoveNet

MoveNet is a family of *lightweight* and *efficient* pose estimation models developed by Google AI for *real-time* human pose estimation. In this thesis, the *lightning* version of the model was used. It is designed for mobile and embedded devices. MoveNet employs a *two-stage* pipeline to achieve real-time performance while maintaining high *accuracy* (MoveNet, 2024). The output structure of the *MoveNet* model can be found in **Figure 2.6**.

The first stage is responsible for detecting and predicting the rough location of the human body in an image or video frame. It utilizes a SSD architecture to generate *bounding box* around the potential person (MoveNet, 2024).

The second stage refines the pose estimation results by utilizing a single-person pose estimation model. This model takes the one bounding box predicted in the first stage and refines it to pinpoint the locations of **17** *keypoints* on the one detected person. The keypoints correspond to prominent joints in the human body, such as the elbows, knees, hips, and shoulders (Khanh, 2021).

The single-person pose estimation model utilizes a heatmap-based approach, where each keypoint is associated with a heatmap that indicates the probability of the keypoint being present at a particular location in the image. The model then refines the bounding box by iteratively adjusting it to maximize the overall likelihood of the keypoints being within the bounding box (Khanh, 2021).

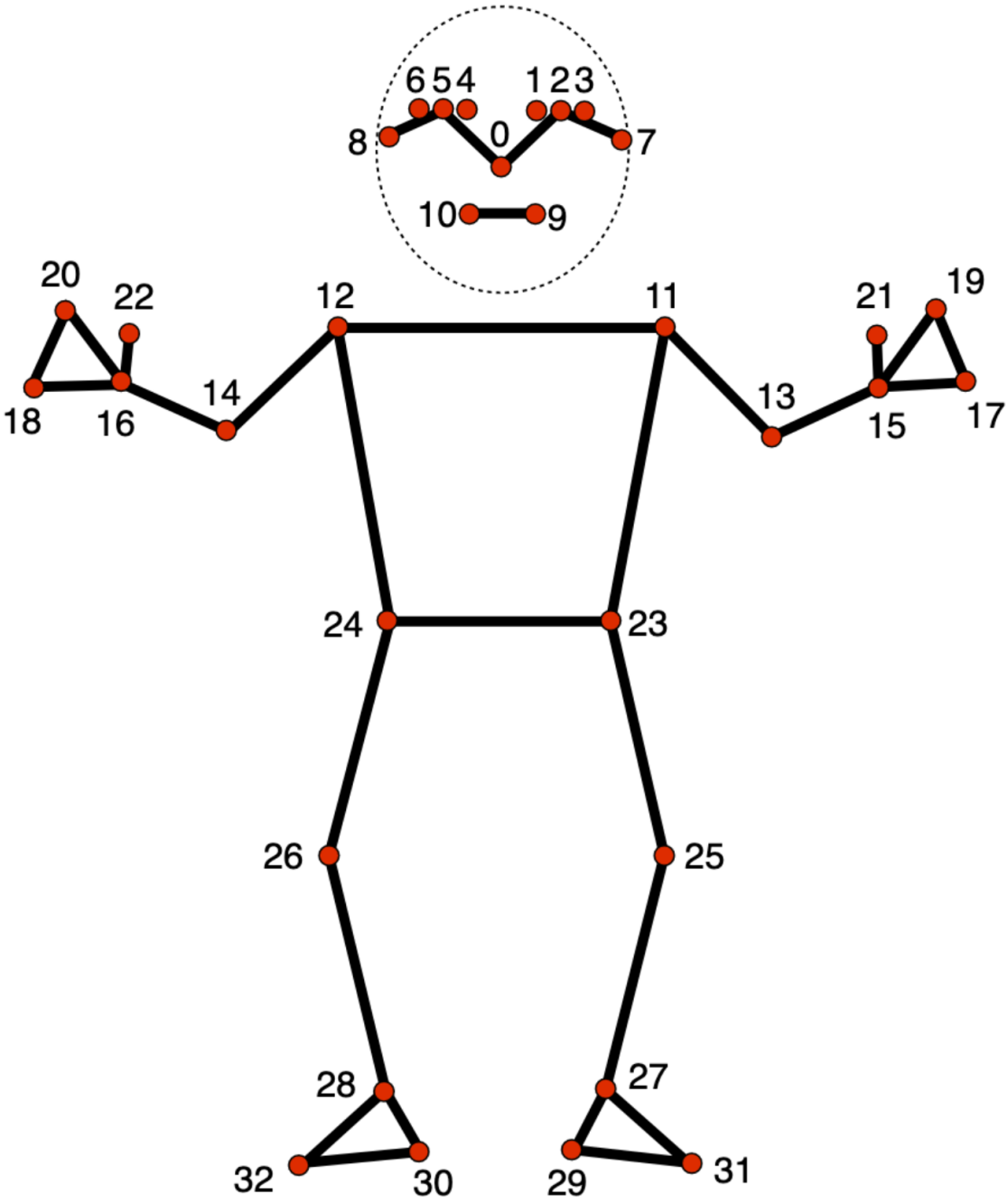

Figure 2.5
PoseNet skeleton structure with IDs to each keypoint. The skeleton representation plays a crucial role in introducing the unified format as described in **section 3.3** on **page 35**. Source: (PoseNet, 2024).

MoveNet focus on detecting the pose of the person who is closest to the image centre and ignores the other people who are in the image frame (i.e. background people rejection) (Google, 2021).

The pose refinement process is repeated multiple times to improve the accuracy of the pose estimation results. The final output is a set of 17 keypoints for the one detected person. These keypoints provide a detailed representation of the person's pose, including the positions of their joints, limbs, and other landmarks (Khanh, 2021).

Table 2.2 MoveNet model features

Feature	**Description**
Input	RGB image or video frame
Output	Pose landmarks for a person detected in the input
Landmarks	17 keypoints
Accuracy	Up to 88% on the COCO dataset
Speed	Up to 30 FPS

2.8 MMPose

This section describes the model and architecture used for multiple human pose estimation in the *MMPose* library (MMPose, 2020). The model is based on a CNN that is trained on a large dataset of images and their corresponding ground truth human poses. The network can predict the positions of **133** *keypoints* on the human body. In addition to **17** *body* keypoints, model detects **68** *face* keypoints, **21** *lefthand* keypoints, **21** *righthand* keypoints, **6** *feet* keypoints. The output structure of the *MMPose* model can be found in **Figure 2.7**.

The model is divided into *two* main stages. The first stage detects human bodies in the input image. This is done using a *Faster R-CNN* detector, which is a *two-stage* object detection network. The detector first extracts a set of *region proposals* from the image, and then *classifies* each proposal as either a *human* or *not* (Ke et al., 2019).

