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 24, 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.

Recent advancements in computer vision have led to the widespread adoption of neural networks for human pose estimation, yet challenges persist in applying these technologies to real-world scenarios. While accuracy on benchmark datasets continues to improve, many existing datasets fail to capture the complexities encountered in practical applications, such as individuals at varying distances from the camera, crowded environments, and heavily occluded scenes. Consequently, models trained on such datasets often exhibit significant underperformance when deployed in real-world settings. This thesis introduces a novel Unified Format approach to address these challenges by standardizing the aggregation of outputs from diverse neural network models. By unifying the outputs of models like MoveNet, PoseNet, and MMPose into a cohesive structure, the Unified Format facilitates the creation of customized datasets tailored to real-world conditions. The Unified Formats performance evaluation reveals promising results, particularly in metrics such as Average Percentage Error (APE) and Mean Squared Error (MSE), which indicate close alignment between predicted and ground-truth keypoints. However, challenges arise with the Object Keypoint Similarity (OKS) metric due to stringent evaluation criteria and prediction discrepancies among models. Nonetheless, the Unified Format performs satisfactorily in accurately localizing keypoints, especially compared to individual model outputs. In summary, while this thesis does not present evidence of substantial improvement over existing benchmark performances, it successfully addresses the challenges associated with human pose estimation dataset creation by proposing and implementing a Unified Format. The format facilitates the aggregation of outputs from multiple neural network models, streamlining the dataset generation process. Despite remaining challenges, continued refinement and exploration of techniques offer opportunities to improve the Unified Format further and advance the field of human pose estimation.

Key words

pose estimation, human skeleton detection, data generation

Abstract

Sova, Damián. *Automatizované učenie neurónových sietí na presnejšiu detekciu ľudskej kostry v reálnych podmienkach.* Master's Thesis. Brno : Mendel University in Brno, 2024.

Aktuálne pokroky v počítačovom videní viedli k širokému prijatiu neurónových sietí pre odhad ľudskej pózy, no výzvy pretrvávajú pri aplikácii týchto technológií na reálne scenáre. Zatiaľ,Aktuálne pokroky v počítačovom videní viedli k širokému prijatiu neurónových sietí pre odhad ľudskej pózy, no výzvy pretrvávajú pri aplikácii týchto technológií na reálne scenáre. Zatiaľ čo sa presnosť na benchmarkových datasetoch stále zlepšuje, mnohé existujúce datasety nezachytávajú komplexity, s ktorými sa stretávajú modely v praktických aplikáciách, ako sú osoby vzdiaľené od kamery, preplnené prostredia a silne prekrývajúce sa scény. V dôsledku toho modely trénované na takýchto datasetoch často preukazujú výrazný nedostatok výkonu pri nasadení v reálnych prostrediach. Táto práca uvádza nový prístup nazývaný Unified Format, zameraný na riešenie týchto problémov prostredníctvom štandardizácie agregácie výstupov z rôznych modelov neurónových sietí. Spojením výstupov modelov ako MoveNet, PoseNet a MMPose do jednotnej štruktúry umožňuje Unified Format vytvárať prispôsobené datasety prispôsobené reálnym podmienkam. Hodnotenie výkonu Unified Formátu odhaľuje sľubné výsledky, najmä v metrikách ako je Priemerná Percentuálna Chyba (APE) a Stredná Kvadratická Chyba (MSE), ktoré naznačujú blízke zarovnanie medzi predpovedanými a skutočnými kľúčovými bodmi. Avšak vznikajú výzvy s metrikou Object Keypoint Similarity (OKS) kvôli prísnym hodnotiacim kritériám a rozdielom v predpovediach medzi modelmi. Napriek tomu Unified Formát preukazuje uspokojivý výkon pri presnom lokalizovaní kľúčových bodov, najmä pri porovnaní s jednotlivými výstupmi modelov. I keď táto práca neponúka dôkazy o významnom zlepšení oproti existujúcim benchmarkovým výkonom, úspešne rieši výzvy spojené s vytváraním datasetov pre odhad ľudskej pozície tým, že navrhuje a implementuje Unified Formát. Tento formát uľahčuje agregáciu výstupov z viacerých modelov neurónových sietí, čím zjednodušuje proces generovania datasetov. Napriek pretrvávajúcim výzvam ponúka príležitosti na ďalšie zlepšovanie Unified Formátu a posun vpred v oblasti odhadu ľudskej pózy. čo sa presnosť na benchmarkových datasetoch stále zlepšuje, mnohé existujúce datasety nezachytávajú komplexity, s ktorými sa stretávajú modely v praktických aplikáciách, ako sú osoby vzdialené od kamery, preplnené prostredia a silne prekrývajúce sa scény. V dôsledku toho modely trénované na takýchto datasetoch často preukazujú výrazný nedostatok výkonu pri nasadení v reálnych prostrediach. Táto práca uvádza nový prístup nazývaný Unified Format, zameraný na riešenie týchto problémov prostredníctvom štandardizácie agregácie výstupov z rôznych modelov neurónových sietí. Spojením výstupov modelov ako MoveNet, PoseNet a MMPose do jednotnej štruktúry umožňuje Unified Format vytvárať prispôsobené datasety prispôsobené reálnym podmienkam. Hodnotenie výkonu Unified Formátu odhaľuje sľubné výsledky, najmä

Contents

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 specialised datasets** for **human skeleton detection** under realistic conditions through the application of existing **neural networks** (NNs).

The evolution of **Human Pose Estimation** has revolutionised its applicability in real-world scenarios such as smart surveillance, public safety, and medical assistance. However, despite achieving impressive accuracy rates on popular datasets, translating these results to real-world settings remains challenging 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 (Alinezhad Noghre et al., 2022).

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 detection. Acquiring sensor data is essential for constructing a comprehensive dataset for the NN training. However, training data generation often occurs in controlled "laboratory conditions," using props and actors (Yang, 2018). Consequently, creating such a model becomes resource-intensive, requiring significant time, computational resources, human effort, and hardware investments. Furthermore, the model's accuracy is constrained by the level of correlation

between simulated activities and real-world conditions. Manual annotation is another method for creating specialised datasets. However, this process is resource-intensive and time-consuming.

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

This thesis introduces a cost-effective approach to generating datasets tailored for real-world applications. This method offers flexibility, allowing any detection model to be utilised according to the specific requirements of the dataset. For instance, a combination of models designed for crowded scenes and long-distance detection would be employed in creating a dataset for detecting individuals in shopping malls, where subjects may be distant from the camera.

1.2 Current State and Problem to Be Addressed

This thesis addresses the current limitations in accuracy and practicality associated with human skeleton detection training methodologies. A notable gap exists in the market for tools and methodologies specifically designed to train models for this task using pre-existing NNs (Yang et al., 2016). While existing tools focus on model optimisation, compression, and transfer learning, there's a lack of knowledge regarding approaches that combine these NNs for specialised dataset generation.

To bridge this gap, the following approach will be undertaken:

(1) **Analysis:** Existing training methodologies for human skeleton detection will be analysed to identify limitations in accuracy and practicality.
(2) **Proposal:** A novel approach that leverages the strengths of existing NNs to generate high-quality annotations specifically tailored for human skeleton detection datasets will be proposed.
(3) **Implementation:** The proposed approach will be developed and implemented as a tool or framework.
(4) **Testing:** The effectiveness of the proposed approach will be rigorously evaluated by calculating suitable metrics, which will describe the performance of the implemented approach to dataset generation.

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 estimating human pose in the real-world environment. Additionally, it introduces the fundamental concepts of NNs, convolutional neural network (CNN) and region-based convolutional neural network (RCNN). Moreover, it explores existing NNs for human pose estimation, including **PoseNet**, **MoveNet**, and **MMPose** detection models. 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 or healthcare. 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 (Alinezhad Noghre et al., 2022):

(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 (Andriluka et al., 2014), AI Challenger (Wu et al., 2019), and COCO (Tsung-Yi, 2015) mainly feature unoccluded individuals close to the camera in non-crowded scenes. Although specialised datasets like CrowdPose (Li et al. 2019), OCHuman (Song-Hai et al. 2019), and Tiny People Pose (Neumann and Vedaldi, 2019) have been introduced to tackle specific concerns, they each focus on a single issue and present

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

challenges in their annotation styles and validation methods, making it difficult to train a comprehensive model. No dataset adequately addresses all three 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 labelled; in the bottom right image, most of the crowd needs to be labelled. 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, the count of null key points is deducted if additional skeletons lacking annotations are identified. Moreover, COCO automatically disregards all but the 20 skeletons with the highest confidence to prevent undue penalisation of networks for estimating skeletons of unlabeled individuals.

The limitations outlined above disproportionately impact bottom-up approaches, often preferred in real-world applications due to their lower computational complexities and superior real-time execution capabilities. Unlike top-down methodologies, which first detect entire human bodies before estimating individual keypoints, bottom-up methods focus on detecting individuals independently. However, the absence of labelled data in real-world scenarios poses significant challenges for training and validation processes within bottom-up approaches (Alinezhad Noghre et al., 2022).

The scarcity of labelled data in real-world environments hampers the effectiveness of bottom-up approaches during the training phase. The model's ability to accurately detect keypoints for such scenarios is compromised without sufficient labelled data representing distant individuals or individuals in crowded scenes. Consequently, the model may fail to generalise well to real-world conditions, leading to suboptimal performance in deployment scenarios (Alinezhad Noghre et al., 2022).

Similarly, the lack of labelled data poses challenges when validating bottom-up approaches. The model may struggle to detect keypoints in scenarios where distant individuals or crowded scenes are prevalent. However, due to the absence of ground truth annotations for these challenging instances, false positives generated by the model may go unnoticed during validation. This lack of proper penalisation for false positives can result in inflated validation accuracy metrics, masking the model's performance shortcomings in real-world scenarios (Alinezhad Noghre et al., 2022).

The limitations imposed by the absence of labelled data disproportionately affect bottom-up approaches in human pose estimation. Despite their advantages in computational efficiency and real-time execution, bottom-up methods face significant hurdles in accurately detecting keypoints for distant individuals and crowded scenes in real-world environments. Addressing these challenges requires innovative approaches to dataset collection, annotation, and model training/validation strategies, ensuring the robust performance of bottom-up approaches in diverse real-world scenarios (Alinezhad Noghre et al., 2022).

