Human Pose Estimation for Decubitus Prophylaxis

Benedikt Weber

Technische Berichte Nr. 153

des Hasso-Plattner-Instituts für
Digital Engineering an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Digital Engineering an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Digital Engineering an der Universität Potsdam | 153

Benedikt Weber

Human Pose Estimation for decubitus prophylaxis

Universitätsverlag Potsdam

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available on the Internet via `http://dnb.dnb.de/`.

Universitätsverlag Potsdam 2023
`http://verlag.ub.uni-potsdam.de/`

Am Neuen Palais 10, 14469 Potsdam
Phone: +49 (0)331 977 2533 / Fax: 2292
Email: verlag@uni-potsdam.de

The series **Technische Berichte des Hasso-Plattner-Instituts für Digital Engineering an der Universität Potsdam** is edited by the professors of the Hasso Plattner Institute for Digital Engineering at the University of Potsdam.

ISSN (print) 1613-5652
ISSN (online) 2191-1665

Layout: Tobias Pape
Print: docupoint GmbH Magdeburg

ISBN 978-3-86956-551-4

Also published online on the publication server of the University of Potsdam:
`https://doi.org/10.25932/publishup-56719`
`https://nbn-resolving.org/urn:nbn:de:kobv:517-opus4-567196`

Decubitus is one of the most relevant diseases in nursing and the most expensive to treat. It is caused by sustained pressure on tissue, so it particularly affects bed-bound patients. This work lays a foundation for pressure mattress-based decubitus prophylaxis by implementing a solution to the single-frame 2D *Human Pose Estimation* problem. For this, methods of *Deep Learning* are employed. Two approaches are examined, a *coarse-to-fine Convolutional Neural Network* for direct regression of joint coordinates and a *U-Net* for the derivation of probability distribution heatmaps.

We conclude that training our models on a combined dataset of the publicly available *Bodies at Rest* and *SLP* data yields the best results. Furthermore, various preprocessing techniques are investigated, and a *hyperparameter optimization* is performed to discover an improved model architecture. Another finding indicates that the heatmap-based approach outperforms direct regression. This model achieves a mean per-joint position error of 9.11 cm for the Bodies at Rest data and 7.43 cm for the SLP data. We find that it generalizes well on data from mattresses other than those seen during training but has difficulties detecting the arms correctly.

Additionally, we give a brief overview of the medical data annotation tool *annoto* we developed in the bachelor project and furthermore conclude that the *Scrum* framework and agile practices enhanced our development workflow.

Contents

1. Introduction

Despite advances in medical care, bed-bound patients are still a common victim of *pressure ulcers*, i.e., skin and tissue damage caused by constant pressure that may extend down to the bone or muscle [13, 82]. These sores considerably impair quality of life and can ultimately lead to death [13]. Yet often, the most serious consequences would be easily preventable with regular movements and repeated, thorough examinations of the patient [27, 82]. Overburdened or inexperienced nurses, including ambulatory or family caregivers, often do not have the time, capacity, or knowledge to manage these extensive and responsible tasks appropriately [4, 47, 60].

This is where the *WiseMat* project[1], a cooperation between our project partner, the *GETEMED Medical and Information Technology AG*[2], and *Softline Schaum Storkow*[3], comes into play: With the help of a mattress with integrated pressure sensors, body areas with an increased risk of developing pressure ulcers should be identified. Based on this information, the nursing staff could then examine and treat the affected patients in a targeted manner, reducing the workload and mitigating the threat of pressure ulcers.

Such a risk assessment could, among others, be based on exceeding certain thresholds of duration since the last change of the patient's posture or of cumulative pressure applied to a body part over time. To ensure data protection and privacy, this evaluation should ideally take place on an embedded system on the edge of the mattress so that only less sensitive warnings and alerts need ever be transmitted. A continuously running, automated system like the one proposed can monitor the patient's health status in a way human caretakers never could [24]. The scope of application includes hospitals (specifically intensive care units) and ambulatory care but can be easily adapted to similar use cases like decubitus prophylaxis for people who use wheelchairs [7].

This work aims to provide a basis for evaluation procedures like the one previously introduced by presenting a method to estimate the patient's pose, i.e., the 2D positions of joints and limbs, from single-frame pressure mattress recordings. For that, we employ, adapt, and compare two approaches (direct regression and heatmap-based regression) based on well-known *Deep Learning* architectures (*Convolutional Neural Network* and *U-Net* [68]). We train and evaluate our models on openly available datasets as well as data provided by the project partner.

[1]*WiseMat*. URL: `https://www.getemed.de/en/getemed/research/wisemat`.
[2]*GETEMED*. URL: `https://www.getemed.de/en/home`.
[3]*Softline Schaum Storkow*. URL: `https://www.softline-schaum.de/`.

This thesis is structured as follows: In chapter 2, we introduce the medical and technical background, i.e., pressure ulcers, the WiseMat project, the *Human Pose Estimation* problem, data procurement, and *Machine Learning*. Next, chapter 3 presents related applications and work. We introduce *annoto* in chapter 4, a data labeling tool we developed during the bachelor project. This section focuses on the versatile requirements, their implementation, and our agile software development process with the Scrum framework. After this, we show our solution to the Pose Estimation problem in chapter 5, that is, the metrics, datasets, and models used as well as experiments performed, and summarize, compare, and analyze our results in chapter 6. Finally, chapter 7 concludes and discusses this thesis and identifies opportunities for future work.

2. Background

This chapter starts with an overview of the nursing background, mainly a definition of *pressure ulcers* and a quick look at the current best practices for its prevention, followed by an introduction to the *WiseMat* project. After this, we continue with a brief overview of the technical background, consisting of the *Machine Learning* and *Deep Learning* methods our work is based upon, and an introduction to the *Human Pose Estimation* problem.

2.1. Pressure ulcers

The term ***p**ressure **u**lcers* (PUs), also known as *pressure sores*, *bedsores*, or *decubitus ulcers*, refers to localized areas of skin or tissue damage that develop because of pressure over a bony prominence [13, 82]. PUs affect not only bed-bound, stationary inpatients but are also a major risk for ambulant outpatients [82].

The main cause of PUs is pressure on bony protrusions, often in connection with shear forces and the resulting friction [32]. When the patient is in the supine position, the lateral areas of the heel and sacral area are most vulnerable [79]. Properties mentioned in the context of major risk factors include low mobility or physical activity, urinary and fecal incontinence or other problems concerning body hygiene, poor positioning techniques, medication, decreased consciousness, and advanced age [13, 32, 35, 74]. Additional promoting circumstances cover dehydration, malnutrition, and metabolic disorders since these impact blood circulation [32]. Recently, microclimatic conditions on the body surface have received increasing attention in PU prevention research [17, 79]. This concerns factors like temperature, skin moisture, and airflow. Sex, however, is not considered an independent risk factor, nor are different prevention strategies required [50].

Possible consequences include pain, reduced patient autonomy, increased risk of infection and sepsis, osteomyelitis, conduction of additional surgical procedures, long periods of hospital stay, significant physical-social and self-care dysfunction, depression, and death [13]. PUs are the third most costly disease after cancer and cardiovascular diseases, making them the nursing disease that is most expensive to treat [13]. Providing care for one ulcer may already cost between $\$3,500$ and $\$60,000$ [77]. In the US alone, about 2.5 million hospitalizations are due to PUs, costing an estimated $6.4 billion per year - That is 1.2% of total health care costs in the US [44, 64].

The state of development of a pressure ulcer is primarily defined by its depth and often assigned to one of four categories [82]. These differ in the body parts affected by the ulcer (that is, outer skin down to the muscle or bone, see Figure 2.1) and the resulting identifying characteristics as well as its danger to the patient's condition. If preventive measures are not taken, the ulcer progresses to the next, more risky stage. Ulcers may be *unstageable*, for instance, if the bottom is not visible due to thick eschar. In this case, the PU should be treated as if it were in the most dangerous stage four [27]. It is also typical that an ulcer follows an inside-out pathway, starting in the deeper, soft tissue under still intact skin [79]. These ulcers are referred to as *deep tissue injury*. Figure A.1 visualizes characteristic PUs of all four stages, an unstageable ulcer, and an example of deep tissue injury. Please note that these images are rather displeasing.