The second stage estimates the poses of the detected human bodies. This is done using a *top-down* pose estimation network, which is a CNN that takes as input the bounding boxes of the detected bodies and outputs a set of heatmaps that represent the probability of each keypoint being located at each pixel in the image (Ke et al., 2019).

The top-down pose estimation network is based on the *HRNet* architecture, which is a deep CNN that is designed for human pose estimation. The network consists of a series of *residual blocks*, each of which consists of two convolu-

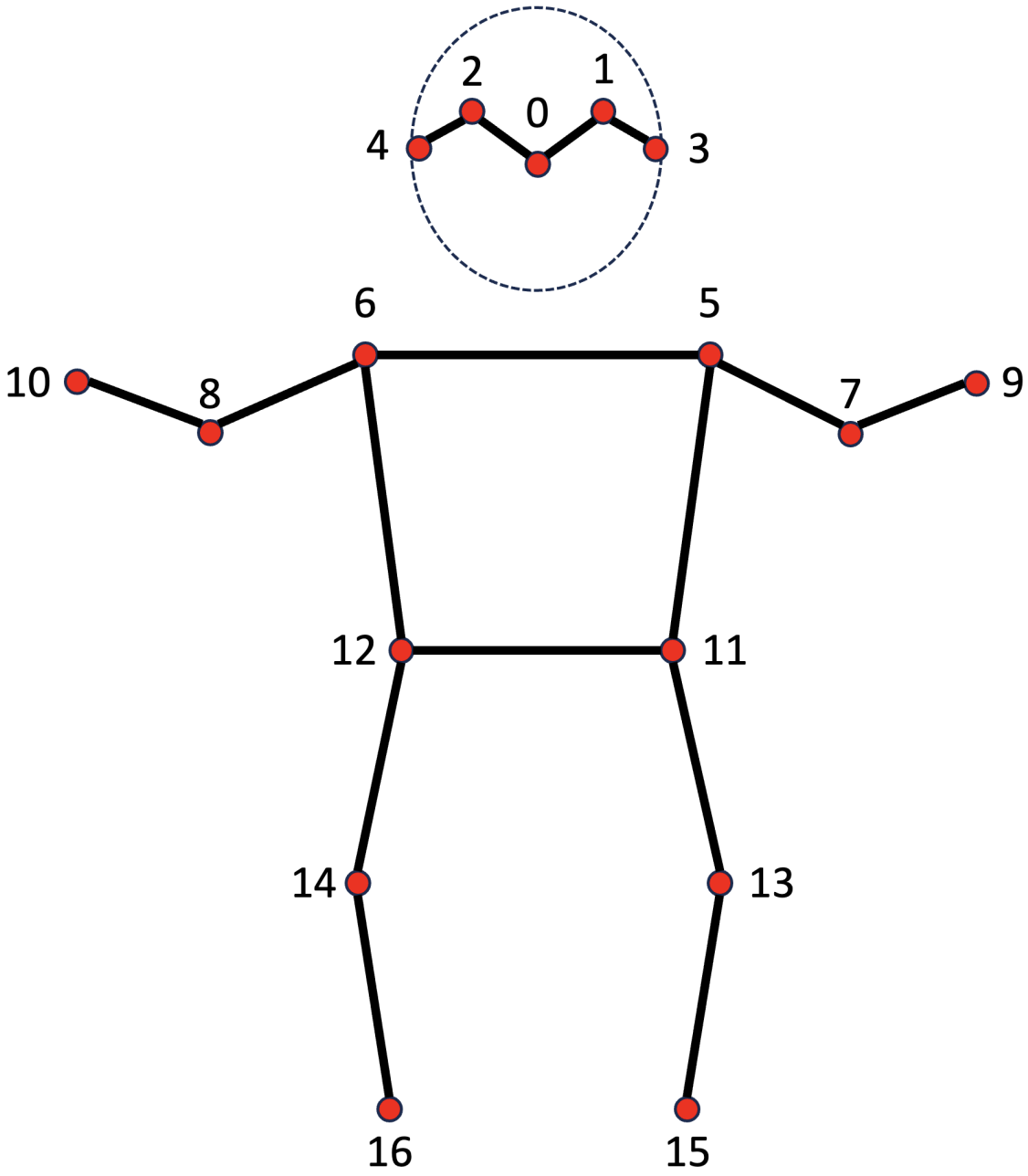

Figure 2.6
MoveNet skeleton structure with IDs to each keypoint. This model simplifies the pose detection process compared to the PoseNet described in **section 2.5** on **page 21**, which contributes to its superior performance. As a result, the MoveNet detection results do not contribute significantly to the accuracy of the unified format described in **section 3.3** on **page 35**.

tional layers with a *stride* of 1 followed by two convolutional layers with a stride of 2. This allows the network to capture both local and global information in the image (Ke et al., 2019).

The human pose estimation results are then evaluated using the COCO Whole-Body metric (Jin et al., 2020; Xe et al., 2022), which is a measure of the accuracy of the predicted keypoints.

Table 2.3 MMPose model features

Feature	**Description**
Input	RGB image or video frame
Output	List of pose landmarks for each person detected in the input
Landmarks	133 keypoints
Accuracy	76.3% on the COCO WholeBody dataset
Speed	Requires a powerful GPU for real-time use

2.9 Metrics

In this section, we take a closer look at how to effectively measure the model accuracy concerning the pose estimation. Multiple metrics, such as Average Percentage Error (APE), Mean Squared Error (MSE) and others will be investigated for the usability for pose estimation evaluation.

2.9.1 Average Percentage Error

The APE is a metric commonly used to evaluate the accuracy of human pose detection models. It measures the average difference between the predicted keypoint locations and their corresponding ground truth locations in a pose annotation(for each human instance separately, then averages all instances).