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 concludes 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

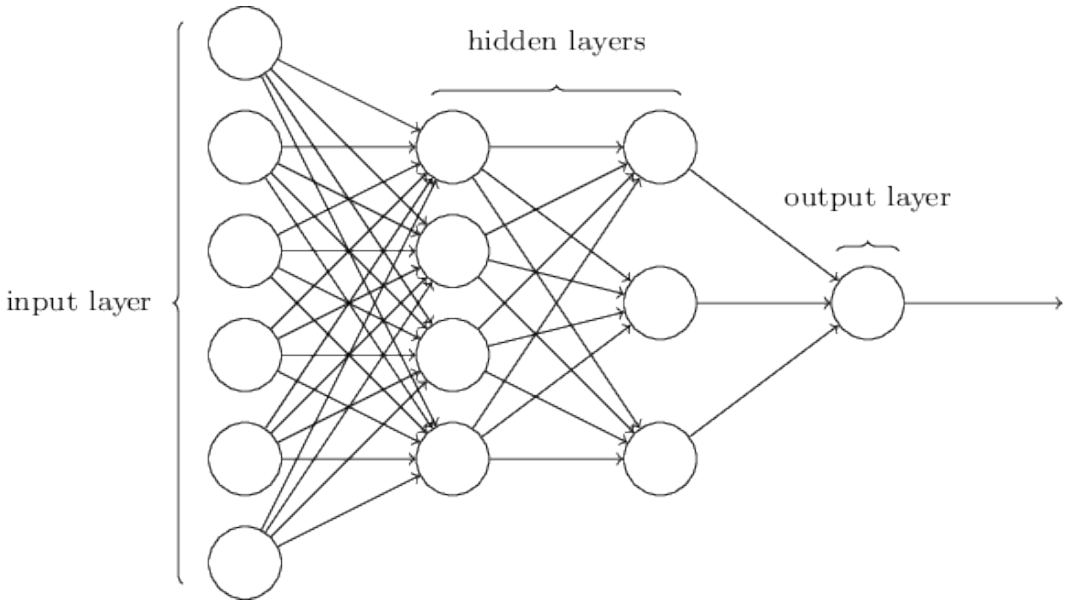

Figure 2.2
Example neural network schema. A very simple structure introduces the input layer with six dimensions followed by the two hidden layers. The first layer has four dimensions, and the second has three dimensions. Finally, the output layer has only one dimension. This means that the multidimensional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

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 a classification or regression task. Across various fields, NNs have consistently demonstrated their robustness, excelling in classification, prediction, filtering, optimisation, 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 organised 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, utilises **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 minimise **errors**

(Mazur, 2015). This learning process enables the neural network to recognise 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 analysing 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 training, allowing the CNN to learn the important patterns and relationships 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 feature is essential 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, 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 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 represents the input image highlighting the patterns relevant to the task (Agarwal et al., 2019).

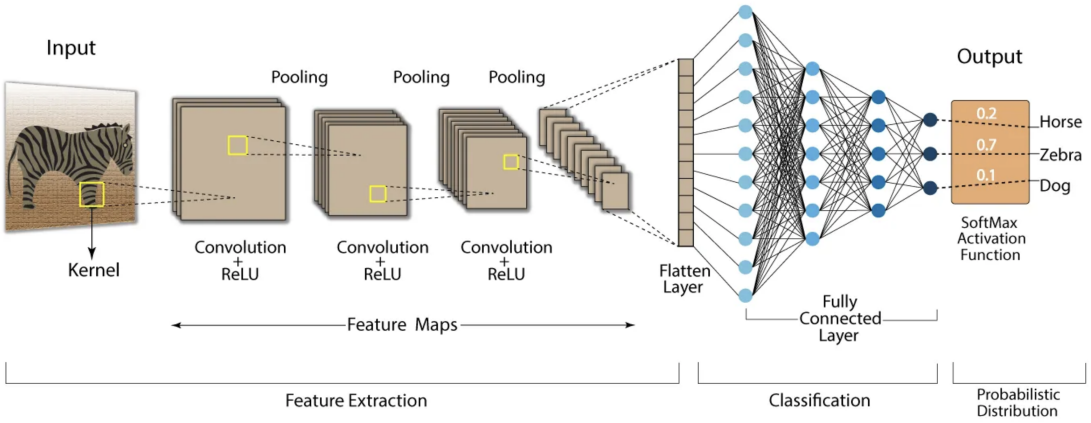

Figure 2.3
A simple classification architecture by CNN. Source: (Koushik, 2023)

For example, in the case of human skeleton detection, a filter might be used to extract features 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 reduce the network's computational cost and makes it more invariant to small changes in the input data.

Pooling layers divide the feature map into smaller regions, taking each region's maximum or average value. This produces a smaller feature map that still contains the most critical 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 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 **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 a backpropagation algorithm (Mazur, 2015), which iteratively updates the weights to minimise errors 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 video frame 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, these methods still have limitations. 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**, making it challenging to accurately detect skeletons in real-world scenarios.

Researchers continue developing 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).

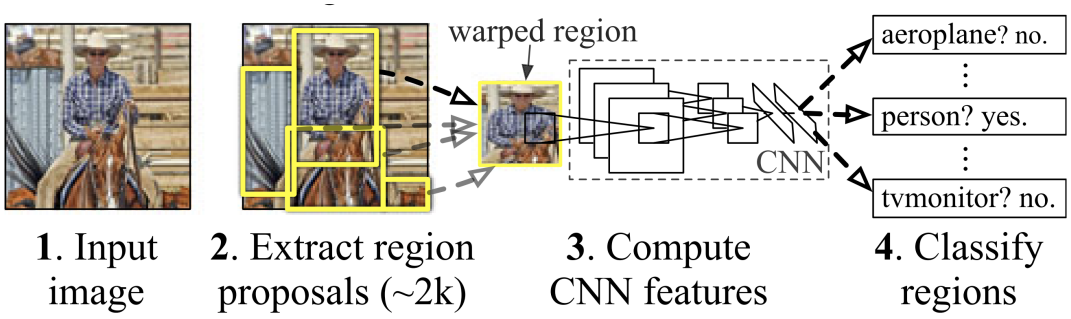

Figure 2.4
RCNN stages. Source: (Girshick, 2016)

2.4 Region-based Convolutional Neural Network

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

- **Region Proposal:** The first stage of an RCNN involves generating a set of region proposals, which are candidate **bounding boxes** (BBOX) 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 using a CNN to extract feature vectors from each proposal and then applying a classifier to determine whether the features indicate the object (Girshick et al., 2016).

The original RCNN architecture has been criticised for its computational **inefficiency**, involving two separate processing stages (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 system's overall performance (He et al., 2015).

2.5 Existing NNs for Human Pose Estimation

Several NN architectures have been developed for skeleton detection. The following models are used based on the low resource requirements. The key criterion was that the models could be executed on the CPU unit and do not require GPU. Another limitation was the model complexity so that the detection could be used in real time. Since the COCO evaluation set comprises **5K** images and the training subset comprises **118K** images, which are utilised for model detection purposes, the models must be capable of executing the detection process within a reasonable timeframe, considering the resource constraints imposed by the available hardware resources for this thesis. 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.

A comprehensive examination of each model and its unique characteristics will be undertaken. Within each section, essential insights into the models' qualities, details regarding their output formats, and a summarisation table will be provided to aid in understanding their performance. This comprehensive overview aims to clarify and facilitate a detailed evaluation of each model's capabilities within the study context.

2.6 PoseNet

Pose_landmark (PoseNet) is a single-person detection model from the MediaPipe family 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 first to detect person **BBOX** 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 BBOX 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 various actions. This training data helps the model identify the keypoints on human bodies in various poses and orientations. In the Table below are some of the critical 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 utilises a SSD architecture to generate **BBOX** around the potential person (MoveNet, 2024).

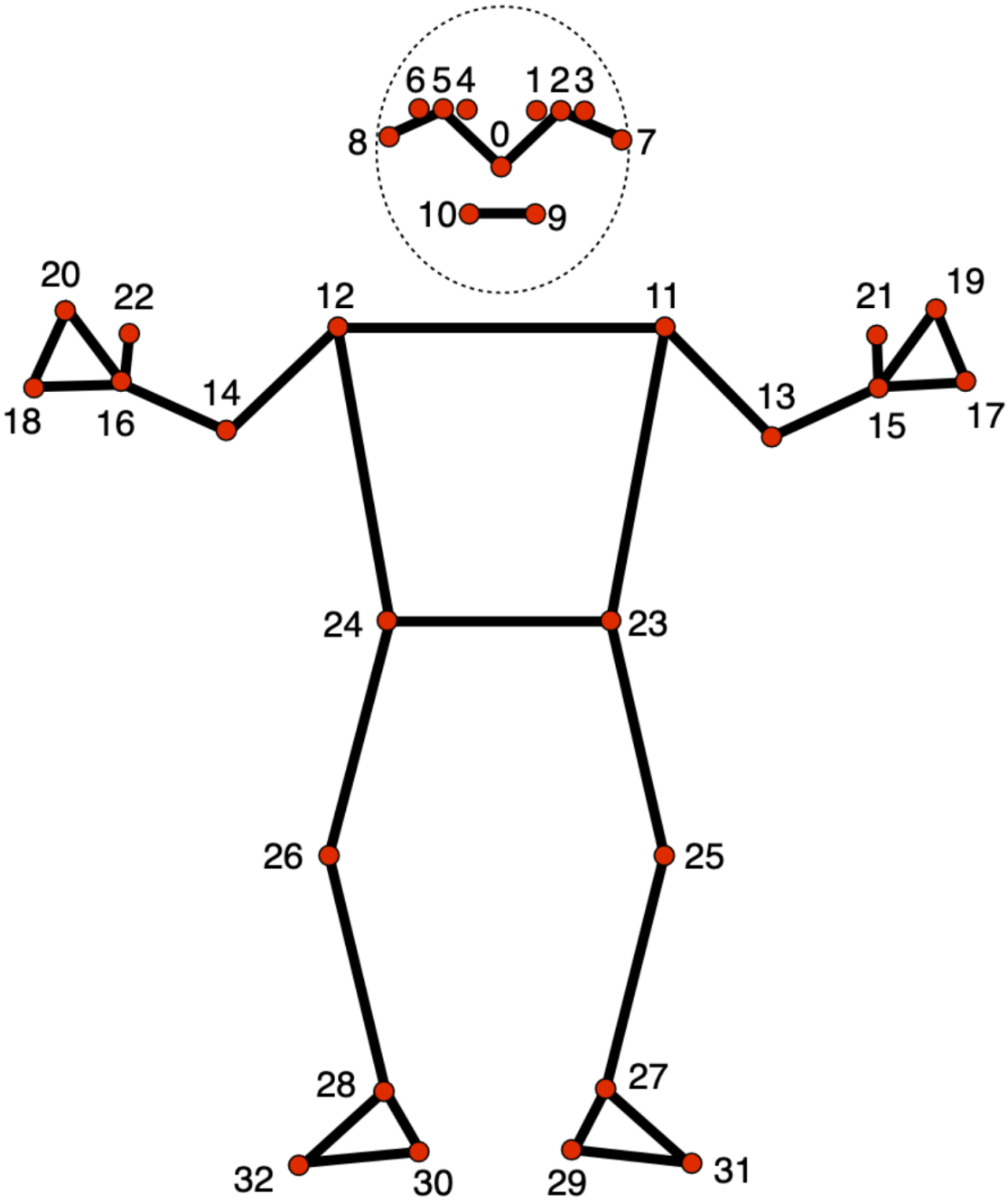

Figure 2.5
PoseNet skeleton structure with IDs to each keypoint. The skeleton representation plays a crucial role in introducing the Unified Format as described in **section 3.4** on **page 43**. Source: (PoseNet, 2024).