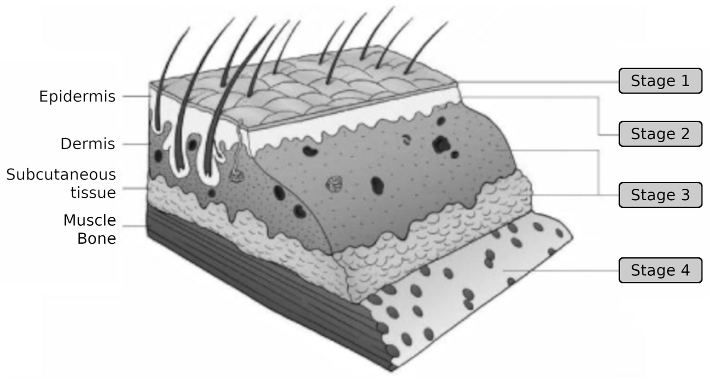


Figure 2.1.: Visualization of *pressure ulcer* stages in layers of the human skin [82]. Stage one ulcers affect only the outer skin; The underlying tissue is unaffected. In stage two, the ulcer reaches the epidermal layer and may extend into the dermis. A stage three ulcer extends into the subcutaneous tissue; Underlying bone, tendon, or muscle may be visible. When the ulcer extends into the bone, tendon, or muscle, it has reached stage four.

The treatment of PUs includes palliative measures that provide adequate pain relief as well as curative measures like the elimination of pressure on the affected areas, removal of necrotic and devitalized tissue, preservation of the integrity of the surrounding healthy skin and tissue, reduction of bacterial load, facilitation of the body's mechanism of natural healing, and treatment of risky conditions like malnutrition and anemia [82]. The European Pressure Ulcer Advisory Panel and National Pressure Ulcer Advisory Panel published a joint report on guidelines for prevention and treatment of pressure ulcers [27].

Besides repeated skin assessments that determine whether PUs are present, clinicians should also regularly re-evaluate each patient's risk of developing ulcers [82]. The most common tool for this purpose is the *Braden score*, which sums individual risk assessment scores in the categories of sensory, moisture, activity, mobility, nutrition, friction, and shear [82].

The main preventive measure concerns the regular turning of patients with mobility problems. The frequency depends on several factors, including individual tissue tolerance, level of activity and mobility, general medical condition, overall treatment objectives, and assessment of individual skin condition [27, 82]. Clinicians should lift and reposition patients rather than slide them across the mattress because friction and shear forces can lead to a disruption of capillaries and small vessels, promoting the formation of PUs [82].

As early as 1859, Florence Nightingale realized: "If [the patient] has a bed-sore, it is generally the fault not of the disease, but of the nursing." [59] This still applies today: Overburdened or inexperienced nurses, including ambulatory or family caregivers, often do not have the time, capacity, or knowledge to appropriately identify and treat PUs [4, 47, 60]. Thus, in addition to best practices for clinicians, there are research and commercial solutions that aim to support the work of nurses using pressure load measurements, accelerometers, or moisture sensors. Gallinger et al. present *DekuProSys*, a platform that captures and reports risk factors, assists in decision-making and documentation, and provides care information and instructions [32]. The *Mobility Monitor* of the company *Compliant Concepts* detects movements using a pressure sensor and warns clinicians if inactivity lasts too long [72]. The company *Leaf Healthcare* offers wearable sensors to monitor movement [86].

Ongoing research also covers surfaces and materials. As described by Tomova-Simitchieva et al., the *skin microclimate*, i.e., temperature, humidity and airflow, is affected by the support surface, cover, bedsheet, and clothing materials [79]. The authors compare several factors that promote the development of PUs for three mattresses (made of gel, air, and foam) and conclude that the gel and air mattresses are more protective. In general, a distinction is made between *active mattresses*, which require electricity to alternate pressure, and *reactive mattresses*, which adjust their load distribution in response to the load applied [79].

2.2. WiseMat

We saw that pressure ulcers are a major health threat for many patients. The risk of developing PUs is different for every person, and so are the actions nurses have to take. At the same time, too much unnecessary movement can even facilitate the formation. The core goal of the *WiseMat* project is to mitigate the threat of these decubitus ulcers. To this end, it aims to support clinicians in their work, thus saving them valuable time, preventing patient suffering, and reducing healthcare system costs. This section, co-authored with Berndt [12], briefly introduces the project.

The WiseMat project, which was developed as part of the state of Brandenburg's competition *digital health for a better life*, is part of a collaboration between the companies *GETEMED Medical and Information Technology AG* and *Softline Schaum Storkow*, funded by the *Ministry for Economic and European affairs of the State of Brandenburg*, and aims to build

1. A pressure-sensing mattress that is robust, precise, and comfortable enough to be used in hospitals and telemedicine, and

2. Software components that preprocess, analyze, and evaluate the data recorded from the latter with the goal of issuing warnings before pressure ulcers can form.

This thesis is in the area of the second objective. Ideally, the software would run as an embedded system on the edge of the mattress. The thesis of Bauknecht [10] describes our project's work in this field, studying the feasibility of Machine Learning in embedded systems. In such a setup, instead of continuous streams of sensitive raw pressure data, only warnings and recommendations would ever leave the device, supporting data protection and privacy assurance.

At the time of writing, the mattress hardware is still in a prototypical state. According to GETEMED, many issues have to be cleared out before a first field test can be started, such as mechanical dependencies and the resulting geometry shifts, tolerance of the foam materials used, as well as nonlinearities and other inaccuracies of the sensors employed. The project partners aim for the final product's price to be between €1,500 and €2,000. A size of $200 \times 88 \times 16$ centimeters is targeted with 44×16 pressure sensors evenly distributed and integrated into the upholstery of the mattress. Each sensor reports 10-bit values linearized from slightly more precise raw measurements.

Because the WiseMat hardware was not yet available during our project, we fell back on targeting our work to a pressure mattress that was already present at Softline. It has a resolution of 64×26 pressure elements, each measuring fixed point numbers between 0 and 128 with one decimal place[1].

2.3. Human Pose Estimation

Human Pose Estimation is defined as "the problem of localization of human joints (also known as key points - elbows, wrists, etc.) in images or videos" [8]. These key points are often visualized in the form of a skeletal representation where anatomically adjacent joints are connected, like the one you can for instance see in Figure 5.1.

The problem of Pose Estimation has been analyzed since the seventies: In 1973, Fischler et al. proposed an algorithm to match pictorial structures, for example, human key points like elbows or wrists. They introduce a descriptive scheme and a metric their algorithm minimizes using dynamic programming in order to find matches [30].

Pose Estimation can be based on a variety of different criteria. For many common cases, two-dimensional joint coordinates are sufficient, but some applications, like assistive robotics, require precise locations in 3D space [8]. In addition, images may be constrained to single persons, as is the case for the pressure mattress data, or

[1] *XSensor X3 Specification Sheet.* URL: http://www.nbn.at/fileadmin/user_upload/Vertretungen/XSENSOR/Products/PDF/XSENSOR-X3-DISPLAY-Medical-Mattress-System-1.pdf (visited on 2022-07-17).

allowed to contain multiple. Thus, Pose Estimation algorithms usually follow a top-down or a bottom-up approach [61]: A top-down algorithm starts by identifying single persons and continuously refines to skeletal joint positions. On the other hand, a bottom-up approach first identifies individual key point proposals and continuously puts those into the context of the bigger picture.

Another essential factor is the consideration of temporal consistency, in other words, whether we operate on single- or multi-frame data [6]. Many applications are only possible if the skeleton can be derived in real-time [22]. Thus, execution speed may be a critical criterion as well.

Just as importantly, we need to make sure that we deal appropriately with uncertainty [22]: Body parts may be missing in the image because they were hidden behind another object or lifted from the pressure-sensitive mattress. The input data is under-constrained; Thus, several equally plausible interpretations may exist. In such cases, we could either try to perform an educated guess or communicate that no prediction was possible. Depending on the application, one or the other may be desirable.