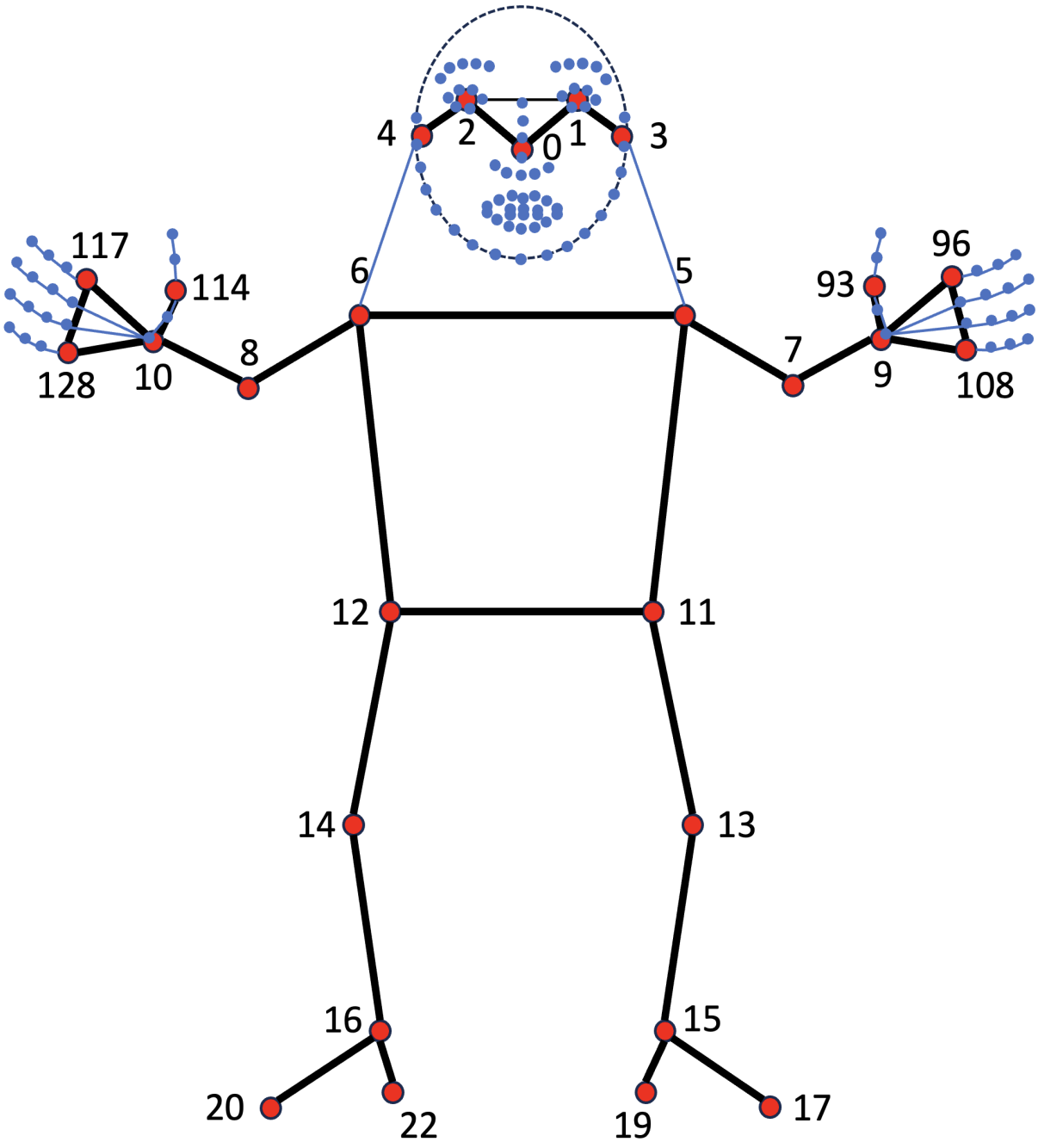

Figure 2.7
MMPose skeleton structure with IDs of used keypoint in the further processing. For simplicity, the small blue points do not have ID ensuring good visibility. Additionally, the blue keypoints have been omitted to achieve the unified format described in **section 3.3** on **page 35**.

The APE is calculated for each pose prediction in a dataset. Here's the breakdown:

(1) **Distance Calculation:** The **Euclidean** distance between each predicted keypoint and its corresponding ground truth keypoint is calculated.
(2) **Averaging:** The individual distances are then averaged across all keypoints for a single pose.
(3) **Normalization:** To account for image size variations, the average distance is normalized by the maximum dimension (width or height) of the image containing the pose. This normalization is achieved by dividing the average distance by the maximum dimension obtained from the bounding box information in the ground truth data.
(4) **Percentage Conversion:** Finally, the normalized average distance is multiplied by 100 to express the error as a percentage.

The Euclidean distance between the predictions and the annotated dataset is computed as follows:

$$D_2(x,\tilde{x}) = \frac{1}{N}\sum_{i=1}^{N} \mid p_i(x) - d_i(\tilde{x}) \mid^2 \tag{2.1}$$

where $p_i(x)$ is the 2D points annotated in the reference image of the database, $d_i(\tilde{x}) \in \mathbf{R}^2$ is the prediction in the image of the 2D coordinates of the target i knowing the predicted pose $\tilde{x}$ (Ababsa et al., 2020).

A lower APE value indicates a more accurate pose prediction. Ideally, the APE should be as close to 0% as possible. However, the acceptable APE threshold depends on the specific application and the level of precision required.

Here is the list of advantages:

- **Simple to understand:** APE provides a clear and interpretable measure of error.
- **Image size agnostic:** Normalization by image size allows for fair comparison across images of varying resolutions.

And here are some limitations of the APE metric:

- **Limited information:** APE only considers the average distance between keypoints, neglecting potential outliers or specific joint errors.
- **Normalization dependence:** The accuracy of normalization depends on the quality of bounding box information.

APE is a valuable metric for evaluating human pose detection models. However, it is recommended to use APE in conjunction with other evaluation metrics, such as **Precision-Recall** (PR) curves or **Object Keypoint Similarity** (OKS), to obtain a more comprehensive understanding of the model performance.

2.9.2 Mean Squared Error

The MSE is a metric commonly used to evaluate the accuracy of human pose detection models and is very similar to the APE metric from previous **Subsection 2.9.1**. It involves squaring the difference (**Euclidean distance**, see **Formula 2.1**) between each predicted keypoint coordinate and its corresponding ground truth value, summing these squared errors for all keypoints in a pose, and then averaging the sum.

The advantages of this metric is that it focuses on larger errors. By squaring the errors, MSE gives more weight to significant deviations between predicted and ground truth keypoints. This can be helpful in identifying poses with substantial errors in specific joints.

There are also some disadvantage behaviour in this metric, such as sensitivity to outliers. Since squaring amplifies larger errors, MSE can be overly influenced by a single incorrectly predicted keypoint, potentially inflating the overall error score.

Another fact is the interpretability for human. Unlike APE which is a percentage, MSE produces raw squared distance values that are not directly interpretable in terms of accuracy.

2.9.2.1 Combining APE and MSE

Using both APE and MSE provides a more comprehensive view of the model's performance. APE offers a general sense of average error, while MSE highlights poses with substantial keypoint localization issues.

One should consider the normalization for the MSE. While normalization by instance size is not strictly necessary for MSE it can be helpful for comparing results across datasets with varying image scales. You can normalize MSE by dividing it by the square of the maximum instance dimension (width or height of the BBOX). One have to keep in mind the interpretation. When reporting MSE it is crucial to mention that the values represent squared distances and **not percentages** for proper context.