The second stage refines the pose estimation results using a single-person pose estimation model. This model takes the one BBOX 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 utilises 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 BBOX by iteratively adjusting it to maximise the likelihood of the keypoints within the BBOX (Khanh, 2021).

MoveNet focuses on detecting the pose of the person closest to the image centre and ignores the other people 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**.

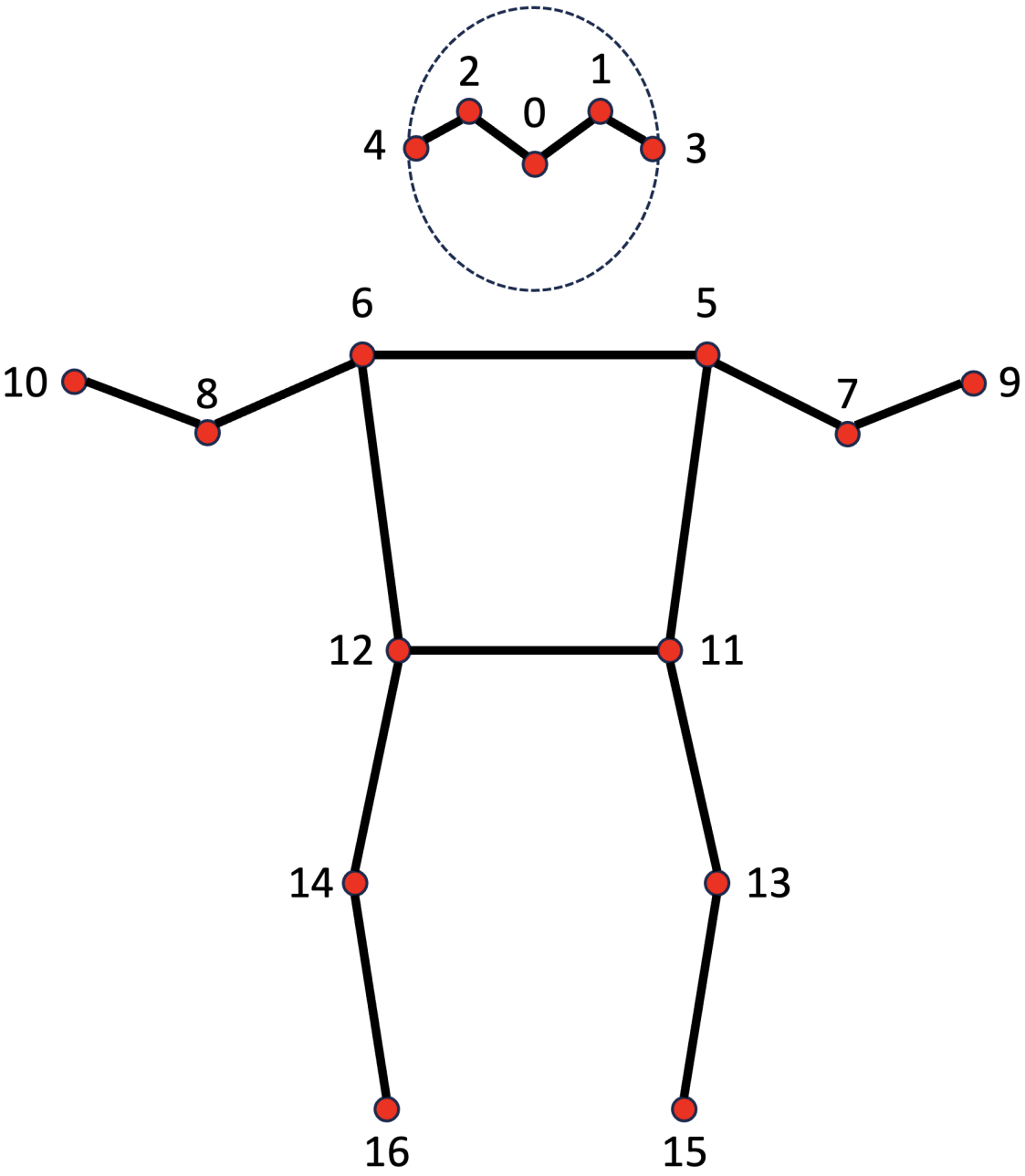

Figure 2.6
MoveNet skeleton structure with IDs to each keypoint. This model simplifies the pose detection process compared to the PoseNet described in **section 2.5** on **page 23**, which contributes to its superior performance. As a result, the MoveNet detection results do not contribute significantly to the accuracy of the Unified Format described in **section 3.4** on **page 43**.

The model is divided into **two** main stages. The first stage detects human bodies in the input image. This uses a **Faster R-CNN** detector, 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 BBOXes 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, a deep CNN designed for human pose estimation. The network consists of a series of **residual blocks**, each consisting of two convolutional layers with a **stride** of 1 followed by two convolutional layers with a stride of 2. This allows the network to capture the image's local and global information (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 measures the accuracy of the predicted keypoints. The COCO WholeBody dataset pose annotations have the same format as the output of the MMPose model.

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 Human Pose Models for Unified Format

In this segment, an analysis of three distinct human pose estimation models—**PoseNet**, **MoveNet**, and **MMPose**—is conducted, all utilised within the context of this thesis to establish a Unified Format for the representation of

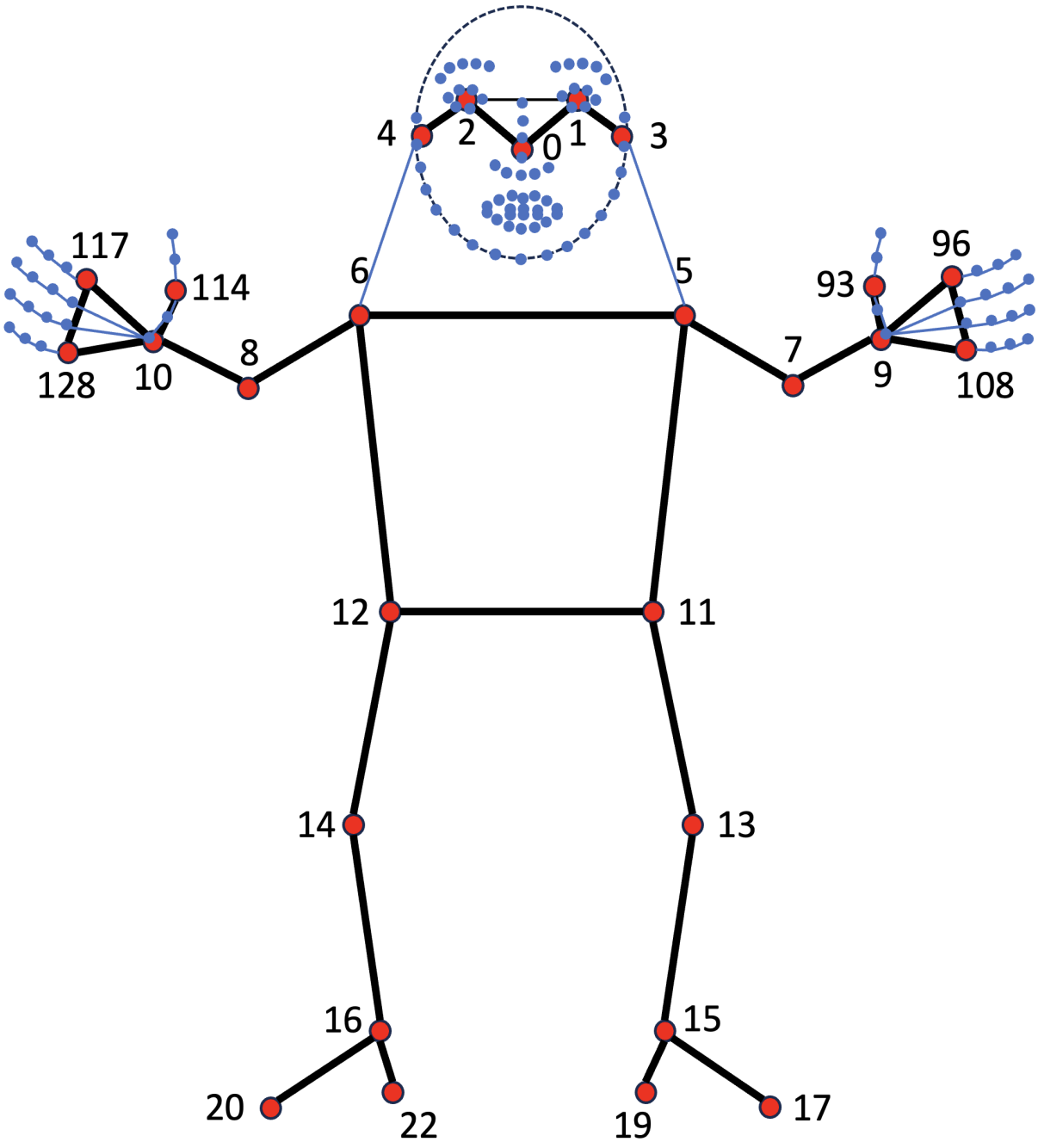

Figure 2.7
MMPose skeleton structure with IDs of used keypoint in the further processing. For simplicity, the minor blue points do not have IDs, ensuring good visibility. Additionally, the blue keypoints have been omitted to achieve the Unified Format described in **section 3.4** on **page 43**.

human pose. The merits and demerits of each model will be scrutinised, and their contributions towards the development of a resilient unified pose format will be examined.

Commencing with **PoseNet**, efficiency in single-person pose estimation is demonstrated, achieving commendable **accuracy** of up to **83%** on the COCO dataset. Its **lightweight CNN architecture** renders it adaptable to diverse applications, encompassing **33 keypoints** that capture crucial body joints. However, PoseNet's limitation to single-person detection results in neglecting scenarios involving multiple individuals, and the model may encounter challenges with intricate poses or occlusions.

MoveNet, conversely, is distinguished by its exceptional **lightweight design**, making it ideal for real-time applications, operating at speeds of up to **30 FPS**. It attains high accuracy levels, reaching up to **88%** on the COCO dataset, specifically for single-person poses. Emphasising the detection of the individual closest to the centre of the image, MoveNet reduces processing requirements. Nonetheless, like PoseNet, MoveNet is confined to single-person detection and offers a more concise set of **17 keypoints**, potentially lacking in detail for complex poses.