2.4. Machine Learning

Whereas classical algorithms describe data transformations thought of by humans, ***M**achine **L**earning* (ML) derives transformations or new insights from data. ML algorithms perform these derivations. The resulting algorithm is often referred to as a *model*. Using additional preprocessing steps for data cleaning, normalization, transformation, or feature extraction and selection can help machine learning algorithms and models to achieve better performance [43].

In literature, a distinction is made between *supervised* ML, where labeled data supervise the ML algorithm in deriving a model, and *unsupervised* ML, where some inherent structure is discovered in unlabeled data [88]. In supervised learning, we may, for instance, want to compare our input data to all the labeled data we know and output the label of the closest such data point: This rather basic procedure is well-known as the *1-nearest neighbors algorithm* [23]. A basic example of unsupervised learning is *data clustering* [53].

2.4.1. Evaluation

Given a set of data points $\mathbf{X}$ with well-known labels $\mathbf{Y}$ and a ML model defining a function $m()$, we can use *loss metrics* to objectively evaluate the quality of our model's predictions $\hat{\mathbf{Y}} = m(\mathbf{X})$. One common and simple example is the ***M**ean **S**quared **E**rror* (MSE), also known as $||\cdot||_2$ or ℓ_2-norm, shown in Equation 2.1 [88].

$$\text{MSE} = \frac{1}{k}\sum_{i=0}^{k}(\mathbf{Y}_i - \hat{\mathbf{Y}}_i)^{\mathrm{T}}(\mathbf{Y}_i - \hat{\mathbf{Y}}_i) \tag{2.1}$$

It would be misleading to assume that the loss determined on our data is a good indicator of the quality of our network: In the worst case, it could simply memorize all the samples it saw and return incorrect values for every new point instead of adapting to the actual underlying function. This phenomenon of ML models performing better on our dataset than on unseen data is called *overfitting* [41]. In order to mitigate such misjudgment, we usually randomly shuffle and split all the labeled data we have into a *training set* $\mathbf{X_{train}}, \mathbf{Y_{train}}$ and a *test set* $\mathbf{X_{test}}, \mathbf{Y_{test}}$: Whereas the training set is used to adapt the model, the test set is only used for evaluation purposes [88].

The method of *k-fold cross-validation* is strongly related [41]. Here, the dataset is split into k disjoint sets of roughly equal size. For each of the k sets, we train our model on the other $k-1$ sets and evaluate it on the remaining one. To summarize the model's overall quality, the mean of the individual runs is often used [41]. Setting k to the total number of dataset entries is called *leave-one-out cross-validation* [14]. Choosing k decides the trade-off between low bias and correspondingly high confidence (higher k) and few training runs and correspondingly high speed (lower k) [14]. K-fold cross-validation is especially useful when data is scarce, whereas train-test-split is sufficient when more data are available than are necessary for a sufficient training [88].

We usually experiment with several models and see which one fits best, choosing the one with the lowest test loss. However, in doing so, we can once more no longer make an objective statement about our model's quality: We may have decided upon the one that coincidentally fits our test dataset best, not the one that best matches the true underlying function. Once more, we solve this problem by splitting our test data into a test set $\mathbf{X_{test}}, \mathbf{y_{test}}$ and a *validation set* $\mathbf{X_{val}}, \mathbf{y_{val}}$: Whereas the validation set is used to evaluate the model during model experimentation, only the chosen model will ever see the test data for the final evaluation [88].

Often, our datasets include multiple samples recorded from different subjects. Since a *group leakage*, i.e., a distribution of samples recorded for one subject across training, validation, or test data, can lead to the model overfitting on subjects and thus make us assume an exaggerated model performance, we should split our data by subjects instead of purely randomly [63, 85]. It is also vital to ensure that the split datasets remain representative, i.e., that proportions from the original dataset will be preserved, because an imbalanced dataset can lead to misjudgments [85].

2.4.2. Deep Learning

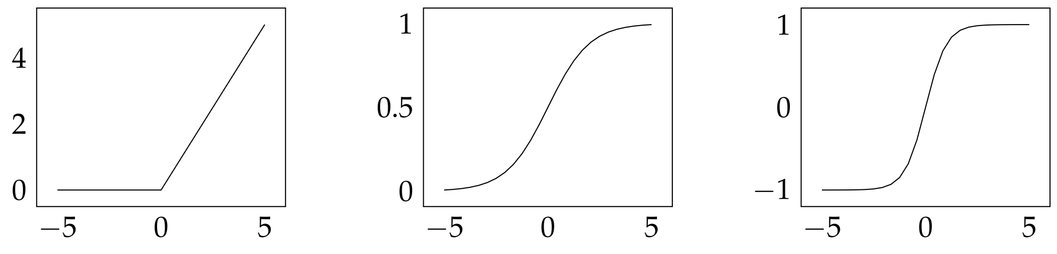
*A**rtifical **N**eural **N**etworks* (ANNs) are one of the structures that ML addresses. They refer to a mathematical model originally designed after neurons in the brains of animals. You often find visualizations of ANNs in the form of weighted, directed graphs (Figure 2.3a). In practice, many applications get along well without cycles: The so-called ***F**eedforward **N**eural **N**etworks* (FNNs) are ANNs restricted to *Directed Acyclic Graphs* [88].

An artificial neuron, the core building block of an ANN, is a node that takes a set of input variables $\mathbf{x} = (x_1, \ldots, x_n)$ and transforms them into a single output y.

As can be seen in Equation 2.2, this is done by calculating a linear combination of **x** and subsequently applying a nonlinear *activation function* φ [88].

$$y = \varphi(x_1 w_1 + \cdots + x_n w_n + b) = \varphi(\mathbf{x} \cdot \mathbf{w} + b) \tag{2.2}$$

Commonly used activation functions comprise the ones shown in Figure 2.2, i.e., ReLU, sigmoid and tanh. We find that ReLU is popular because it achieves good results and mitigates the vanishing gradient problem [88].



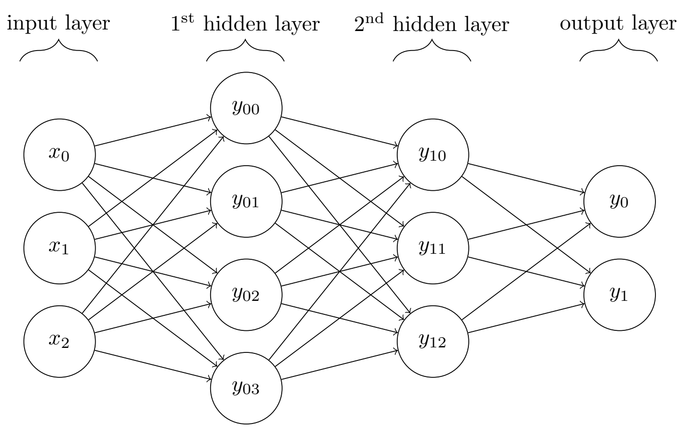
(a) $\mathrm{ReLU}(x) = \max(x, 0)$ **(b)** $\mathrm{sigmoid}(x) = \frac{1}{1+e^{-x}}$ **(c)** $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

Figure 2.2.: Visualizations of common *activation functions*. Whereas the last two are derivable, ReLU is highly popular because it mitigates the vanishing gradient problem and achieves good results [88].

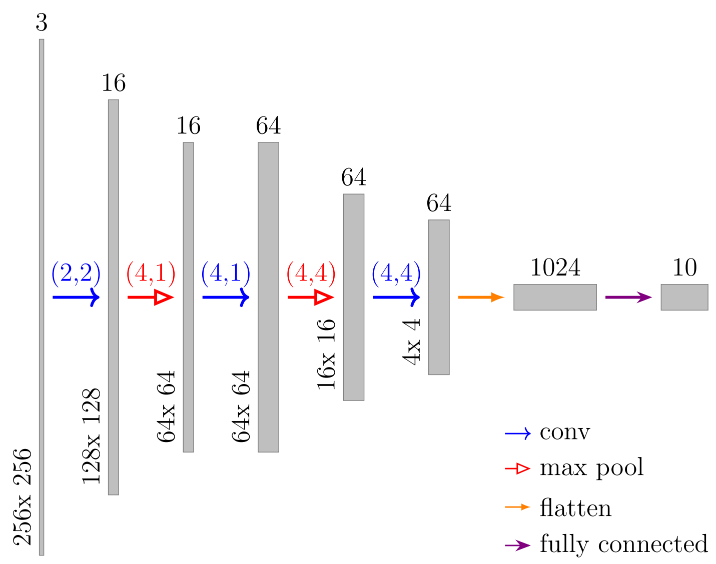
A set of interconnected neurons form an ANN. Often, they are additionally strictly organized in separate consecutive *layers*. Usually, a set of input variables $\mathbf{x} = (x_1, \ldots, x_n)$ are connected to the first (input) layer and a set of output variables $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_m)$ are connected to the final (output) layer [88]. Various architectures have been proposed for the intermediate layers (inner layers), e.g. the *Fully-Connected* and *Convolutional Neuronal Network* (CNN) patterns described in more detail in Figure 2.3.