MSE can be a valuable addition to APE for human pose detection evaluation. By combining them, you gain insights into both the average error and the presence of significant localization errors. However, the limitations of MSE needs to be kept in mind to ensure clear interpretation when reporting the results.

2.10 Chapter Summary

This chapter introduced the key concepts of NNs, CNNs, RCNNs and existing models for human pose estimation. CNNs excel in processing visual data for skeleton detection, while RCNNs, including Faster RCNN enhance efficiency through a two-stage pipeline for object detection, alongside existing NN models like PoseNet, MoveNet, and MMPose, achieving high accuracy in real-time human pose estimation. Finally, the key concepts of the detection performance evaluation were explored namely APE and the MSE metrics.

3 Practical part

This chapter comprehensively examines the various stages involved in creating a custom human pose estimation dataset. The initial phase leverages existing models outlined in **Section 2.5** (on **page 19**). Subsequently, these models are integrated into a unified tool for custom dataset creation. Additionally, the tool's performance is evaluated using two metrics, APE and OKS.

Throughout this thesis, several implementation challenges emerged, leading to certain technical limitations outlined in a dedicated **Section 3.5** on **page 38**.

3.1 Overall process introduction

In this section, the overall process will be described to provide better understanding of the thesis as a whole. The key idea behind this process will be described in the graph representing individual steps which this thesis implemented.

To better explain the process, see **Figure 3.1**. The initial phase of the process is the application of the existing skeleton detection models. Custom implemened scripts were used in this step which runs the detection using existing models as described in the next **Section 3.2**. The detection is executed on the evaluation subset of the COCO dataset.

3.2 Individual Models detection

This section introduces an approach to implementing individual model detection. It details key concepts and tools for creating complex detection scripts in the **Python** programming language.

The primary tool for implementation is Python and its extensive libraries. The following libraries were used in the initial step of creating detection scripts for individual models:

(1) PoseNet:
- ***Mediapipe*** - A library providing the PoseNet model and tools for drawing detections on images.

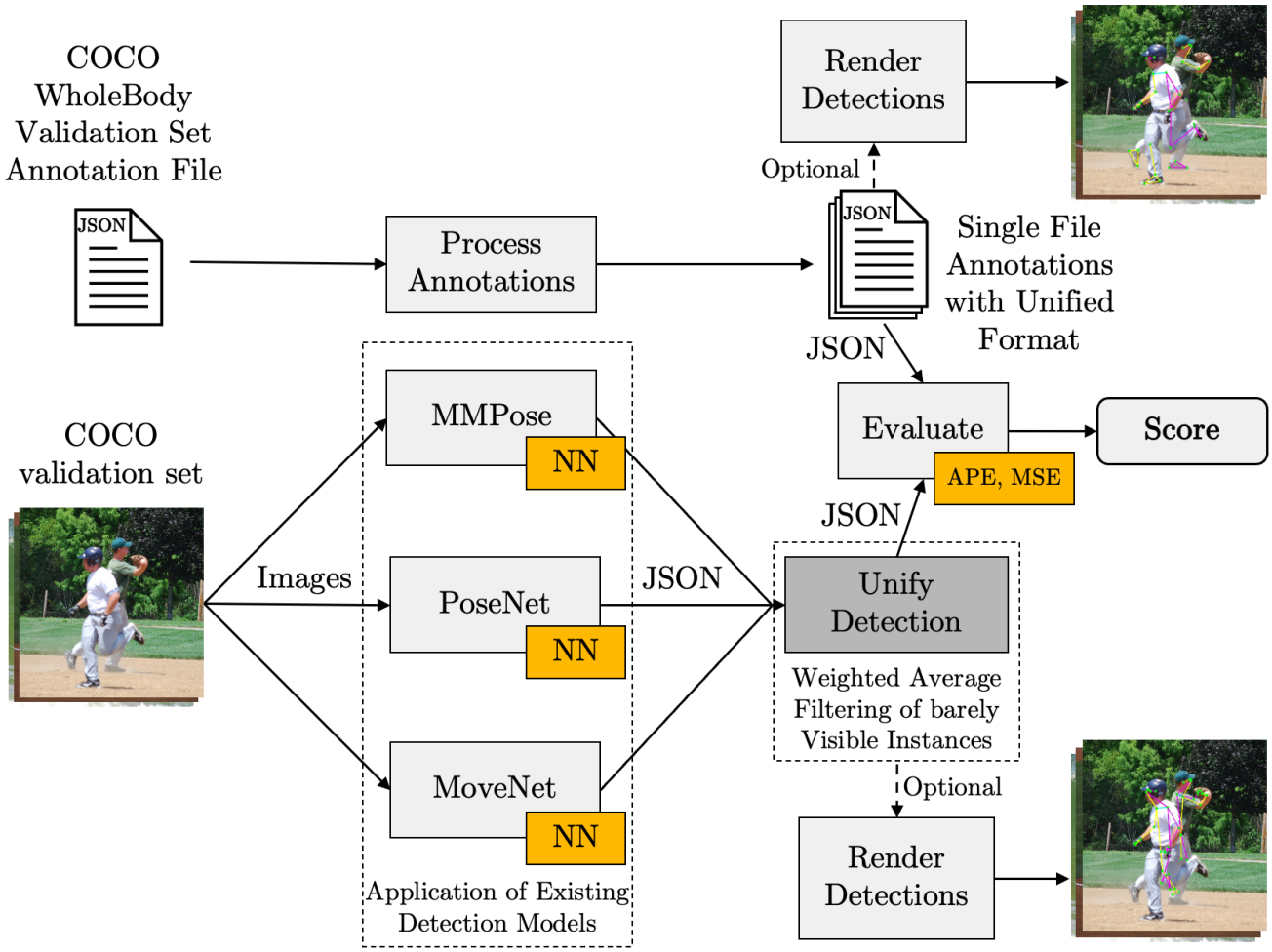

Figure 3.1
Graph of the process introducing individual stages of this thesis

(2) MoveNet:
- ***Tensorflow*** - A library used for working with image data structures.
- ***Tensorflow Hub*** - A repository of pre-trained machine learning models containing the MoveNet model.

(3) MMPose:
- ***MMCV*** - A foundational library for computer vision research.
- ***MMPOSE*** - An open-source toolbox for pose estimation based on **PyTorch**.
- ***MMDET*** - An open-source object detection toolbox based on **PyTorch**.