Transitioning to **MMPose**, the model facilitates **multi-person pose estimation**, crucial for scenarios featuring multiple subjects. Providing the most comprehensive **keypoint array**, consisting of **133 keypoints** encompassing facial features, hand gestures, and foot positions, MMPose adopts a two-stage methodology involving Faster R-CNN for detection and HRNet for top-down pose estimation. However, MMPose's demand for a powerful GPU restricts real-time application on resource-constrained devices, and it achieves a lower accuracy rate (**76.3%** on COCO WholeBody) compared to PoseNet and MoveNet for single-person tasks.

The rationale for a Unified Format is to combine insights from different models to create a more robust and versatile pose representation. Augmenting detection capabilities and enhancing keypoint data, the Unified Format ensures adaptability across diverse scenarios. Strategies to mitigate redundancy, such as prioritisation and data fusion techniques, further contribute to its effectiveness within the framework of this thesis. Reference to the comparison **Table 2.4** aids in better understanding the performance metrics of each model.

Table 2.4 Individual Models Comparison

Model	Strengths	Weaknesses
PoseNet	Efficient single-person pose estimation with good accuracy. Lightweight and suitable for various applications. Provides 33 essential body joint keypoints.	Limited to single-person detection. May struggle with complex poses or occlusions.
MoveNet	Extremely lightweight and efficient for real-time applications. High accuracy for single-person poses. Focuses on the person closest to the image centre.	Limited to single-person detection. Provides fewer keypoints, potentially lacking detail for complex poses.
MMPose	Enables multi-person pose estimation. Provides a comprehensive set of keypoints including face, hands, and feet. Utilises a two-stage approach.	Requires a powerful GPU for real-time use. Lower accuracy compared to PoseNet and MoveNet for single-person tasks.

2.10 Metrics

In this section, we look at how to measure the model accuracy concerning the pose estimation effectively. Multiple metrics, such as **OKS**, **APE** and **MSE** will be investigated for the usability for pose estimation evaluation.

2.10.1 Object Keypoint Similarity

The **OKS** (COCO, 2024) is a metric used and introduced by the COCO Consortium for the accuracy evaluation of human pose estimation. It comprehensively assesses pose estimation performance by considering the similarity between predicted and ground truth keypoint locations.

The OKS metric calculates the similarity between predicted keypoints and their corresponding ground truth keypoints. It takes into account the spatial distance between keypoints, as well as the scale of the person in the image. The formula for calculating OKS is as follows:

$$OKS = \frac{\sum_i \exp(-\frac{d_i^2}{2s^2k_i^2})\delta(i)}{\sum_i \delta(i)} \tag{2.1}$$

where d_i represents the Euclidean distance between the predicted keypoint i and its corresponding ground truth keypoint, s denotes the scale of the person in the image, k_i is a per-keypoint constant that controls the falloff, and $\delta(i)$ is an indicator function that equals one if keypoint i is visible in both the prediction and ground truth and 0 otherwise (COCO, 2024).

The OKS metric ranges from 0 to 1, where a value closer to 1 indicates a higher similarity between predicted and ground truth keypoints.

One advantage of OKS is its ability to account for scale variation and per-keypoint constants, providing a more robust evaluation of pose estimation accuracy. Additionally, OKS is suitable for comparing pose estimations across different datasets and scenarios.

However, OKS also has some limitations. It may not adequately capture errors in specific keypoints or joint configurations and relies on accurate per-keypoint constants and scale information for accurate evaluation.

In summary, Object Keypoint Similarity (OKS) is a valuable metric for assessing the accuracy of human pose estimation models. Its consideration of spatial distance, scale, and per-keypoint constants makes it a comprehensive evaluation tool. However, it should be used in conjunction with other metrics to provide a thorough assessment of model performance.

2.10.2 Average Percentage Error

The **APE** (also known as **MAPE** (De Myttenaere et al., 2016)) is a human-readable metric used to evaluate the accuracy of human pose estimation 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).

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

(1) **Distance Calculation:** The **Euclidean distance** in pixels between each predicted keypoint and its corresponding ground truth keypoint is calculated.
(2) **Averaging:** The individual distances are averaged across all single pose's keypoints.
(3) **Normalisation :** For instance, in size variations, the average distance is normalised by the maximum dimension (width or height) of the BBOX containing the instance pose. This normalisation is achieved by dividing the average distance by the maximum dimension obtained from the BBOX information in the ground truth data.
(4) **Percentage Conversion:** Finally, the normalised average distance is multiplied by 100 to express the error as a percentage.

The Euclidean distance between the prediction keypoint x and the annotated keypoint $\tilde{x}$ is computed as follows:

$$D(x,\tilde{x}) = \sqrt{\sum_{i=1}^{N}(a_i(x) - p_i(\tilde{x}))^2} \tag{2.2}$$

where $a_i(x)$ is the i-th dimension coordinate of the annotated point in the reference image of the dataset, $p_i(\tilde{x}) \in \mathbf{R}^2$ is the prediction in the image of the i-th dimension coordinate of the target i knowing the predicted pose $\tilde{x}$. N is the number of dimensions of the point. For 2D detection, the number is 2.

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.

The APE formula is as follows:

$$APE = \frac{1}{N}\sum_{i=1}^{N} D(x,\tilde{x})\frac{100}{max(w_{bbox}, h_{bbox})} \tag{2.3}$$

where x is the 2D annotated point in the reference image of the dataset, $\tilde{x} \in \mathbf{R}^2$ is the 2D prediction point in the image. N is the number of keypoints of the detection instance. The w_{bbox} and h_{bbox} are the dimensions of the detection instance BBOX.

Here is the list of advantages:

- **Simple to understand:** APE provides a clear and interpretable measure of error.
- **Image size agnostic:** Normalisation 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 BBOX 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.

2.10.3 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 2.10.2**. It involves squaring the difference (**Euclidean distance**, see **Formula 2.4**) between each predicted keypoint and its corresponding ground truth keypoint, summing these squared errors for all keypoints in a pose, and then averaging the sum.

MSE between the prediction pose x and the annotated pose $\tilde{x}$ is computed as follows:

$$MSE(x,\tilde{x}) = \frac{1}{N}\sum_{i=1}^{N}(a_i(x) - p_i(\tilde{x}))^2 \tag{2.4}$$

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

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

This metric also has some disadvantageous behaviours, such as sensitivity to outliers. Since squaring amplifies more significant errors, MSE can be overly influenced by a single incorrectly predicted keypoint, potentially inflating the overall error score.

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

2.10.4 Combining OKS, APE and MSE

When evaluating human pose estimation models, combining multiple evaluation metrics such as **OKS**, **APE** and **MSE** offers a more **comprehensive** understanding of the model's **performance**.

OKS takes into account the similarity between predicted and ground truth keypoints, considering **spatial distance**, **scale**, and **per-keypoint constants**. APE provides a **general sense** of **average error** across all keypoints, allowing for a **straightforward** assessment of **overall accuracy**. On the other hand, MSE **highlights** poses with substantial keypoint **localisation issues** by measuring the squared distances between predicted and ground truth keypoints. It is crucial to note that when reporting MSE the values represent squared distances and not percentages for proper context.

Combining OKS APE and MSE allows for a more nuanced evaluation of human pose estimation models. By considering average error, keypoint localisation issues, and similarity to ground truth keypoints, researchers gain valuable insights into model **performance across different aspects of pose estimation**.

However, it is essential to be aware of the limitations of each metric. While APE and MSE provide insights into overall accuracy and localisation errors, they may not capture errors in specific keypoints or joint configurations. Similarly, OKS relies on accurate per-keypoint constants and scale information for accurate evaluation and may not adequately capture all types of pose estimation errors.

In conclusion, combining OKS APE and MSE offers a more comprehensive evaluation of human pose estimation models, allowing researchers to gain insights into various aspects of model performance. However, it is vital to use these metrics in conjunction with each other and with other evaluation techniques to ensure a thorough assessment of model accuracy and effectiveness.

2.11 Chapter Summary

This chapter introduces the theoretical underpinnings of the proposed automated **NN dataset generation approach for human skeleton detection**. It delves into the complexities of human pose estimation in real-world environments and introduces key concepts essential for understanding subsequent discussions.

Challenges in Real-World Human Pose Estimation

The chapter begins by elucidating the challenges inherent in human pose estimation in real-world scenarios (**Section 2.1**). These challenges include **varying distances** between individuals and cameras, **occlusions**, and **crowded scenes**. The chapter underscores the importance of addressing these challenges to develop effective pose estimation models.

Neural Networks: Fundamentals and Architectures

A fundamental understanding of NNs is crucial for comprehending automated dataset generation approaches (Sections **2.2**, **2.3** and **2.4**). The chapter introduces **NNs**, **CNNs**, and **RCNNs**, outlining their roles and applications in skeleton detection tasks.

Existing NNs for Human Pose Estimation

The chapter explores existing NN models tailored for human pose estimation (Sections **2.6**, **2.7** and **2.8**). Noteworthy models such as **PoseNet**, **MoveNet**, and **MMPose** are discussed in detail, elucidating their architectures, functionalities, and performance characteristics. Understanding these models lays the foundation for developing novel approaches to dataset generation.

Evaluation Metrics for Detection Performance

Lastly, the chapter delves into the evaluation metrics employed to assess the performance of pose estimation models (**Section 2.10**). Metrics such as **OKS**, **APE** and **MSE** are explained, along with their significance in gauging the accuracy and efficacy of detection algorithms.

This chapter provides a comprehensive overview of the theoretical foundations for understanding the proposed automated NN dataset generation approach for human skeleton detection. By elucidating the challenges, introducing key concepts, and exploring existing models and evaluation metrics, this chapter lays the groundwork for the subsequent discussions on dataset generation methodologies and experimental analyses.

3 Practical Part

This chapter comprehensively examines the various stages of creating a custom human pose estimation dataset. The initial phase introduces the solution proposal and details the whole process. The following phase leverages existing models outlined in **Section 2.5** (on **page 21**). Subsequently, these models are integrated into a unified tool for custom dataset creation. Additionally, the tool's performance is evaluated using three metrics, OKS APE and MSE.

Throughout this thesis, several implementation challenges emerged, leading to certain technical limitations outlined in a dedicated **Section 3.6** on **page 57**.

3.1 Solution Proposal

This section will propose solutions for the specific dataset generation using existing **NN** detection models. The content of this section outlines one of the possible solutions, grounded in the analysis from the preceding paper chapter. It specifies the technologies that should be used and explains their suitability for this task.

The preceding chapter conducted an in-depth analysis of the current market landscape, exploring available technologies and datasets and their inherent limitations. As highlighted in **Section 2.1**, existing datasets lack the complexity and robustness necessary for training a comprehensive detection model capable of excelling in production environments.

A cost-efficient method for generating a tailored dataset using open-source tools and models was explored to address this shortfall. **Python** emerged as a prime choice in selecting the technological foundation due to its widespread adoption in data processing and machine learning. Python offers an extensive array of libraries such as **Numpy** for numerical computations, **Pandas** for data manipulation and analysis, and deep learning frameworks like **PyTorch** and **TensorFlow**. This rich ecosystem simplifies the implementation of complex algorithms and fosters innovation and efficiency in dataset generation processes.