Given such a network, we can calculate a prediction $\hat{\mathbf{y}}$ for an input **x** layer by layer: $\hat{\mathbf{y}} = \mathrm{net}(\mathbf{x})$. The arbitrary-width universal approximation theorem proven by Hornik in 1990 states that an FNN with as little as one hidden layer and a finite number of neurons is sufficient to approximate any continuous function with arbitrary precision [37]. Furthermore, it was shown empirically that deeper networks tend to perform better than shallower networks with the same number of neurons [65]. Thus, a large part of research in the field of ANNs concentrates on such deep networks with several hidden layers.

The weights **w** and the bias b of the various neurons determine the function described by the network and thus highly impact the quality of results. However, choosing appropriate weights is a complex matter. One of the most common heuristic procedures uses known data points $\mathbf{X_{train}}, \mathbf{Y_{train}}$ to *train* (that is, iteratively improve) the network to reduce a loss metric and essentially works as follows [88]: Start by randomly initializing all weights. Now, in each iteration, calculate the predictions for a subset $\mathcal{B} \subset \mathbf{X_{train}}$ (a so-called *minibatch*) and store the gradients.



(a) Exemplary *Fully-Connected Neural Network* with two hidden layers. Each neuron in layer i is connected to every neuron in layer $i + 1$. The value of one neuron is calculated as the *activation function* applied to the weighted sum of its predecessor's values. For instance, when using the ReLU activation function, $y_{10} = \mathrm{ReLU}(w_{y_{00}\rightarrow y_{10}} \cdot y_{00} + w_{y_{01}\rightarrow y_{10}} \cdot y_{01} + w_{y_{02}\rightarrow y_{10}} \cdot y_{02} + w_{y_{03}\rightarrow y_{10}} \cdot y_{03})$.



(b) Exemplary *Convolutional Neural Network* (CNN) for image processing. In a convolutional layer (blue), each neuron is connected to only a few spatially close neurons of the previous layer. Being a specialization of the *fully-connected* pattern with many weights set to zero, this approach reduces training effort. We call the set of incoming weights of one neuron a *kernel*. Each convolutional layer consists of various kernels of one fixed size. Convolution is applied across all input channels, and by using parallel kernels, we can generate multiple output channels. The spatial step between two kernels is called stride. In the figure, (kernel size, stride) is above the arrows. It is a common pattern to combine convolution with *pooling* layers (red), which are very similar except that convolutional filters have learnable weights, while pooling filters always calculate a fixed function (e.g., the maximum). In a *coarse-to-fine* CNN architecture, alternating convolutional and pooling layers are used to reduce spatial resolution while increasing dimensionality (the number of channels).

Figure 2.3.: Visualization of two common *Feedforward Neural Network* (FNN) architectures. The value of each neuron in an inner layer i is determined by neurons solely from the previous layer $i - 1$ [88]. It is not uncommon to combine these architectures. For instance, in (b), although the network follows a convolutional architecture, the final layer is fully-connected.

Then calculate the loss and use *backpropagation* [69] together with optimization algorithms like ***S**tochastic **G**radient **D**escent* [67] (SGD) or *Adam* [42] to adapt the weights and biases of all neurons slightly, step by step from the final to the first layer. We can tune this process by parameters like *batch size* and *learning rate*. One *training epoch* refers to repeating iterations until all data points from $\mathbf{X}_{\mathbf{train}}, \mathbf{Y}_{\mathbf{train}}$ have once passed the network. We usually run multiple training epochs on our networks.

Because of their high-dimensional hypothesis space, ANNs often tend to overfit [41]. In order to mitigate these effects, we can procure more training data or artificially *augment* the dataset, add *regularization* terms to the loss metric, or apply *dropout*. The core idea behind the last two points is the same: No single neuron should become too important. A single connection with disproportionately high weight is a good indicator of overfitting and should thus be prevented. Instead of one omniscient neuron, we want the knowledge to spread across multiple neurons. For this, regularization suggests adding the absolute sum of all weights (L1-regularization) or the sum of the squares of all weights (L2-regularization), multiplied by a weighting factor, to the loss [88]. In doing so, networks with lower weights will be preferred during training. Dropout takes a different approach to achieve the same goal: In each training step, we randomly disable a small proportion of the neurons in one layer, and because each neuron will be occasionally unavailable, no single neuron can become too important [57]. However, both techniques can slow down convergence [46].

We summarize all parameters defined before the training process begins under the term *hyperparameters*. These include, among many others, the learning rate, optimization algorithm, number of epochs or criteria for early stopping, whether to use dropout, and kernel sizes. Deciding on a particular network architecture can also be seen as a first step in defining a number of parameters. The choice of hyperparameters can substantially impact the quality of the trained model. Therefore, a large part of the development of deep learning models is devoted to experimentation with hyperparameter values [3]. This concerns manual experiments as well as automated optimization.

2.5. Data Procurement

We saw that supervised ML algorithms require sample data points, in the case of Human Pose Estimation pressure recordings and labels in the form of skeletal annotations, from which to extract patterns. This section gives an overview of openly available datasets that can be used as a basis for implementing a solution to the Human Pose Estimation problem and presents means to collect and label new, custom data.

Several openly available datasets that include pressure mattress recordings exist; Some even include skeleton annotations. A first candidate is the *hrl-ros* dataset [22]: With 67, 177 samples of 18 patients, it is sufficient for many procedures. A mattress with a resolution of 64×27 sensors was used, which is very similar to our target hardware. However, many samples are oversaturated, and the skeletal annotations comprise only ten joints, which is just enough to get a rough idea of the patient's pose but not very detailed. Furthermore, many samples are pretty similar since whole motion sequences were recorded. On the other hand, this makes the dataset well suited to train a model that considers temporal consistency.

The *SLP* dataset [52] contains 13,770 single samples of 102 patients in various prescribed postures. In addition to pressure mattress recordings and skeletal annotations, the dataset provides camera, infrared, and depth images. With 14 joints, the skeletal annotations provide a sufficient level of detail, and their mattress has a very high resolution of 192×84 sensors.

Many datasets only provide labels for the posture, i.e., supine, prone, etc. This includes the data provided by Clever et al. [18] as well as *pyhsionet's Pressure Map Dataset* [9]. Although their recordings are of very high quality, we cannot readily benefit from their work as the skeleton annotation process is considerably time-consuming.

On the other hand, custom data gives control over the representativeness of the dataset (for example, range of body types and variance of postures) and can assist the model in learning hardware specifics (like its resolution and pressure sensitivity distribution). A common approach to generating a new dataset is to first record the data and then manually label it. However, this process is time-consuming and may be error-prone, depending on the labeler's experience and attention. Therefore, Clever et al. propose using motion capture to annotate data on the fly [22]. As motion capture equipment is expensive, with the cheapest *OptiTrack*[2] system offered at the time of writing starting at \$9,745, another approach worth considering might be to utilize a well-known Pose Estimation algorithm trained on camera images to generate labels from video recorded in parallel with the pressure measurements. Since these models are usually not optimized for lying postures, we are uncertain about the quality to be expected. It should be expected that at least some manual adjustments will still be necessary.