Several other libraries are employed in the scripts to furnish essential functionality. Here is a list of the most crucial libraries: **OS**, **Sys**, **OpenCV** and **Numpy**. Additionally, the **typing** library serves as a helper, primarily aimed at providing type hints and enhancing clarity regarding input/output structures. For this purpose, custom datatypes were devised, namely:

```
1 from typing import List, Dict, Set
2
3 DetectionInstance = List[List[float]]
4 Frame = List[DetectionInstance]
5 Detection = Dict[str, Frame]
6 SkeletonEdgesPainting = Dict[Set[int], str]
```

Source code 3.2
Custom Datatypes

These custom datatypes address the detection output format and are organized hierarchically. The first, **DetectionInstance**, represents a single human instance in the image, containing a list of keypoints, as described in **Subsection 3.2.1**. Then follows the **Frame**, which is a list of *detection instances*. At the top of this hierarchy lies the **Detection** itself, which is a dictionary of the *frames*. Lastly, the final custom datatype, **SkeletonEdgesPainting**, is a structure devised to store skeleton colour codes, assigning a single colour to each keypoint pair.

All these custom datatypes are utilized across various scripts to support type hints and enhance clarity regarding the utilized data structures. Subsequent references will be made to these types.

The main idea behind all detection scripts is to provide a very similar interface with the same functionality. Each detection model uses different resources, meaning the detection part itself is different for every case and needs to be implemented separately. Similarly, the model initialization parts are different. However, the script skeleton remains the same for every model.

The scripts utilize common custom functions gathered in the *utils* package. These methods manipulate images to provide the same frame format for every model. Additionally, they include helper functions for processing script arguments, such as handling processed file paths, output file names, and exporting detections to JSON files. Notably, the functions for drawing keypoints and skeletons on detected human instances are specific to the MoveNet model, as there is no library with this functionality for MoveNet, unlike the other two models.

See the script structure in the following code:

```
def main():
    # Process program arguments
    parse_arguments()

    # Model loading and initialisation
    model_init()

    # Frame stream processing according to given input type
    if input_type in ['webcam', 'video']:
        process_video_detection()

    elif input_type == 'image':
        process_image_detection()

    elif input_type == 'directory':
        # Process every file in the directory
        for file in directory_file_list:
            if input_type == 'video':
                process_video_detection()

            elif input_type == 'image':
                process_image_detection()
```

Source code 3.3
Simplified Detection Script Backbone (Pseudocode)

The detection scripts process various file types based on the provided program arguments. They accept both image and video files. If the input is a directory, the script iterates through all video and image files within it, executing detection on each. Refer to the single-frame detection function for a deeper understanding of the core detection implementation.

This function demonstrates the concept behind image detection execution for the MoveNet model. Similar approaches are used for MMPose and PoseNet models, with variations in the specific detection implementation and visualization. The first task is to load the frame or image. Optionally, a function for resizing the frame is available if it's too large. Then follows the pose detection itself. Program arguments manage actions like saving the detection file (JSON), the frame with drawn detections, and displaying the drawn detection frame. Optionally, correctly formatted detections are exported to a JSON file. The functions for drawing the detections onto the frame are then executed, followed by optional saving or displaying of the frame.

```python
def process_image_detection(args, model, input_name: str, input_size: int, output_file: str) -> int:
    """Handle the image detection."""

    saved_detections = 0

    # Load image
    frame = cv2.imread(input_name)

    # Resize frame if too big while keeping the aspect ration
    frame = resize_while_keep_aspect_ratio(frame, args.max_height)

    # Obtain detected keypoints
    results = detect_pose(model, frame, input_size)

    # Save prediction results
    if args.save_predictions:
        detection = formate_detection_result(results)
        detection_save_path = get_detection_save_path(args.output_root, input_name)
        save_detections_to_json(detection_save_path, detection)
        saved_detections += 1

    # Rendering
    draw_connections(frame, results, KEYPOINT_EDGE_TO_COLOR)
    draw_keypoints(frame, results)

    # Save image
    if output_file:
        cv2.imwrite(output_file, frame)

    # Display results
    if args.show:
       cv2.imshow('MoveNet Lightning', frame)
       cv2.waitKey(1)

    return saved_detections
```

Source code 3.4
Python Implementation of MoveNet Detection Function.

3.2.1 Detection Format

This subsection will take a close look into the individual detection models output format which is crucial for further processing and creation of unified format described in the **Section 3.3** on **page 35**.

The detection scripts described in the previous section produce the same detection output format for every detection model. To understand the detection JSON file format, see the following example:

```
[
    {
        keypoint_scores: [
            [
                391.4875183105469,
                178.5988311767578,
                0.2409534603357315
            ],
            ...
            [
                377.72906494140625,
                192.2097930908203,
                0.15006868541240692
            ]
        ],
        bbox: [
            385.12078857421875,
            172.7812042236328,
            400.7684631347656,
            207.59317016601562
        ]
    }
]
```

Source code 3.5
Detection Script Output Format (Example)

This format pertains to the *single-frame* detection mode. When conducting detection on a video file without the *—single–frame–output* option, the entire detection process is encapsulated within an additional dictionary to distinguish individual video frames. However, for this thesis, which focuses on dataset creation, there is no necessity to save the frame sequence. The *—single–frame–output* argument was implemented to segment the video file into individual frames and generate dedicated images for easily processed detection files.

The top level of the output format example includes a **list** of **detection instances** (objects). Each detection instance consists of the keypoints **list**. The number of keypoints varies depending on the chosen detection model. A single **keypoint** is represented by a **list** of three **float** values. The first two values represent the **coordinates** in pixels, and the last value signifies the **visibility**. Additionally, the MMPose model produces the BBOX position for the instance which is not present in the other model's output.

3.3 Created Unified Format

A highly cost-effective method of obtaining a custom dataset with real-life footage data is to leverage existing NNs to generate labels. This approach enables the training of a new model specifically tailored to the target detection task. However, for effective training, a unified format is required to aggregate the results of these individual models. This section precisely addresses the concept of a unified format introduced in this thesis, capitalizing on the strengths of existing models. As explained in the previous chapter, each model estimates a different number of keypoints, emphasizing different qualities. Refer to **Table 3.1** for a comprehensive understanding of these differences.

The rationale behind the unified format is to identify commonalities among individual formats and address their variations. Essentially, the common format is based on **MoveNet**, which comprises **17** keypoints, excluding *hands* and *feet* estimation compared to the unified format. **PoseNet** introduces additional eye keypoints (compared to unified format) that need elimination, while **MMPose** includes **107** unnecessary keypoints, particularly detailed facial keypoints and non-crucial hand keypoints (individual joints of each finger). The unified format optimally encompasses **27** keypoints, providing satisfactory detail for hands, feet, and face. Refer to **Figure 3.6** for a visual representation of the structure. You can see that the format is straightforward. It offers basic pose representation with sufficient detail to both hands and feet.