Initially, the production data executes existing models designed to detect human instances and estimate their poses. Namely, these include **PoseNet**, **MoveNet**, and **MMPose** models described in the Sections **2.7**, **2.6**, and **2.8**. The primary reason for applying these models is outlined in **Section 2.5**, which includes the models' hardware requirements and ease of implementation in

Python, the underlying technology proposed in this section. Following the execution, multiple detection formats become available. It is crucial to propose a Unified Format for these detections, considering their respective keypoint structures and identifying a common skeleton. Once the Unified Format is established, the unification process can commence, utilizing the existing detections from the initial step.

Subsequently, the sequential processing of these detections becomes essential, wherein corresponding frames are loaded from each detection model. Subsequent steps involve matching individual human instances by comparing instance **BBOX**-es or individual keypoint analysis. Once instances are paired, aggregation occurs, with methods varying from weighted to simple averaging based on the utilized models and their respective accuracy levels.

This process leads to the complete dataset processing, resulting in aggregated detection results. Processing one source frame at a time transforms all detections from individual models into the Unified Format.

Finally, evaluation becomes necessary, comparing the Unified Format against an annotated dataset (with a similar skeleton as the Unified Format) and providing ground truth data for assessment. For evaluation, several metrics need inclusion to cover the overall performance of this Unified Format, including the **APE**, **MSE**, and **OKS** metrics.

A significant advantage of this approach lies in its flexibility; it is not restricted to specific models and can be fully customized to suit specific requirements. This implies combining any available detection model (or any number of detection models) and using them in the dataset generation process. The usage of existing models does not need to be confined to Python scripts introduced in this thesis. The only condition is to run the detection on some production data and prepare the results for the unification process.

3.2 Solution Implementation

The moment has arrived to delve into the realm of implementation. This section will present an overview of the entire process, aiming to enhance comprehension of the thesis as a cohesive entity. Central to this elucidation will be the graphical representation of the sequential steps undertaken in this thesis. These steps will be comprehensively expounded upon in subsequent sections, offering detailed insights into their execution.

To elucidate the process further, refer to **Figure 3.1**. The initial phase entails employing the existing skeleton detection models outlined in Sections **2.7**, **2.6**, and **2.8**. This step involves the utilization of custom-implemented scripts designed to execute detections using the models above, as elaborated in the subsequent

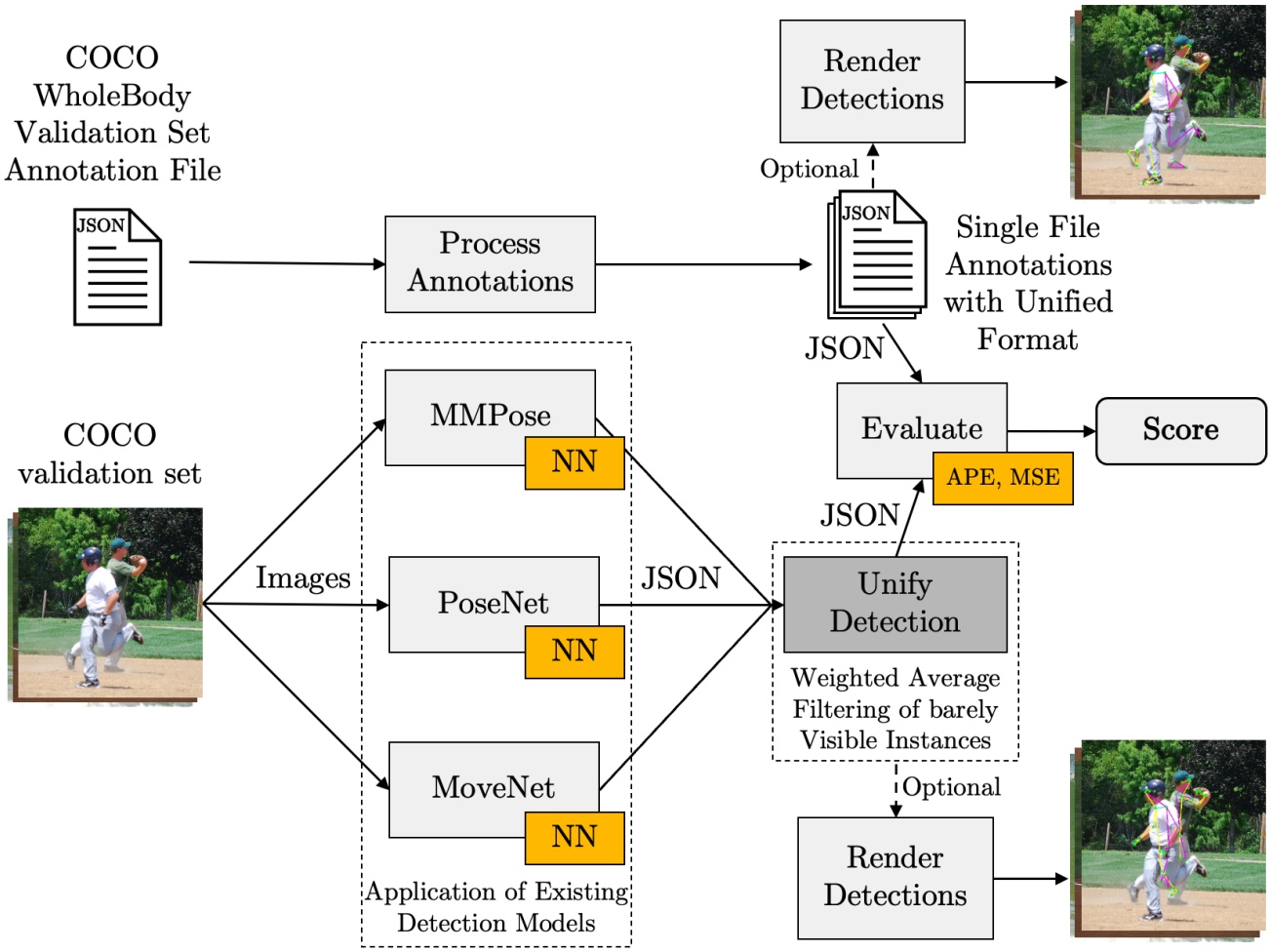

Figure 3.1
Graph of the process introducing individual stages of this thesis

Section 3.3. The detection process is conducted on the evaluation subset comprising 5,000 images from the COCO dataset. Outputs from individual detections are stored in distinct JSON files, the format of which is detailed in Code 3.5.

These JSON files serve as inputs for the Unification script, which also requires keypoint map files for each detection model. These map files facilitate the transformation of native model outputs into the Unified Format mandated by the script. The Unification script aggregates individual detections into a specified format, generating a JSON file for each image (or video, depending on the script argument). Optionally, these Unified Format JSON files can be utilized for detection rendering, a functionality provided by the custom Render script.

Concurrently, the COCO WholeBody annotations (comprising 133 keypoints) undergo processing in a custom script. This script takes a single JSON file containing annotations for the entire subset (either training or validation) and produces individual JSON outputs for each image within the subset. Additionally, it generates detections in the same format as the Unification script,

consolidating annotations for various body parts into a simplified 27 keypoint format. Optionally, these annotations can be rendered onto the subset images for visual analysis.

The final step entails evaluating the performance of the Unified Format. For this purpose, a custom script is employed, which compares the annotations (ground truth) with the Unified Format detections (predictions), calculating various statistical metrics detailed in **Section 3.5**.

3.3 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 implementation tools are 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.

(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 (See **Code 3.2**).

```python
from typing import List, Dict, Set

DetectionInstance = List[List[float]]
Frame = List[DetectionInstance]
Detection = Dict[str, Frame]
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 3.3.1**. Then follows the **Frame**, a list of *detection instances*. At the top of this hierarchy lies the **Detection** itself, a dictionary of the *frames*. Lastly, the final custom datatype, **SkeletonEdgesPainting**, is a structure that stores 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 similar interface with the same functionality. Each detection model uses different resources, meaning the detection part is different for every case and must be implemented separately. Similarly, the model initialization parts are different. However, the script skeleton remains the same for every model.

The scripts utilize standard 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 *main()* function structure in the following **Code 3.3**. 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, executing detection on each. Refer to the single-frame detection function *process_image_detection()* in **Code 3.4** for a deeper understanding of the core detection implementation.

This function demonstrates the concept behind image detection execution for the MoveNet model. Similar MMPose and PoseNet models use similar approaches with specific detection implementation and visualization variations. The first task is to load the frame or image. Optionally, a function for resizing the frame is available if the image is too large. Then follows the pose detection itself. Program arguments manage actions like saving the detection file (JSON), the frame

```
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)

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.

3.3.1 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 3.4** on **page 43**.

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 **Code 3.5**. 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 quickly processed detection files.

```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.

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

```
[
   {
      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)

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.

3.4 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 training in a new model tailored to the target detection task. However, for practical 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. At the same time, **MMPose** includes **107** unnecessary keypoints, exceptionally 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. From the skeleton figure, it is visible that the format is straightforward. It offers essential pose representation with sufficient detail to both hands and feet.

Another crucial aspect is the accurate aggregation of individual model estimations, encompassing the coordinates of keypoints and their visibility values. A weighted average of these values was introduced to address this, 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 (From manual inspection of individual detections, the PoseNet and Movenet often provide significant errors in pose estimation, which can be explained by the fact that they are more focused on real-time use-cases. There is always a trade-off between the detection time and accuracy.). Detailed values for the weighted average are available in the model's comparison table. The assignment of the highest weight to the **MMPose** model is justified by its superior accuracy in all metrics described in the **Section 3.5**. This approach ensures the uniformity of estimations made by individual models across the entire dataset, as prepared in the previous section.

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	–

The weighted average is not applied in scenario processing keypoints on the hands or feet. This limitation arises because 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 data is from all three models.