A completely different approach concerns the use of synthetic data. Clever et al. used a physics simulation to generate the large *Bodies at Rest* database with a wide range of body types and highly varying postures [19]. This dataset comprises a total of 206,000 samples and is the largest of the ones presented in this section. Their mattress resolution of 64×27 is also very close to our target hardware, and it is in the nature of the approach that the images are clear and noise-free.

Finally, *data augmentation* could be applied to provide the big datasets many machine learning algorithms require in order to avoid overfitting [73]. This includes simple content preserving transformations like mirroring, translating, and adding noise. However, some popular transformations like scaling could be problematic for pressure mattress data since the pressure distribution does not grow linearly with the human's size [22].

[2] *OptiTrack*. URL: `https://optitrack.com/systems` (visited on 2022-07-18).

3. Related Work

In this chapter, we will first look at applications that could be enhanced or improved by using pressure-sensing mattresses for Human Pose Estimation. The rest of this chapter gives a rough overview of techniques that have been explored previously, either from our own domain or from related fields.

3.1. Applications

The Oxford Dictionary defines *human posture* as "the position and carriage of the limbs or the body as a whole" [66]. Thus, *Human Posture Classification* is defined as the problem of mapping images or videos to a set of well-known body positions (like supine, prone, etc.). Although not part of this work, the problem is closely related and has been investigated by Berndt [12] of our bachelor project team.

Several applications based on Human Pose Estimation from pressure data have been explored previously. Firstly, an estimated pose can be used as the basis for some highly precise and efficient Posture Classification algorithms, as shown by LeViet et al. [48]. The work of Achilles et al. proposes a method for detecting epileptic seizures [2]. Metsis et al. recognize and evaluate sleep patterns [56]. Clever et al. derive a 3D human model for control of an assistive robot [20, 22]. And Grimm et al. use human pose for the automatic setup of CT scanners [34].

3.2. Human Pose Estimation from pressure images

For Human Pose Estimation from data recorded using pressure-sensing mattresses, Liu et al. propose a classically algorithmic procedure that uses the *pictorial structures model* [51]. This algorithm does not derive skeletal joints but only certain body parts and may thus not be considered a real solution to the Pose Estimation problem. Farshbaf et al. explore a method that uses the *k-nearest neighbors* algorithm on the output of a *Principal Components Analysis* for posture classification and then labels limbs accordingly [28]. This algorithm will, however, not work for new or uncommon postures. The algorithm of Casas et al. uses hashing to accelerate a k-nearest neighbors search, predicting the average of the most similar known datapoint's joints [16]. Additionally, they implement a 7-layer coarse-to-fine CNN that directly regresses joint coordinates.

Clever et al. use a 6-layer CNN to compare direct regression of joint coordinates with regression of a prameterization of a *kinematics body model* that ensures certain anatomical constraints and from which the actual joint positions can then be calculated [22]. In addition, they use multiple forward passes with dropout enabled to obtain a confidence score for each joint. In their subsequent work from 2020, they refine their approach to regress a parameterization of the more advanced ***S**kinned **M**ulti-**P**erson **L**inear **M**odel* (SMPL) [20]. Using *representation learning*, Davoodnia et al. train an *hourglass model* that converts pressure recordings into images that can be understood by standard pose estimators designed for RGB camera images [26].

3.3. Human Pose Estimation from other data sources

Whereas Pose Estimation from pressure recordings is a niche area in active research, many approaches have been implemented and evaluated for the more familiar RGB or depth camera images. We can strongly benefit from their findings and well-established techniques.

The classical approach to Human Pose Estimation is the pictorial structures model [30], introducing descriptive schemes and metrics the algorithm minimizes using dynamic programming in order to find matches. Other common approaches include Hierarchical and Non-Tree models. *Hierarchical models* [76] recursively represent objects as collections of parts at different levels of detail, where parts at each level are connected to the respective coarser level by a parent-child relationship. *Non-Tree models* [25] employ nonlinear, multi-layered joint regressors which predict joint locations by modeling interdependence and co-occurrence of parts, thus circumventing typical ambiguities.

DeepPose was one of the first papers to apply Deep Learning to the Pose Estimation problem by training a CNN-based model to directly regress Cartesian joint coordinates [81]. Additionally, they propose to incrementally evaluate the model on a reduced region of interest depending on the prediction for the previous one.

Tompson et al. suggest proceeding in two stages: First, use a CNN to generate coarse belief belief heatmaps for each joint's position from a gaussian pyramid of the input image, and then use a *Siamese network* to refine the heatmaps based on a cropped portion of the input image [80]. Cartesian joint coordinates can be derived from these heatmaps via the argmax function.

Convolutional Pose Machines use a CNN to predict belief values for each joint for a single pixel of the image and obtain full belief maps by sliding this network across the image [84]. In order to refine this first coarse estimate, additional stages can be added: For stage $t > 1$, a CNN processes not only the input image itself but also the belief map produced in stage $t - 1$.

Ronneberger et al. presented the *U-Net* architecture, which consists of two stages: The first stage is once more a coarse-to-fine CNN that derives some high-level features. These are then input to the second stage, which in each of its layers upsamples its previous values and stacks them together with the outputs of the respective layer from the first stage. In each of these steps, some additional convolutions are applied. Finally, the output of the penultimate layer of the second stage is stacked on the original input image, and some final convolutions produce a transformed image. The *stacked hourglass network* suggested by Newell et al. consists of a series of hourglass networks [58]. The hourglass architecture is similar to that of the U-Net, and stacking multiple of these hourglasses together helps the network to generalize better.

Clever et al. developed a model that infers human body pose as well as pressure distribution images from depth camera images [21]. Building on their previous work [20], this procedure bypasses the need for expensive mattresses at the expense of privacy.

4. Annoto

During the second half of the bachelor project phase, *annoto*, a tool for annotating medical data was developed in addition to the work on the topics of the bachelor thesis. The requirements were given by GETEMED, our project partner and customer. The tool was designed and implemented in cooperation with Berndt [12], König [45], and Sauer [70]. Unlike the rest of the thesis, this chapter is in the different domain of Software Architecture, Requirements Engineering, and Social Processes. It was written in collaboration with the developers mentioned above and presents the motivation and requirements given for building this tool as well as some implementation details. The development team used the *Scrum* framework to structure their work; The experiences with it are covered in the final section of this chapter.

4.1. Motivation and Requirements

One decisive factor of ML applications is the amount and quality of data for training and validation. In order to enable supervised learning techniques, the data have to consist of pairs of samples and associated labels. However, especially for very specific tasks, labeled data are often rare. Therefore, a standardized annotation process for generating labeled data can help to improve the quality of machine learning applications.

User Requirements The annotation tool should be as simple as possible to access and use while only requiring domain knowledge for the specific labeling task. For this, the customer required the annotation tool to be a web-based application with an easily accessible manual that explains how to operate the different parts of the application. Furthermore, it should be possible to include multiple persons as annotators in order to divide the workload and allow for faster results.

Medical Requirements In order to authorize machine learning applications for medical purposes, detailed information about the data that will be utilized for training and testing has to be stored and submitted to the authorities.

The Guideline for AI for medical products by Johner et al., for instance, lists 60 requirements related to the topic of data management [40]. These requirements are further divided into the four categories *Data collection*, *Data annotation*, *Procedure for (pre-)processing of data*, and *Documentation and version control*. Most of these requirements apply equally to training, testing, and validation data. The high number of different requirements in this guideline emphasizes that data utilization for machine learning applications in medical contexts must be well documented, especially when a custom annotation process is used for retrieving labeled data.

Annotation Metadata To fulfill the requirements given by the project partner as well as those extracted from the guideline that apply to the annotation procedure, the team decides on certain metadata information that must be stored along with each annotation:

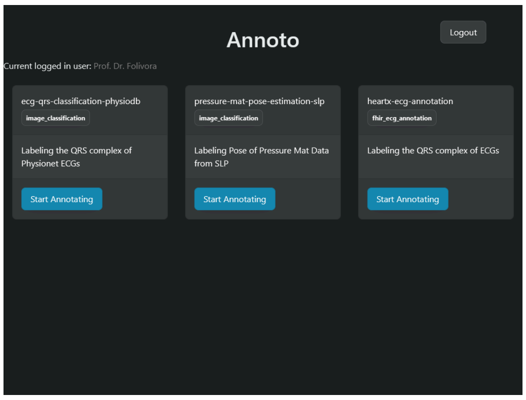
- A link to the data sample
- The timestamp of the creation of the annotation
- The name of the annotator
- A proof of competency of the annotator for the specific labeling task
- A confirmation that the annotator was attentive and capable of annotating a data sample at the time the annotation was created
- The actual label that is associated with the data sample

Requirements Checking While some of this metadata information, e.g., the actual label and the data sample link, can simply be retrieved and stored at the time of annotation creation, others require more sophisticated procedures, the implementations of which are further explained in section 4.2. Sauer [70] presents more details about our project's work on the confirmation of attentiveness.