Another crucial aspect is the accurate aggregation of individual model estimations, encompassing both the coordinates of keypoints and their visibility values. To address this, we introduce a weighted average, mitigating weaknesses in faster models such as MoveNet and PoseNet. Given that these models are designed for real-time estimation, and we utilize the "lightning" model version for PoseNet, accuracy is inherently limited. Detailed values for the weighted average are available in **Table 3.1**. The assignment of the highest weight to the **MMPose** model is justified by its superior accuracy with APE metric described in the **Section 3.4**. This approach ensures the uniformity of estimations made by individual models across the entire dataset, as prepared in the previous section.

The weighted average is not applied in scenarios involving the processing of keypoints on the hands or feet. This limitation arises from the fact that the **MoveNet** model does not provide these keypoints, necessitating a basic average calculation of the two values. In other words, the weighted average is applied only when there is data from all three models.

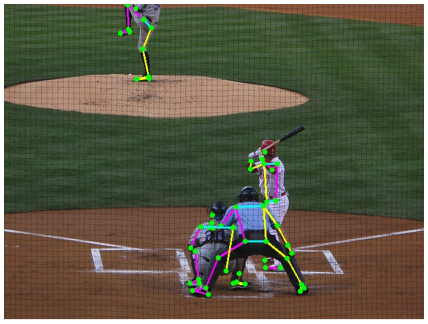

Figure 3.7 COCO WholeBody annotation simplified to unified format

Figure 3.8 MMPose model detection with 133 keypoints format

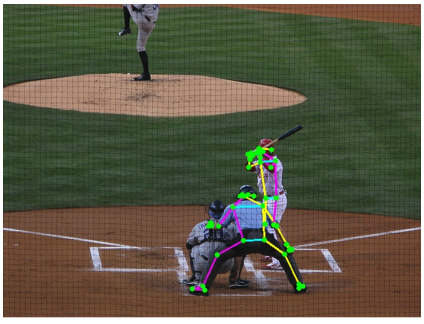

Figure 3.9 Unified format detection with 27 keypoints

Table 3.1 Comparison of the individual models detection format

Model	Keypoints	Multi-person detection	Weight
PoseNet	33	No	0.3
MoveNet	17	No	0.2
MMPose	133	Yes	0.5
Unified Format	27	Yes	–

3.3.1 Implementation of the Unified Format

This subsection aims to provide description for the unification script key concepts alongside the code examples.

3.4 Evaluation

In this section, the COCO WholeBody dataset (Xu et al. 2022) will be used as a corpus for the unified detection format evaluation. Additionally, the manipulations and details regarding the processing and use of the dataset will be investigated.

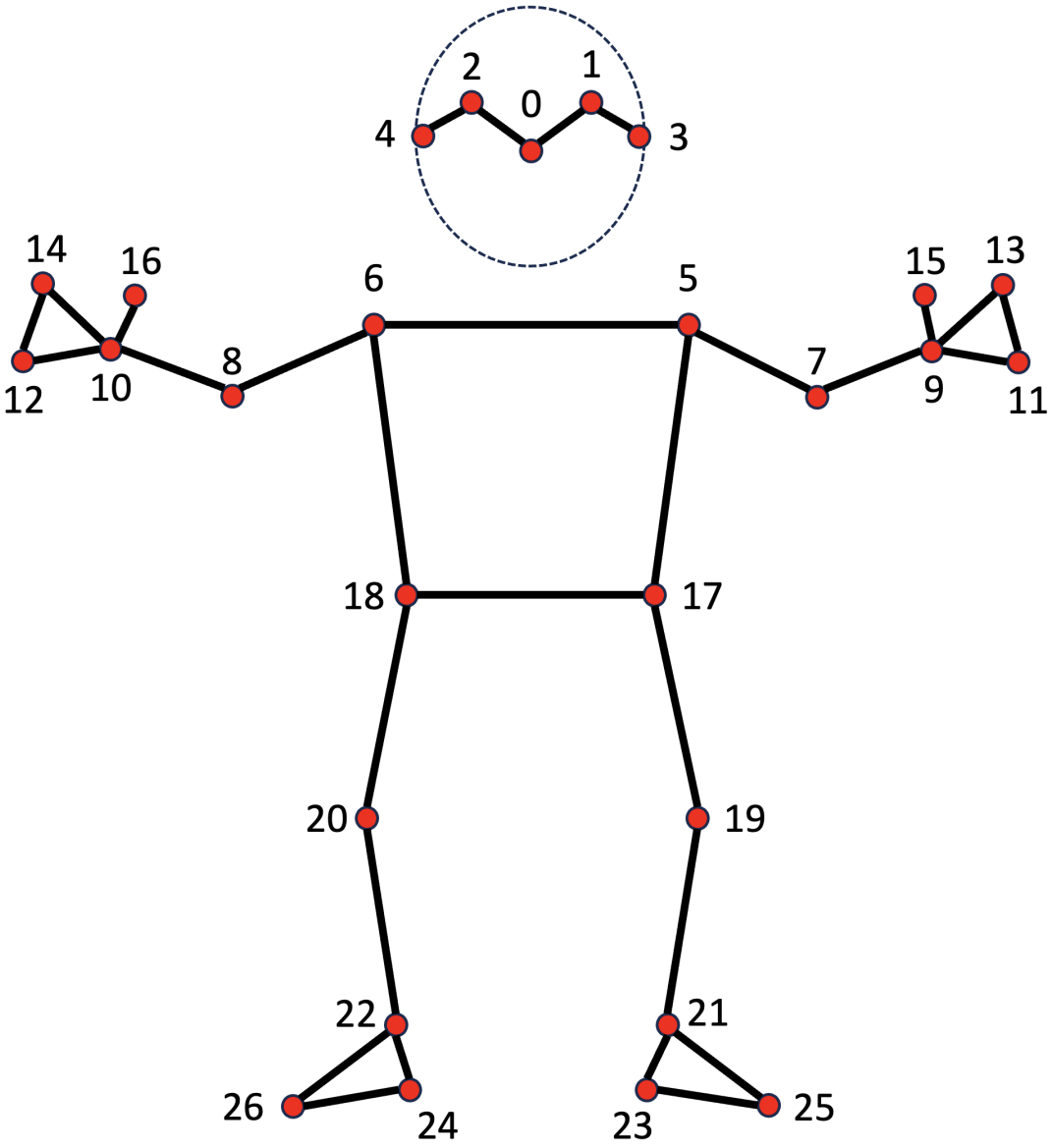

Figure 3.6
Unified format structure with IDs to each keypoint

As for the unified detection format evaluation, the **COCO WholeBody** dataset was used. Specifically the **validation 2017** subset. The total **APE** was equal to **4.37%**. This metric was calculated only on the matching instances of the predicted annotations concerning the ground truth annotations. This means that the differences in the predictions and the ground truth were not taken into account. Specifically the missing or additional (detection) instances in the predictions. But as for the corresponding instances, the evaluation serves as a great indicator of how well the individual detection instances were obtained. The criterion used for defining whether the instances from predictions correspond to