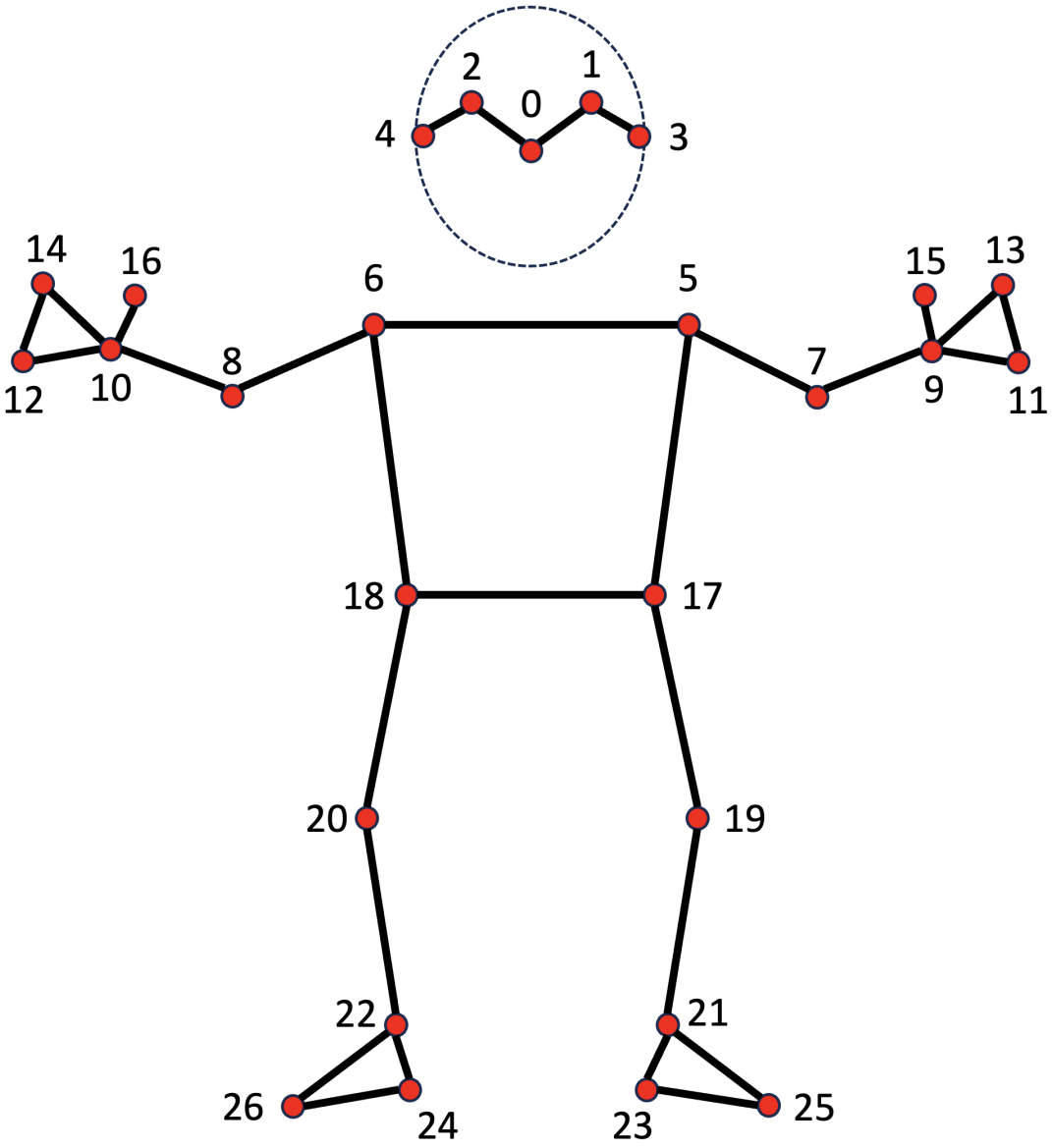

Figure 3.6
Unified format structure with IDs to each keypoint

To clarify and explain the detection outcomes, refer to Figures 3.7, 3.8 and 3.9. The initial depiction presents the original COCO WholeBody annotations in a simplified format, mirroring the Unified Format with 27 keypoints. All individuals within the image are annotated, meticulously considering the overlap and occlusion of human figures. Additionally, non-visible skeleton edges are omitted. Notably, no facial keypoints are detected, aligning with the absence of visible faces, thus demonstrating accurate annotation.

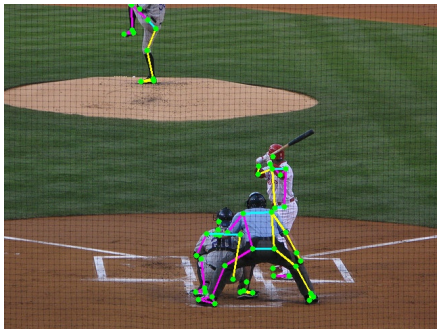

Figure 3.7
COCO WholeBody annotation simplified to Unified Format

Figure 3.8
MMPose model detection with 133 keypoints format

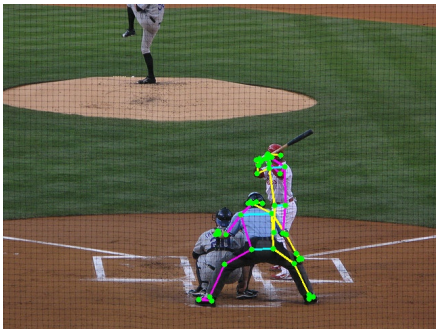

Figure 3.9
Unified format detection with 27 keypoints

Subsequently, **Figure 3.8** portrays a detection outcome from the MMPose model. Regrettably, the individual performing a deep squat remains undetected, while facial keypoints are erroneously identified for two adjacent individuals facing away from the camera. Despite these anomalies, the overall detection performance is commendable.

Lastly, the Unified Format, primarily informed by the MMPose model, is depicted in the third Figure. Notably, a single individual in the upper portion of the image remains undetected. This occurrence is attributed to the unification script, which employs an **average instance visibility threshold**, set experimentally at **40%**, to determine inclusion in the Unified Format. This threshold serves as a filter for wrongly detected human instances by one of the models. Usually, this occurs using the MMPose model, a multi-person detector. Nevertheless, the precision of individual detections is highly satisfactory, ensuring precise localization of keypoints when individuals are successfully detected within the image. This is proven in the evaluation **Section 3.5** at **page 52**.

3.4.1 Unification Process Implementation

This subsection aims to describe the unification script key concepts alongside the code examples. The whole process is described in the previous section. Now, the implementation details will be described.

The unification script facilitates processing detection data from multiple sources into a unified format. It offers flexibility in handling various input configurations and provides options for customizing the output structure. Check out the **Code 3.10** to see the program arguments documentation, which can be described as follows:

```
python3 unified_detection_processor.py
    [--source-folder SOURCE_FOLDER]
    [--output-folder OUTPUT_FOLDER]
    [--models-detection-folders [MODELS_DETECTION_FOLDERS ...]]
    [--skeleton-map-files [SKELETON_MAP_FILES ...]]
    [--single-frame-output]
    [--include-bbox]
    [--instance-visibility-threshold INSTANCE_VISIBILITY_THRESHOLD]
    [--verbose]
```

Source code 3.10
Command for running the Unification script and its options

- **–source-folder**: Specifies the main source folder containing models detection subfolders.
- **–output-folder**: Specifies the destination folder for the unified detection results.
- **–models-detection-folders**: Specifies the folders to process, allowing multiple folders to be provided (For each detection model, a separate folder with detections.).
- **–skeleton-map-files**: Specifies the files containing unified skeleton mappings, allowing multiple files to be provided (For each detection model, separate mapping file.).
- **–single-frame-output**: Optional flag indicating that only image sources (not videos) are expected, resulting in simplified output structure (The top-level dictionary is changed to a simple list with detection instances.).
- **–include-bbox**: Optional flag indicating to include BBOX information for each detection instance.
- **–instance-visibility-threshold**: Optional parameter specifying the visibility threshold for removing instances from detection unification. The default value is 0.4. If the average instance visibility is below the threshold, the detection will not be included in the output.
- **–verbose**: Optional flag enabling verbose mode for detailed output information.

When the program arguments are understood, let us look at the main function structure, which is available in **Code 3.11**.

The first step is to deal with the program arguments explained above. After the arguments are processed, the main loop begins, where it sequentially processes the individual detection files (from each detection model). The *load_detections_objects()* function is a generator which streams the filename of the current processing detection file (the shared name by all detection models) and the dictionary with the individual model's raw detections loaded into the *DetectionProcessor* class, which provides an interface for the manipulation. For example, the *get_unified_detection()* method processes the raw detection and simplifies it to

```python
def main():
    """Unify the detection results done by different models.
    """

    args = parse_arguments()
    unified_detections = 0

    # Generate dict of detection objects by model,
    # each model with the name of the source file
    for filename, detection in load_detections_objects(args):
        unified_detection = UnifiedDetection(
            args.output_folder,
            filename,
            args.single_frame_output
        )
        model: DetectionProcessor
        model_weight_for_averaging = []
        for model in detection.values():
            if args.verbose:
                print(f'{unified_detections}. ({model.model})
                Weight: {model.weight_for_averaging},
                Loading detection: {model.source_file}.')
            model_weight_for_averaging.append(model.weight_for_averaging)
            unified_detection.load_unified_kepoints(
                model.get_unified_detection()
            )
        unified_detection.simplify_unified_detection()
        unified_detection.aggregate_values(model_weight_for_averaging)
        unified_detection.export_unified_detection()
        unified_detections += 1


    print(f'Successfuly unified {unified_detections} detections
    to "{args.output_folder}".')

if __name__ == '__main__':
    main()
```

Source code 3.11
Unified Format Class

a Unified Format (possible NULL values, for example MoveNet detects only 17 keypoints, Unified Format has 27, resulting in 10 NULL keypoints) defined by the skeleton files for each detection model. These skeleton files map each detection model output to the Unified Format output. In other words, to obtain the Unified Format from the model's detection, which keypoints need to be used, and to what place do they correspond in the Unified Format? They are in the JSON format, and the keypoints map can be found in the **Code 3.12**.

The skeleton files contain the indexes from the original detection mapped to the corresponding keypoints in the Unified Format. Another vital piece of information is this model's **Weight** for averaging purposes. The last item is the original detection description in the manner of keypoint names (*nose*, *left_eye*, *right_eye*, ...).

```
{
    "unified_skeleton_kpt_idx": [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, null, null,
    null, null, null, null, 11, 12, 13, 14, 15, 16, null, null, null, null],
    "weight_for_averaging": 0.2,
    "keypoint_map": {
        "0": "nose",
        "1": "left_eye",
        "2": "right_eye",
        "3": "left_ear",
        "4": "right_ear",
        "5": "left_shoulder",
        "6": "right_shoulder",
        "7": "left_elbow",
        "8": "right_elbow",
        "9": "left_wrist",
        "10": "right_wrist",
        "11": "left_hip",
        "12": "right_hip",
        "13": "left_knee",
        "14": "right_knee",
        "15": "left_ankle",
        "16": "right_ankle"
    }
}
```

Source code 3.12
MoveNet Skeleton File with Keypoints Map to Unified Format

Back to the process loop. The next step is to load all individual model detections into the object *UnifiedDetection*, which will be the main manipulator of the detection format. The main task of this class is to assemble the final Unified Format detection file for either an image or a video. After this object is initialized, there comes a time to loop through the individual model's detection provided by the *get_unified_detection()* method. The output of this method goes directly to the *UnifiedDetection* object method *load_unified_keypoints()*, which appends the detection to the internal structure.