4.2. The Tool

A login form is displayed to the user when opening the annotation tool in a web browser. Having signed in, an overview of the available labeling tasks is shown. This overview is structured as a grid of cards (Figure 4.1a). Each card contains the task title, a short description, and the type of the task. Task types may, among others, comprise classification of heartbeats, annotation of skeletons on pressure data, and measurement of *electrocardiograms* (ECGs). A task can be opened by clicking on the button on the respective card.

Every time users open a task, they are presented with a dialog (Figure 4.1b) asking them to confirm that they have read the manual and are attentive and capable of annotating the data. To ensure that annotators gives the required confirmation, the dialog is non-dismissible until the checkbox is selected. Once this precondition is met and the dialog is closed, a labelling view that is specific to the task is shown (figures 4.1c, 4.1d). More details about the implementation of different types of labeling tasks can be found in subsection 4.3.2.



Annoto

Logout

Current logged in user: Prof. Dr. Folivora

ecg-qrs-classification-physiodb

image_classification

Labeling the QRS complex of Physionet ECGs

Start Annotating

pressure-mat-pose-estimation-slp

image_classification

Labeling Pose of Pressure Mat Data from SLP

Start Annotating

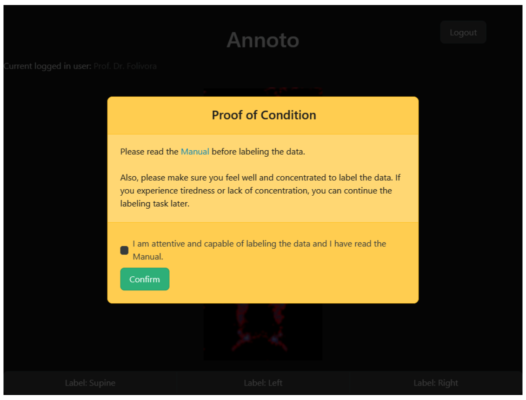
heartx-ecg-annotation

fhir_ecg_annotation

Labeling the QRS complex of ECGs

Start Annotating

(a) Overview of available tasks



Proof of Condition

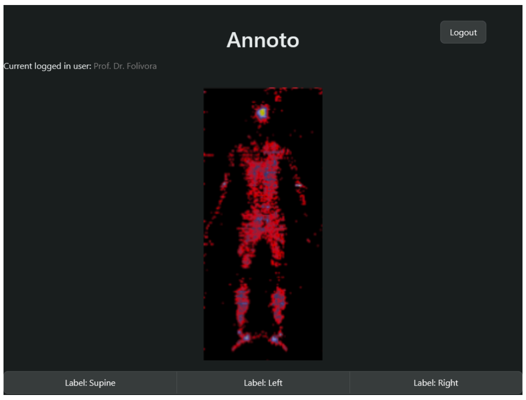
Please read the Manual before labeling the data.

Also, please make sure you feel well and concentrated to label the data. If you experience tiredness or lack of concentration, you can continue the labeling task later.

I am attentive and capable of labeling the data and I have read the Manual.

Confirm

(b) Dialog confirming attentiveness and training



Annoto

Logout

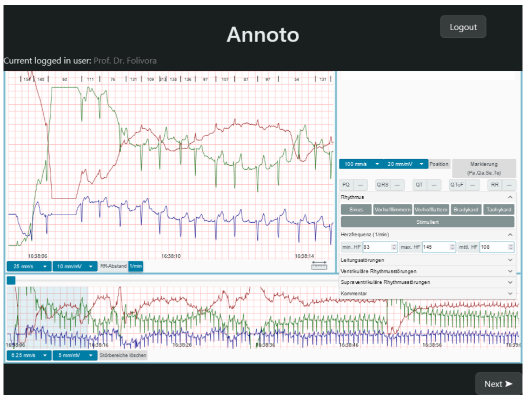
Current logged in user: Prof. Dr. Folivora

Label: Supine

Label: Left

Label: Right

(c) View for WiseMat posture classification



Annoto

Logout

Current logged in user: Prof. Dr. Folivora

Rhythmus
Herzfrequenz (1/min)
Leitungsstörungen
Ventrikuläre Rhythmusstörungen
Supraventrikuläre Rhythmusstörungen
Kommentar
Next

(d) View for measurement of ECGs

Figure 4.1.: Selection of important views of the *annoto* annotation tool. (a) When users log in, they are presented with an overview of available tasks. (b) Before the annotation process can start, users have to confirm their condition, i.e., attentiveness and training. The labeling view depends on the type of task, for instance, (c) picture classification or (d) ECG measurement.

4.3. Implementation Details

This section presents technical details of the implementation of the annotation tool. This includes the work on the application itself just as well as the technical processes we employ to ensure high stability, good code quality, and early feedback loops.

4.3.1. Tech Stack

For the front end, we use *VueJS*, a progressive *JavaScript* framework for building web user interfaces[1]. We utilize the default package manager *npm*[2]. To benefit from static typing, we employ *TypeScript* in conjunction with VueJS. We perform unit and end-to-end testing with *Cypress*[3], static code analysis with *eslint*[4], and opinionated code formatting with *Prettier*[5].

We also need a back end that manages the data to be annotated. It should likewise be capable of saving the annotations and administering different user accounts so the person that created an annotation can be identified. This back end is built using *FastAPI*[6], a modern Python web framework for writing *RESTful* [29] ***A**pplication **P**rogramming **I**nterfaces* (APIs). The API tests are implemented with *pytest*[7], and we also enable optional static type checking with *mypy*[8]. As a formatting standard, we agree on *black*[9], and use *pylint*[10] for static code analysis. We decide to employ an SQL-based relational database for storing data. Locally, this is handled by *SQLite*, a small, fast, self-contained, open-source SQL database[11], whereas *PostgreSQL*, an advanced, powerful, open-source database[12], is used within the production environment. Additionally, we set up a *Docker* container [55] for local development that creates an isolated environment, installs all dependencies, loads the test data, and launches the application's back and front end.

4.3.2. Plugin System

As mentioned in section 4.2, one core requirement for the application is the ability to support a wide range of labeling tasks. To fulfill this requirement while keeping the codebase easily readable, maintainable, and expandable, we decide to introduce a plugin system. Each task is assigned a task type that identifies the responsible plugin. The plugin is then used to load the data to be annotated, choose the annotation view for the front end, and save the annotations created.

[1]*Vue.js - The Progressive JavaScript Framework.* URL: `https://vuejs.org/` (visited on 2022-06-27).

[2]*npm.* URL: `https://www.npmjs.com/package/npm` (visited on 2022-06-27).

[3]*Cypress - JavaScript End to End Testing Framework.* URL: `https://www.cypress.io/` (visited on 2022-06-27).

[4]*ESLint - Pluggable JavaScript Linter - Find and fix problems in your JavaScript code.* URL: `https://eslint.org/` (visited on 2022-06-27).

[5]*Prettier - Opinionated Code Formatter.* URL: `https://prettier.io/index.html` (visited on 2022-06-27).

[6]*FastAPI.* URL: `https://fastapi.tiangolo.com/` (visited on 2022-06-27).

[7]H. Krekel, B. Oliveira, R. Pfannschmidt, F. Bruynooghe, B. Laugher, and F. Bruhin. *pytest 7.1.1.* 2004. URL: `https://github.com/pytest-dev/pytest` (visited on 2022-06-27).

[8]*mypy - Optional Static Typing for Python.* URL: `http://mypy-lang.org/` (visited on 2022-06-27).