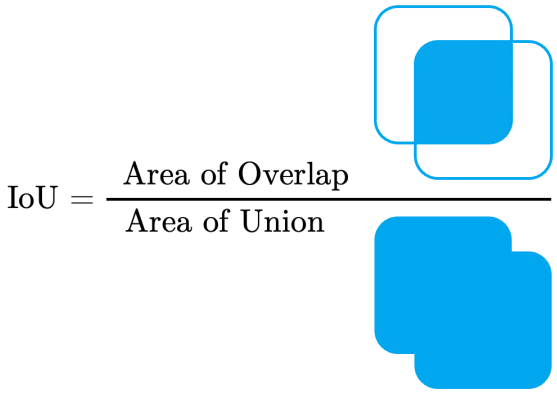

Figure 3.10
Calculation of the Intersection over Union

the ground truth annotations is the bounding box overlap. The threshold used for the overlap value, **Intersection over Union** (IoU), is the **65%**. To better understand the IoU metric see the **Figure 3.10**. The result is very satisfactory even though the averaging of the keypoints position in the unification process causes some errors.

There was also an evaluation on the APE metric just for the **MMPose** model because only this model provides us with the BBOX-es in the detection. The total APE was: **1.29%** which provided us with the information about the model precision on individual keypoints of detected instances. This information will be used for setting up the weights for weighted average in the process of unified format creation in the following **section 3.3** on **page 35**. Generally, the MMPose model is showing great performance on the detection tasks. Visually, there are only occasional differences in the prediction and the ground truth.

Because of the fact, that human pose estimation is a very complex task, it is required to use multiple metrics to express the performance of the detection models. For this reason, additionally, the mean square error(or OKS) is evaluated as well.

3.5 Implementation Problems and Technical Limitations

Throughout the course of this thesis, there have been many technical limitations and implementation struggles. This section will list some of them.

The initial plan was to utilize the **BodyPoseNet** (NVIDIA, 2024) detection model from **NVIDIA**, which supports multi-person detection and boasts high power and precision. However, after numerous unsuccessful attempts to imple-

ment BodyPoseNet detection script, a decision was made to move forward with a different model to avoid further time constraints. The primary challenge was that BodyPoseNet necessitated an NVIDIA GPU unit in conjunction with the DeepStream toolkit. Unfortunately, the development environment for this thesis consisted of macOS **Sonoma** on a **16-inch MacBook Pro with M1 Max chip and 64GB of RAM**, which lacks an NVIDIA GPU. The initial attempts to implement BodyPoseNet detection involved a virtual machine running *Ubuntu 22.04* on *Parallels Desktop*. Subsequently, this same virtual machine environment was used for all other detection models explored. Once it was discovered that detection could be performed directly on the native MacBook environment, the virtual machine was no longer required.

4 Conclusion

Evaluation of the achieved results
Suggestions for further improvements
Summary of the results of the work

References

Ababsa Fakhreddine, Hadj-Abdelkader Hicham, Boui Marouane 3D Human Pose Estimation with a Catadioptric Sensor in Unconstrained Environments Using an Annealed Particle Filter. In *Sensors* [on-line!]. Dec 2020 [cit. 2024-04-05]. Available at: http://dx.doi.org/10.3390/s20236985.

Alinezhad Noghre Ghazal, Danesh Pazho Armin, Sanchez Justin *ADG-Pose: Automated Dataset Generation for Real-World Human Pose Estimation.* Cham : Springer International Publishing, 2022. [cit. 2024-04-11]. 800 pp. ISBN 9783031092824. Available at: http://dx.doi.org/10.1007/978-3-031-09282-4_22. DOI: 10.1007/978-3-031-09282-4_22.

Agarwal Shruti, Nagrath Preeti, Saxena Anmol Human Pose Estimation Using Convolutional Neural Networks. In *2019 Amity International Conference on Artificial Intelligence (AICAI)* [on-line!]. Feb 2019 [cit. 2024-01-28]. Available at: http://dx.doi.org/10.1109/AICAI.2019.8701267.

Ce Zheng, Wenhan Wu, Chen Chen Deep Learning-Based Human Pose Estimation: A Survey. In *ACM Computing Surveys* [on-line!]. 2020 [cit. 2024-01-20]. Available at: https://doi.org/10.48550/arXiv.2012.13392.

Girshick Ross, Donahue Jeff, Darrell Trevor Region-Based Convolutional Networks for Accurate Object Detection and Segmentation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* [on-line!]. 2016 [cit. 2024-01-29]. Available at: https://ieeexplore.ieee.org/document/7112511.

Goodfellow Ian, Bengio Yoshua, Courville Aaron *Deep Learning* [on-line!]. Cambridge : MIT Press, 2016. [cit. 2024-01-22]. 800 pp. Available at: http://www.deeplearningbook.org.

Google movenet. In *Kaggle* [on-line!]. Apr 2021 [cit. 2024-01-31]. Available at: https://www.kaggle.com/models/google/movenet/frameworks/tensorFlow2/variations/singlepose-lightning/versions/4.

He Kaiming, Zhang Xiangyu, Ren Shaoqing *Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition.* New York City : Springer International Publishing, 2014. [cit. 2024-01-29]. x pp. ISBN 9783319105789. Available at: http://dx.doi.org/10.1007/978-3-319-10578-9_23. DOI: 10.1007/978-3-319-10578-9_23.

Jin Sheng, Xu Lumin, Xu Jin Whole-Body Human Pose Estimation in the Wild. In *Proceedings of the European Conference on Computer Vision (ECCV)* [on-line!]. Dec 2020 [cit. 2024-02-01]. Available at: https://arxiv.org/abs/1902.09212.

Ke Sun, Bin Xiao, Dong Liu Deep High-Resolution Representation Learning for Human Pose Estimation. In *arXiv* [on-line!]. 2019 [cit. 2024-02-01]. Available at: https://arxiv.org/abs/1902.09212.

Khanh LeViet, YuHui Chen Pose estimation and classification on edge devices with MoveNet and TensorFlow Lite. In *TensorFlow Blog* [on-line!]. Aug 2021 [cit. 2024-01-31]. Available at: https://blog.tensorflow.org/2021/08/pose-estimation-and-classification-on-edge-devices-with-MoveNet-and-TensorFlow-Lite.html.

Koushik Ahmed Understanding Convolutional Neural Networks (CNNs) in Depth. In *Medium* [on-line!]. Nov 2023 [cit. 2024-01-29]. Available at: https://medium.com/@koushikkushal95/understanding-convolutional-neural-networks-cnns-in-depth-d18e299bb438.