When all detections from every model are loaded, the simplification process begins. This serves for clearing the *name* attribute from the internal structures of the Unified Format, which is available in **Code 3.13**. This class (*UnifiedFormat*) represents a single human detection instance. The class provides the functionality for manipulating the detection on the lower level than the *UnifiedDetection*, which provides an interface for the detection of the whole image. In other words, every method on the UnifiedDetection level eventually leads

to the method on the UnifiedFormat level. The *name* attribute was included to interpret the structure better. However, for the aggregation process, it does not provide any value, which is why it is removed before further processing.

```
class UnifiedFormat:
    """Template for the unified format structure."""

    def __init__(self) -> None:
        self.unified_format = {
        0: dict(x_axis=[], y_axis=[], visibility=[], name="nose"),
        1: dict(x_axis=[], y_axis=[], visibility=[], name="left_eye"),
        2: dict(x_axis=[], y_axis=[], visibility=[], name="right_eye"),
        3: dict(x_axis=[], y_axis=[], visibility=[], name="left_ear"),
        4: dict(x_axis=[], y_axis=[], visibility=[], name="right_ear"),
        5: dict(x_axis=[], y_axis=[], visibility=[], name="left_shoulder"),
        6: dict(x_axis=[], y_axis=[], visibility=[], name="right_shoulder"),
        7: dict(x_axis=[], y_axis=[], visibility=[], name="left_elbow"),
        8: dict(x_axis=[], y_axis=[], visibility=[], name="right_elbow"),
        9: dict(x_axis=[], y_axis=[], visibility=[], name="left_wrist"),
        10: dict(x_axis=[], y_axis=[], visibility=[], name="right_wrist"),
        11: dict(x_axis=[], y_axis=[], visibility=[], name="left_pinky"),
        12: dict(x_axis=[], y_axis=[], visibility=[], name="right_pinky"),
        13: dict(x_axis=[], y_axis=[], visibility=[], name="left_index"),
        14: dict(x_axis=[], y_axis=[], visibility=[], name="right_index"),
        15: dict(x_axis=[], y_axis=[], visibility=[], name="left_thumb"),
        16: dict(x_axis=[], y_axis=[], visibility=[], name="right_thumb"),
        17: dict(x_axis=[], y_axis=[], visibility=[], name="left_hip"),
        18: dict(x_axis=[], y_axis=[], visibility=[], name="right_hip"),
        19: dict(x_axis=[], y_axis=[], visibility=[], name="left_knee"),
        20: dict(x_axis=[], y_axis=[], visibility=[], name="right_knee"),
        21: dict(x_axis=[], y_axis=[], visibility=[], name="left_ankle"),
        22: dict(x_axis=[], y_axis=[], visibility=[], name="right_ankle"),
        23: dict(x_axis=[], y_axis=[], visibility=[], name="left_heel"),
        24: dict(x_axis=[], y_axis=[], visibility=[], name="right_heel"),
        25: dict(x_axis=[], y_axis=[], visibility=[], name="left_foot_index"),
        26: dict(x_axis=[], y_axis=[], visibility=[], name="right_foot_index"),
        27: dict(coordinates=[], name="bbox")
        }
```

Source code 3.13
Unified Format Class

Now comes the most critical task of aggregating the detections and formulating the final Unified Format. For this purpose, the UnifiedDetection object has *aggregate_values()* method, which takes a list of weights for individual models in order as the detections were processed. Internally, for each human instance in the detection, the *aggregate_instance()* method of the class *UnifiedFormat* is called with the list of weights as an argument. This method calculates the weighted average for the keypoints present in all individual detections (from every model). Otherwise, the simple average is calculated if values from some

model are missing. Generally, the keypoints of the hands and feet are not present in the MoveNet detections, leading to an average of just two values from the other models.

When the aggregation went without problems, it was time to export the Unified Detection into a single JSON file. This functionality is provided by the *export_unified_detection()* method, which generates either a single-frame format or multi-frame output according to a program argument (*–single-frame-outpu*). The output format is shown in **Code 3.14**. Generally, the difference between the single and multi-frame output is that the multi-frame output has a **dictionary** on the top level, where the keys identify individual frames (In the image detections, just one item is present with key 'frame_1'. The video detections contains more than one frame leading to multiple records in the dictionary.). On the other hand, the single-frame output has a **list** on the top level, where this list contains human detection instances directly.

```
[
   [
      [
         429.6509704589844,
         170.8621368408203,
         0.9522638320922852
      ],
      ...
      [
         446.5588684082031,
         289.245361328125,
         0.33409273624420166
      ],
      [
         423,
         159,
         466,
         300
      ]
   ]
]
```

Source code 3.14
Unified Format Output (Single-Frame Version

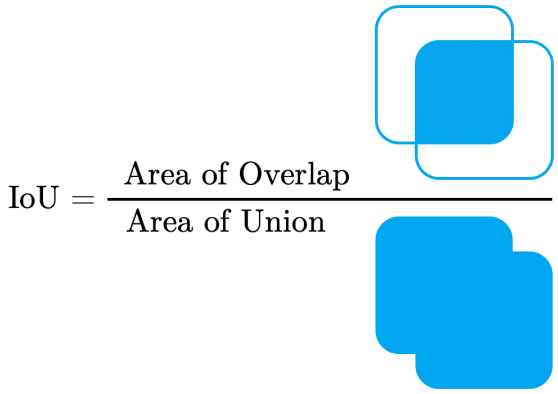

Figure 3.15
Calculation of the Intersection over Union

3.5 Evaluation

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

As for the unified detection format evaluation, the **COCO WholeBody** dataset was used. Specifically, the **validation 2017** subset containing **5000** images. According to a COCO WholeBody validation annotations, only **2693** images have present humans.

The overall metrics results are as follows (see the **Table 3.2**): the **OKS** equal to **0.23**, **APE** equal to **3.3%**, the **MSE** equal to **1213.84** and finally, the total **correctly detected keypoints** **37.93%** (used OKS threshold **0.45**). These metrics were 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 considered—specifically the missing or additional (detection) instances in the predictions. However, as for the corresponding instances, the evaluation serves as a great indicator of how well the individual detection instances were obtained. The criterion for defining whether the instances from predictions correspond to the ground truth annotations is the BBOX overlap. The threshold used for the overlap value, **Intersection over Union** (IoU), is the **65%**. To better understand the IoU metric, see the **Figure 3.15**. The result is satisfactory even though averaging the keypoint's position in the unification process causes some errors.

There was also an evaluation on the metrics just for the **MMPose** model because only this model provides us with the BBOX-es in the detections. The total **OKS** was **0.28**, the **APE** was **0.95%**, the **MSE** was equal to **123.73** and

the total **correctly detected keypoints** was equal to **43.62%**. This evaluation results provided 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 3.4** on **page 43**. Generally, the MMPose model performs well in detection tasks. Visually, there are only occasional differences in the prediction and the ground truth.

As for the other statistics, the evaluation concerned the **2693** ground truth images with overall **11004** pose instances (humans). The Unified Format provided overall **7746** (**70.39%**) detection instances from which the **5833** were matched with the ground truth poses based on the **IoU** metric. Very similar results were achieved by the **MMPose** model, which detected overall **7685** (**69.84%**) detection instances, with the same **5833** matched to the ground truth. The Unified Format provided **589** (**5.35%**) additional detection poses than were in the ground truth annotations and **3847** (**34.96%**) fewer detections than were in the ground truth. Similarly, the **MMPose** model detected **589** (**5.35%**) more poses and **3908** (**35.51%**) fewer poses than the ground truth annotations contained.

Table 3.2 Evaulation Statistics for the Unified and MMpose Format

Statistic	**Unified F.**	**MMPose**
Total Object Kepoint Similarity (OSK):	0.23	0.28
Total Average Percentage Error (APE):	3.3%	0.95%
Total Mean Squared Error (MSE):	1213.84	123.73
Total correct keypoints detected:	37.93%	43.62%
Total matching number of poses (instances):	5833 (53.01%)	5833 (53.01%)
Total predicted number of poses (instances):	7746 (70.39%)	7685 (69.84%)
Model predicted more poses in image (total):	589 (5.35%)	589 (5.35%)
Model predicted less poses in image (total):	3847 (34.96%)	3908 (35.51%)
Total truth number of poses (instances):	11004	11004
Total number ground truth images:	2693	2693

The evaluation statistics for the **Unified Format** and **MMPose** model can be found in Table **Table 3.2**. This table presents various values, ranging from the overall counts of processed images with ground truth annotations to differences in predicted pose counts. The APE metric is easily interpretable, indicating the average error of predicted keypoint positions relative to ground truth annotations per pose. The value of **3.3%** for APE signifies that the predicted detection

Figure 3.16
MMPose simplified output

Figure 3.17
COCO Annotations

Figure 3.18
Unified Format

closely aligns with the ground truth annotation. Conversely, the OKS metric yields a somewhat concerning result, with the value of **0.23** falling towards the lower end of the metrics interval ($< 0 , 1 >$). Generally, higher values of OKS are preferable, indicative of better alignment. The Evaluation script considers a keypoint similar when the OKS for that particular keypoint exceeds the **0.45** threshold (This threshold was set experimentally.).

For a comprehensive understanding of these values, including MSE, refer to example images and their metrics in Figures **3.16** and **3.17**. Additionally, Table **Table 3.3** offers detailed statistics for these two detections. The first Figure illustrates a simplified MMPose model output (consisting of **27 keypoints** in the Unified Format). In contrast, the second Figure also displays the original COCO Wholebody annotations in the simplified format. This comparison facilitates a deeper understanding of the available metrics.

Table **Table 3.3** encompasses evaluation metrics for both the MMPose detection model and the Unified Format on a specific image (**3.16**). This comparison aids in visualizing the APE and MSE metrics. For the MMPose detection, it is evident that the prediction closely aligns with the ground truth pose, resulting

in a low APE of **2.51%**. In contrast, for the Unified Format, as depicted in **Figure 3.18**, the prediction is significantly inaccurate, shifted to the left bottom of the image, while the person is visible in the middle. This discrepancy arises from an incorrect pose prediction by one of the models (PoseNet or MoveNet), which, during the unification process, was averaged with the correct MMPose detection, leading to an erroneous combined prediction.

Table 3.3 Single Image Metrics Evaluation and Comparison

Statistic	**MMPose**	**Unified F.**
Image Average Percentage Error (APE):	2.51%	84.22%
Image Mean Squared Error (MSE):	70.67	53285.03
Image Object Kepoint Similarity (OSK):	0.66	0.00
Image average of correct keypoint detected:	44.44%	00.00%

Nevertheless, this is an excellent illustration for explaining evaluation metrics, where for this particular pose, the **APE** is **84.22%**, indicating a substantial deviation from the ground truth pose. Regarding the **MSE**, this pose resulted in an average squared error (Euclidean distance between predictions and ground truth) of **53, 285.03**, highlighting significant distances between the prediction and ground truth pose.