[9]*Black - The Uncompromising Code Formatter.* URL: `https://github.com/psf/black` (visited on 2022-06-27).

[10]*Pylint 2.14.4.* URL: `https://github.com/PyCQA/pylint` (visited on 2022-07-08).

[11]*SQLite Home Page.* URL: `https://www.sqlite.org/index.html` (visited on 2022-06-27).

[12]PostgreSQL Global Development Group. *PostgreSQL.* URL: `https://www.postgresql.org/` (visited on 2022-06-27).

We implement two exemplary plugins. For classification of images, the *image_classification* plugin loads the pictures from a folder on disk according to the name of the task, implements a view that displays them in the front end along with buttons for the classes specified in the task description, and stores annotations next to the image file in JSON format. The *fhir_ecg_annotation* plugin loads IDs of ECGs that have not yet been annotated from a *FHIR*[13] server, embeds the project partner's *HeartX* viewer via an *iframe* in the front end, and, after saving, fetches the annotation made from the FHIR server to validate the presence of fields required by this task and to store a reference to the annotation locally.

4.3.3. Type safety across the client-server boundary

The code that connects the back end to the front end is generated automatically and can be continuously regenerated when the back end changes. This process is done in two steps: First, FastAPI provides our back end with the possibility to export a completely machine-readable self-documentation following the *OpenAPI* standard[14]. Based on this, *openapi-typescript-codegen*[15] generates TypeScript classes for the described objects, as well as properly typed and documented functions that call the respective REST endpoints.

The two steps were summarized in an npm script. Using the CI pipeline (see subsection 4.3.4) to automatically execute this script ensures that the API and the front end code calling it will hardly ever become incompatible.

4.3.4. Continuous Integration

***C**ontinuous **I**ntegration* (CI) "is a software development practice where members of a team integrate their work frequently [. . .] Each integration is verified by an automated build." [31] Because we already use *GitHub*[16] for version control and to host our codebase, we implement our automated builds using *GitHub Action workflows*[17]. We decide to run our CI pipeline on every push to a feature branch that we were planning to merge into the development or main branches. To ensure that invalid code never enters the latter, we make the CI checks a prerequisite for merging. In addition, at least one approving code review from a team member other than the author is required before any pull request can be merged.

For both the back and front end, we create workflows that build the application, i.e., set up the environment and install all dependencies, check syntactic correctness and type annotations, run the linter, execute all tests, and upload the coverage

[13] *FHIR v4.3.0*. URL: https://www.hl7.org/fhir/ (visited on 2022-06-27).
[14] SmartBear Software. *OpenAPI Specification - Version 3.0.3 | Swagger*. URL: https://swagger.io/specification/ (visited on 2022-06-27).
[15] *openapi-typescript-codegen*. npm. URL: https://www.npmjs.com/package/openapi-typescript-codegen (visited on 2022-06-24).
[16] *GitHub - Build software better, together*. URL: https://github.com (visited on 2022-07-01).
[17] *GitHub Actions Documentation*. GitHub Docs. URL: https://ghdocs-prod.azurewebsites.net/en/actions (visited on 2022-06-27).

report. Additionally, workflows for automatic formatting are added, which format the back and front end code with black and Prettier respectively, and which add the resulting changes, if they exist, as a new commit to the branch. Similarly, we implement a workflow that regenerates the front end's REST client and, should changes exist, adds a new commit for them.

4.3.5. Continuous Deployment

*Continuous **D**eployment* (CD) is a DevOps method that automates the process of releasing new versions of a software [15]. After the CI pipeline has passed successfully, CD automatically integrates the new software into the production environment without requiring manual tasks. We host our production environment on *Heroku*, a cloud platform that focuses on simplifying deploying, configuring, scaling, tuning, and managing web apps[18], providing Python and VueJS environments as well as a PostgreSQL database out of the box. In addition to that, it also makes CD very easy by connecting the GitHub repository to the Heroku app. We set up two environments for staging and development, serving the latest state of the main and development branches respectively.

4.4. Scrum

Finally, we want to briefly introduce the *Scrum* framework, a project management technique employed to manage our development process, and present our experiences with this technique.

4.4.1. Agile software development powered by Scrum

"Scrum is a lightweight framework that helps people, teams, and organizations generate value through adaptive solutions for complex problems" that is becoming increasingly popular in the industry, research, and other fields [71]. Having its roots in software development, it employs an iterative-incremental approach and is highly related to *extreme programming* and other *agile practices*.

Every Scrum team is self-organizing and consists of one *Product Owner*, one *Scrum Master*, and the *Developers*, together working towards achieving the *Product Goal*. The Scrum Guide establishes five *Scrum Events* that each offer the opportunity to inspect and adapt *Scrum Artifacts*. These events comprise *Sprint*, *Sprint Planning*, *Daily Scrum*, *Sprint Review*, and *Sprint Retrospective*. Note that the term Scrum artifact refers not only to increments but also to the Product and Sprint Backlogs. More details about the Scrum process are described in the Scrum Guide [71].

[18]*What is Heroku | Heroku*. URL: `https://www.heroku.com/what` (visited on 2022-06-27).

4.4.2. Personal experiences

We employed Scrum to manage the development process of the annoto tool. Thanks to GETEMED and our project supervisors, we had the opportunity to experience Scrum the way the project partner uses it in their real projects. Within three scrum sprints, each lasting three weeks, we incrementally improved not only our software but also our understanding of the problem domain and the development process itself. Due to its simplicity, we found it quick and efficient to implement Scrum.

We started each sprint with a planning meeting where user stories were defined, estimated, and selected together with the Product Owner, followed by internal team meetings where they were further refined into atomic development tasks. Subsequently, the sprint focused on the implementation of these tasks. Daily stand-up meetings allowed for a constant overview of the current progress and for solving impediments as soon as possible. Each sprint ended with a review meeting where the sprint results were presented and discussed with together GETEMED. This allowed us to generate new insights and receive suggestions for the upcoming sprints. Furthermore, each sprint was followed by a retrospective meeting focusing on improving the various processes, enabling us to reduce irritations, misunderstandings, and inefficiencies steadily.

Our team experienced that the Scrum framework enabled us to produce code of higher quality while reducing development time. *Timeboxing* the meetings made sure they did not become time wasters, while subsequent *follow-up meetings* allowed to discuss highly specific topics with the subset of people to whom they really matter.

Nevertheless, we also discovered some downsides. Most of the time, we found it challenging to estimate user stories well. In one planning meeting, for instance, we were presented four user stories, all of which we estimated to the exact same amount of story points, just to make them fit into our sprint budget. Additionally, we tended to overestimate stories because it was then easier for us to achieve our sprint goal. This was also related to the fact that the project partner implemented Scrum in such a way that additional *task refinement meetings* were held to elaborate new and existing user stories. These meetings could then also be used to add additional items to the Sprint Backlog, which does not strictly adhere to the Scrum Guide but has its own advantages and disadvantages. Furthermore, especially in the beginning, we had some severe timeboxing issues, which we were, however, able to solve by setting a dedicated *time keeper* for each meeting. We also found that the team's prior experience with Scrum plays an important role. Because two of our four developers, but most importantly also the Product Owner and the Scrum Master themselves, have applied Scrum for the first time, many things were a little more difficult. This, however, improved noticeably as the project progressed and people became more accustomed to their roles. Still, the project could have been a bit more productive with more initial experience. Overall, our negative experiences widely match the findings of many Scrum-critical articles [5, 33].

5. Methods

As stated in section 2.2, this work focuses on software components that process data recorded from pressure-sensing mattresses. More precisely, we want to develop a Deep Learning-based solution to the 2D Human Pose Estimation problem from single-frame pressure mattress recordings. This chapter provides a summary of the models used, experiments performed, and decisions made.

5.1. Technical details

This section gives a brief overview of the frameworks, libraries, and hardware on which we base our work.

Frameworks and Libraries *PyTorch* [62] is an open-source end-to-end machine learning framework that "enables fast, flexible experimentation and efficient production through a user-friendly programming front end, distributed training, and an ecosystem of tools and libraries". For our work, we use *PyTorch Lightning* [87], a framework that builds upon PyTorch. It provides a front end that reduces boilerplate, takes care of running code on any hardware, supports logging, profiling, and much more, and is thus well-suited for research.