Liu Qiong, Wu Ying Supervised Learning. In *Encyclopedia of the Sciences of Learning* [on-line!]. Jan 2012 [cit. 2024-01-30]. Available at: http://dx.doi.org/10.1007/978-1-4419-1428-6_451.

Mazur Matt Backpropagation in Neural Networks: An Introduction. In *mattmazu* [on-line!]. march 2015 [cit. 2024-01-20]. Available at: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/.

MMPose Contributors OpenMMLab Pose Estimation Toolbox and Benchmark. In *OpenMMLab* [on-line!]. 2020 [cit. 2024-02-01]. Available at: https://github.com/open-mmlab/mmpose.

Movenet MoveNet: Efficient Human Pose Estimation in the Cloud and on Mobile.. In *TensorFlow* [on-line!]. Jan 2024 [cit. 2024-01-31]. Available at: https://www.tensorflow.org/hub/tutorials/movenet.

Nielsen Michael A. *Neural Networks and Deep Learning* [on-line!]. Online : Determination Press, 2015. [cit. 2024-01-22]. unknown pp. Available at: http://neuralnetworksanddeeplearning.com/.

NVIDIA BodyPoseNet Model Card. In *BodyPoseNet* [on-line!]. March 2024 [cit. 2024-04-02]. Available at: https://catalog.ngc.nvidia.com/orgs/nvidia/teams/tao/models/bodyposenet.

Ouyang Wanli, Chu Xiao, Wang Xiaogang Multi-source Deep Learning for Human Pose Estimation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition* [on-line!]. 2014 [cit. 2024-01-24]. Available at: http://dx.doi.org/10.1109/CVPR.2014.299.

Posenet Pose landmark detection guide. In *Mediapipe* [on-line!]. Jan 2024 [cit. 2024-01-30]. Available at: https://developers.google.com/mediapipe/solutions/vision/pose_landmarker.

Ren Shaoqing, He Kaiming, Girshick Ross Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems* [on-line!]. 2015 [cit. 2024-01-29]. Available at: https://doi.org/10.48550/arXiv.1506.01497.

Simoneau Matthew J., Price Jane Neural Networks Provide Solutions to Real-World Problems: Powerful new algorithms to explore, classify, and identify patterns in data. In *MathWorks* [on-line!]. 1998 [cit. 2024-01-22]. Available at: https://www.mathworks.com/company/newsletters/articles/neural-networks-provide-solutions-to-real-world-problems-powerful-new-algorithms-to-explore-classify-and-identify-patterns-in-data.html.

Singh Anubhav, Agarwal Shruti, Nagrath Preeti Human Pose Estimation Using Convolutional Neural Networks. In *2019 Amity International Conference on Artificial Intelligence (AICAI)* [on-line!]. New York City : IEEE, 2019 [cit. 2024-1-24]. Available at: http://dx.doi.org/10.1109/AICAI.2019.8701267.

Szeliski Richard *Computer Vision: Algorithms and Applications* [on-line!]. 1. ed. London : Springer, 2010. [cit. 2024-1-23]. 812 pp. ISBN 978-1-84882-935-0. Available at: https://szeliski.org/Book/1stEdition.htm.

Toshev Alexander, Szegedy Christian DeepPose: Human Pose Estimation via Deep Neural Networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition* [on-line!]. New York City : IEEE, June 2014 [cit. 2024-01-23]. Available at: http://dx.doi.org/10.1109/CVPR.2014.214.

Tsung-Yi Lin, Michael Maire, Serge Belongie Microsoft COCO: Common Objects in Context. In *arXiv* [on-line!]. 2015 [cit. 2024-01-30]. Available at: https://arxiv.org/abs/1405.0312.

Xu Lumin, Liu Wentao, Xu Jin ZoomNAS: Searching for Whole-body Human Pose Estimation in the Wild. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* [on-line!]. 2022 [cit. 2024-02-01]. Available at: https://github.com/jin-s13/COCO-WholeBody.

Yang Wei, Ouyang Wanli, Wang Xiaolong 3D Human Pose Estimation in the Wild by Adversarial Learning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* [on-line!]. March 2018 [cit. 2024-01-23]. Available at: https://doi.org/10.48550/arXiv.1803.09722.

List of Tables

2.1 PoseNet model features 20
2.2 MoveNet model features 22
2.3 MMPose model features 24
3.1 Comparison of the individual models detection format 36

List of Figures

2.1 Keypoint annotations from COCO dataset. Source: (scc Alinezhad Noghre et al., 2022) 12

2.2 Example neural network schema. A very simple structure introduces the input layer with 6 dimensions followed by the 2 hidden layers. The first has 4 dimensions and the second with 3 dimensions. Finally, the output layer has only 1 dimension. This means, that the multidimensional input given to the NN is generalised and expressed just by one number. This is the key concept for classification models. Source: (scc Nielsen, 2015) 14

2.3 A simple classification architecture by CNN. Source: (scc Koushik, 2023) 16

2.4 RCNN stages. Source: (scc Girshick, 2016) 18

2.5 PoseNet skeleton structure with IDs to each keypoint. The skeleton representation plays a crucial role in introducing the unified format as described in insection[section:unified-format] on atpage[section:unified-format]. Source: (scc PoseNet, 2024). 21

2.6 MoveNet skeleton structure with IDs to each keypoint. This model simplifies the pose detection process compared to the PoseNet described in insection[posenet-skeleton] on atpage[posenet-skeleton], which contributes to its superior performance. As a result, the MoveNet detection results do not contribute significantly to the accuracy of the unified format described in insection[section:unified-format] on atpage[section:unified-format]. 23

2.7 MMPose skeleton structure with IDs of used keypoint in the further processing. For simplicity, the small blue points do not have ID ensuring good visibility. Additionally, the blue keypoints have been omitted to achieve the unified format described in insection[section:unified-format] on atpage[section:unified-format]. 25

3.1 Graph of the process introducing individual stages of this thesis 30

3.7 COCO WholeBody annotation simplified to unified format 36

3.8 MMPose model detection with 133 keypoints format 36

3.9 Unified format detection with 27 keypoints 36

3.6 Unified format structure with IDs to each keypoint 37

3.10 Calculation of the Intersection over Union 38

List of Abbreviations

APE	Average Percentage Error
BBOX	Bounding Box
CNN	Convolutional neural network
IoU	Intersection over Union
MSE	Mean Squared Error
NN	Neural network
OKS	Object Keypoint Similarity
PoseNet	Pose_landmark
RCNN	Region-based convolutional neural network
SSD	Single Shot MultiBox Detector

List of Source Codes

3.2 Custom Datatypes 31
3.3 Simplified Detection Script Backbone (Pseudocode) 32
3.4 Python Implementation of MoveNet Detection Function. 33
3.5 Detection Script Output Format (Example) 34

APPENDICES