Finally, let us delve into the explanation of the **OKS** metric. The MMPose prediction can be interpreted as a satisfactory result, with an **OKS** value of **0.66**, positioning it in the upper part of the interval. Notably, the **OKS** metric demonstrates a stringent assessment criterion; even for highly similar poses, the value may not reach a high threshold, making it challenging to gauge similarity accurately.

Regarding this specific image, the COCO WholeBody annotation appears insufficient, while the MMPose model's pose estimation seems more accurate. This assertion may elucidate the low **OKS** value despite the precision of the prediction.

Conversely, the Unified Format prediction receives an **OKS** value of **0.00**, correctly indicating it as "not similar at all." This stark contrast underscores the divergence between the Unified Format prediction and the ground truth annotation.

3.5.1 Implementation of the Evaluation Process

This subsection describes the evaluation script's fundamental concepts and includes code examples. While the preceding section outlines the entire process, this section delves into the specifics of its implementation.

The core concept of the evaluation script revolves around iterating through the ground truth annotations for individual files and systematically comparing the corresponding detection instances within the images. For each image, the script compares the ground truth detection instances with the predicted ones, storing the results in a dictionary cataloguing all available pairs of instances along with their Intersection over Union (IoU) metrics. Subsequently, the script selects pairs with the highest bounding box (BBOX) overlap within the loop. These instances then undergo processing by a function that, for each keypoint, calculates the distance (refer to **Code 3.19**) and supplies this value to the individual evaluation metric computations.

```
1 def calculate_distance(point1, point2):
2   """Calculates the Euclidean distance between two points."""
3
4   x1, y1 = point1
5   x2, y2 = point2
6   return ((x2 - x1) ** 2 + (y2 - y1) ** 2) ** 0.5
```

Source code 3.19
Implementation of Euclidean Distance

A crucial aspect of the function for single-pose evaluation is its ability to skip invisible keypoints from the ground truth annotations. This exclusion enhances the credibility of the Object Keypoint Similarity (OKS) metric by reducing the total number of keypoints, which is integral to the averaging process. Subsequently, this function propagates the aggregated results to higher levels of analysis.

Another pivotal component of the evaluation process is calculating the OKS metric, which involves determining the *k_values*—per-keypoint constants recommended by COCO in their documentation to fine-tune the calculation. The original constants provided by the COCO evaluation script are inadequate as they only cover 17 keypoints (the COCO dataset's standard), while our dataset features 27 keypoints in a Unified Format. To address this disparity, the *get_k_values()* function (refer to **Code 3.20**) computes the per-keypoint constants based on the averaged standard deviations obtained from the initial evaluation run. Upon processing the entire dataset (COCO evaluation subset), the errors (dis-

tances) between ground truth and prediction are averaged for each keypoint. Subsequently, these newly generated values are exported to a JSON file, and for subsequent evaluation runs, they serve as the basis for OKS metric calculations.

```python
def get_k_values(
  average_squared_distances: List[float],
  base_k: float = 1.0,
  scale_factor: float = 0.1
  ):
  """Calculates k values based on average squared distances without object scale."""

  k_values = []
  for avg_squared_distance in average_squared_distances:
    # Calculate standard deviation
    sigma = math.sqrt(avg_squared_distance)
    # Adjust k based on standard deviation
    k_values.append(round((base_k + sigma * scale_factor) / 100, 3))
  return k_values
```

Source code 3.20
Function for Calculating Per-Keypoint Values for OKS Metric

3.6 Implementation Problems and Technical Limitations

Throughout 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 implement the 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.

Figure 3.21
MMPose

Figure 3.22
PoseNet

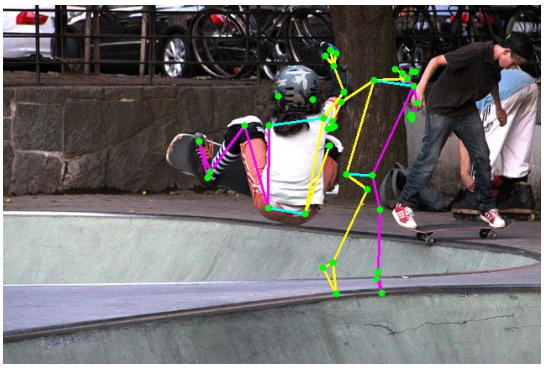

Figure 3.23
Unified Format

Figure 3.24
COCO Annotations

There are also some implementation limitations regarding the creation of the Unified Format. A problem arises when averaging detections, primarily when some models do not provide BBOX values. For instance, PoseNet and MoveNet do not generate BBOX information, making it challenging to map the appropriate detection instances for averaging. This issue is particularly pronounced in examples from the COCO evaluation subset, as illustrated in Figures 3.21, 3.22, and 3.23.

In Figure 3.22, for instance, the mapping of the jumping person from the PoseNet model in the middle of the Figure is incorrectly associated with a different instance from the MMPose model, depicted as a person on the right. Consequently, averaging these two detections results in the placement of the detection in between in the Unified Format output. Although the unified version of the detection appears slightly distorted, it demonstrates how the averaging process combines the original detections. In this case, the MoveNet model did not provide any results, prompting the unification script to consider only MMPose and PoseNet detections.

These issues could be addressed by implementing a custom function that compares two instances from the detection scripts and returns a similarity constant. This constant would serve as a basis for determining if the two detection instances correspond. Utilizing the BBOX information provided by the MMPose model, for example, could facilitate comparison by defining an area of overlap for individual keypoints from the other two models if a certain percentage of keypoints fit into a BBOX (for example, 65%, similar to an IoU threshold introduced in **Section 3.5**), the instances would be considered corresponding.

Another problem arose during the evaluation of detection performance. The **OKS** metric utilizes the object scale to normalize the value and mitigate the disproportionate influence of larger objects. Initially, the object scale is calculated as the product of *bbox_width* and *bbox_height* multiplication, which is then squared in the **OKS** formula. However, this approach yielded excessively high values, resulting in nearly perfect **OKS** values even for completely inaccurate poses. By eliminating the squaring process, the **OKS** values returned to a normal range, accurately representing the similarity between poses.

4 Conclusion

The development of a **Unified Format** for **human pose estimation dataset creation** represents a significant advancement in **simplifying and standardizing** the **aggregation of results** from diverse NN models. By addressing the variations in output formats among existing models like MoveNet, PoseNet, and MMPose, the Unified Format harmonizes these outputs into a cohesive structure. This achievement streamlines the dataset generation process, enabling the training of tailored models for specific detection tasks.

The evaluation of the Unified Format's performance revealed promising results, particularly in metrics like **APE** (**3.3%**) and **MSE** (**1213.84**), where low values indicate close alignment between predicted and ground truth keypoints. The **OKS** metric (**0.23**), though helpful in measuring similarity, faced challenges because of strict evaluation standards and differences in predictions between models. Nonetheless, the Unified Format demonstrated **satisfactory performance** accurately **localizing keypoints**, especially when compared to individual model outputs. The low value of the OKS metric can be attributed to the fact that the selection of detection models prioritized ease of use and speed over maximum accuracy. In real-world applications, where creating specific datasets is paramount, employing more accurate models focused on precision rather than real-time usage would undoubtedly yield higher OKS scores.

Despite the overall success of the Unified Format, several areas warrant further attention for refinement. Firstly, **enhancing** the **unification process** to better handle instances where individual models produce **divergent results** could improve overall accuracy. This could involve refining the weighting scheme for averaging predictions or implementing **adaptive strategies** to accommodate varying model outputs.

Additionally, investigating techniques to mitigate errors introduced during the unification process, such as erroneous pose predictions, could lead to more robust dataset generation. Strategies like **outlier detection** or **dynamic thresholding** based on model confidence scores may help improve the quality of Unified Format outputs.

Furthermore, exploring **alternative evaluation metrics** or refining existing ones, mainly **OKS**, to better reflect the nuances of pose estimation accuracy could enhance the assessment process. This may involve adjusting keypoint similarity criteria or incorporating contextual information to account for pose variations in real-world scenarios.

In summary, the practical part of the thesis has successfully addressed the challenges associated with human pose estimation dataset creation by proposing and implementing a **Unified Format**. This format facilitates the aggreg-

ation of outputs from multiple NN models, streamlining the dataset generation process. Evaluation results indicate promising performance, with low APE and MSE values demonstrating close alignment between predicted and ground truth keypoints. While challenges remain, continued refinement and exploration of techniques offer opportunities to improve the Unified Format further and advance the field of human pose estimation.

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.

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.

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]. x 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.

Andriluka Mykhaylo, Pishchulin Leonid, Gehler Peter 2D Human Pose Estimation: New Benchmark and State of the Art Analysis. In *2014 IEEE Conference on Computer Vision and Pattern Recognition* [on-line!]. 2014 [cit. 2024-04-12]. Available at: http://dx.doi.org/10.1109/CVPR.2014.471.

De Myttenaere Arnaud, Golden Boris, Le Grand Bénédicte Mean Absolute Percentage Error for regression models. In *Neurocomputing* [on-line!]. 2016 [cit. 2024-04-20]. Available at: https://doi.org/10.48550/ARXIV.1605.02541.

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.

COCO Consortium COCO Dataset: Keypoints Evaluation. In *COCO* [on-line!]. 2024 [cit. 2024-04-20]. Available at: https://cocodataset.org/#keypoints-eval.

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.

Li Jiefeng, Wang Can, Zhu Hao CrowdPose: Efficient Crowded Scenes Pose Estimation and a New Benchmark. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* [on-line!]. June 2019 [cit. 2024-04-14]. Available at: http://dx.doi.org/10.1109/CVPR.2019.01112.

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.

Neumann Lukáš, Vedaldi Andrea *Tiny people pose.* Cham : Springer International Publishing, 2019. [cit. 2024-04-14]. x pp. Available at: https://www.robots.ox.ac.uk/~vgg/publications/2018/Neumann18a/neumann18a.pdf. DOI: 10.1007/978-3-031-09282-4_22.

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.

Song-Hai Zhang, Ruilong Li, Xin Dong Pose2Seg: Detection Free Human Instance Segmentation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* [on-line!]. 2019 [cit. 2024-04-14]. Available at: https://doi.org/10.48550/arXiv.1803.10683.

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.

Wu Jiahong, Zheng He, Zhao Bo Large-Scale Datasets for Going Deeper in Image Understanding. In *2019 IEEE International Conference on Multimedia and Expo (ICME)* [on-line!]. Jul 2019 [cit. 2024-04-12]. Available at: http://dx.doi.org/10.1109/ICME.2019.00256.

Zhang Yiqing, Chen Weiting Decision-level information fusion powered human pose estimation. In *Applied Intelligence* [on-line!]. 2022 [cit. 2024-04-20]. Available at: https://doi.org/10.1007/s10489-022-03623-z.

List of Tables

List of Figures

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
PR	Precision-Recall
RCNN	Region-based convolutional neural network
SSD	Single Shot MultiBox Detector

List of Source Codes

APPENDICES