Additionally, we integrate *TensorBoard* [1] to track, visualize and compare our metrics across train runs. *Optuna* [3] is an open-source optimization framework to automate hyperparameter searches that we employ to optimize our models. Similar to TensorBoard, we use the *Optuna Dashboard* to visualize and compare the current progress as well as the insights from individual trials.

Pandas [54, 78] data frames and *Numpy* [36] arrays are employed to store data and serialize it to disk. In order to visualize pressure images and skeletons as well as metrics and distributions, *Matplotlib* [38] and *Seaborn* [83] are utilized.

Hardware The experiments are run on a desktop computer with the specification given in Table 5.1. An exception to this is the hyperparameter optimization, which is executed on the HPI's *Data Engineering Lab*[1] (DE-Lab), an experimental platform for HPI scientists that provides a wide selection of powerful and modern computer architectures. Of the resources available at the time of writing, we use those listed in Table 5.2. The entry host *summon* is used to schedule and manage

[1]*HPI Data Engineering Lab*. URL: `https://hpi.de/forschung/hpi-data-engineering-lab.html` (visited on 2022-07-10).

jobs through *SLURM*[2]. The *NVIDIA Pyxis*[3] plugin allows for running containerized tasks, enabling us to base our work on the official Docker container for PyTorch Lightning[4].

Table 5.1.: Specification of the desktop computer used to perform all experiments except for the hyperparameter optimization

GPU	NVIDIA RTX 2070 SUPER (8 GB GDDR6)
Driver	NVIDIA CUDA Toolkit v11.7
CPU	AMD Ryzen 9 3900X (12 cores, 3.8 GHz)
RAM	2× 16 GB DDR4, 3200 MHz
SSD	Crucial MX500 (SATA)
Operating System	Windows 10

Table 5.2.: Resources the HPI's *Data Engineering Lab* provides access to at the time of writing

2× IBM AC922	1× NVIDIA DGX A100	1× HPE Apollo 6500
512 GB RAM	1 TB RAM	256 GB RAM
4× NVIDIA Tesla V100-SXM2	8× NVIDIA A100-SXM4-40GB	8× NVIDIA A100-SXM4-80GB
2× IBM POWER9 3.3 GHz (16 Cores)		

5.2. Metrics for Human Pose Estimation

The evaluation of a Human Pose Estimation algorithm is not an ordinary question of right or wrong, so we cannot use established metrics like accuracy or F1 score. Instead, we need a metric that defines how close the predicted skeleton is to the real one. In the following, we describe four metrics that are commonly used to assess the performance of a Pose Estimation model.

Percentage of Correct Parts

For the ***P**ercentage of **C**orrect **P**arts* (PCP), "a limb is considered correctly detected if the distance between the two predicted joint locations and the correct limb joint

[2]*Slurm Workload Manager - Overview*. URL: `https://slurm.schedmd.com/overview.html` (visited on 2022-06-15).

[3]*Pyxis*. URL: `https://github.com/NVIDIA/pyxis` (visited on 2022-06-15).

[4]*pytorchlightning/pytorch_lightning - Docker Image | Docker Hub*. URL: `https://hub.docker.com/r/pytorchlightning/pytorch_lightning` (visited on 2022-06-15).

locations is less than half of the limb length" [8]. We consider the left and right limbs upper arm, forearm, upper leg, and lower leg.

Percentage of Correct Key-points
For the ***P**ercentage of **C**orrect **K**ey-points* (PCK), "a detected joint is considered correct if the distance between the predicted and the true joint is within a certain threshold" [8]. This threshold is usually chosen relative to the head bone link or torso diameter. In our work, we use PCK@0.2, meaning we set the threshold to 20% of the torso diameter.

Mean Per Joint Position Error
The ***M**ean **P**er **J**oint **P**osition **E**rror* (MPJPE) describes the average Euclidean distance in centimeters between all actually visible and predicted joints [39]. It is calculated as shown in Equation 5.1, where p_i are the positions of detected joints, t_i the corresponding ground truths, and v_i the corresponding ground truth's visibilities.

$$\text{MPJPE} = \frac{\sum_i ||p_i - t_i||_2 \cdot v_i}{\sum_i v_i} \tag{5.1}$$

In our MPJPE calculation, we assume that all joints are always visible, i.e., set all visibilities to one. Additionally, we refer to the MPJPE with the arms omitted, i.e., that leaves out the joints for elbows and wrists, as MPJPE_{ao}.

Object Keypoint Similarity
The idea behind ***O**bject **K**eypoint **S**imilarity* (OKS) is similar to that behind MPJPE: We want to add up the similarities of corresponding detected and ground truth joints [75]. As shown in Equation 5.2 and in contrast to MPJPE, OKS weighs each joint's distance according to specific so-called *fall-off constants* given in Figure 5.1. Additionally, it includes the Euclidean distance exponentially.

$$\text{OKS} = \frac{\sum_i \exp(-\frac{d_i^2}{2s^2k_i^2})(1 - v_i)}{\sum_i 1 - v_i} \tag{5.2}$$

In this equation, d_i represents the Euclidean distances between the detected key points and the corresponding ground truths, v_i the ground truth's visibilities, k_i the key point specific fall-off constants, and s the scale of the person. We decide against calculating the OKS in our experiments because its constants were designed explicitly for RGB camera images and because we do not regress necessary key points like eyes, ears, and nose.

[5] *CONSORTIUM CoCO2: Prototype system for a Copernicus CO2 service.* URL: `https://coco2-project.eu/consortium` (visited on 2022-05-22)

	Key point	k_i
j_1, j_2	hips	0.107
j_3, j_4	ankles	0.089
j_5, j_6	knees	0.087
j_7, j_8	shoulders	0.079
j_9, j_{10}	elbows	0.072
j_{11}, j_{12}	wrists	0.062
j_{13}, j_{14}	ears	0.035
j_{15}	nose	0.026
j_{16}, j_{17}	eyes	0.025

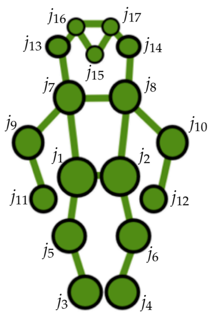


Figure 5.1.: *Fall-off constants* used as joint-specific weights in ***O**bject **K**eypoint **S**imilarity* (OKS) calculation [75]. The values presented here were proposed by the group of researchers from *CoCo project consortium*[5]. Smaller weights mean lower fault tolerance. We see that the nose and eyes, the finest key points, have the lowest weights, thus lowest fault tolerance and highest precision. Because we do not regress necessary key points like eyes, ears, and nose and because the OKS constants were designed explicitly for RGB camera images, we decide against calculating OKS in our experiments.

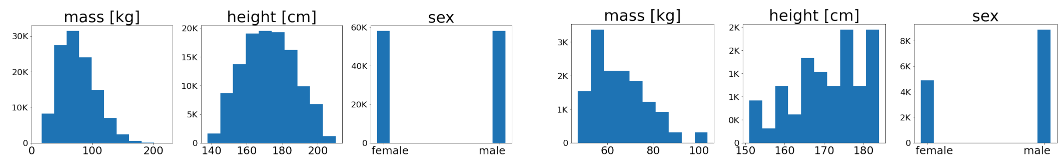
5.3. Datasets

Based on our research in section 2.5, we decide to use the Bodies at Rest synthetic dataset [18] as well as the SLP dataset [52] to train and evaluate our model. Additionally, we label some self-recorded samples to further evaluate how well our model can be expected to function on the actual target hardware. This section covers the preparation and splitting of the data and introduces our labeling procedure.

5.3.1. Data Preparation and Splitting

We start by preparing a single table for each of the datasets Bodies at Rest and SLP that includes for each sample a link to the 64×27 pressure image (8-bit unsigned precision), normalized skeletal joint annotations, sex, height, and mass. A Pandas data frame is used to store this table, which we write to disk as a *pickle* file. Each pressure recording is stored as a Numpy array file.

In order to extract the 24 joint coordinates from one sample of the Bodies at Rest dataset, we apply the provided SMPL parametrization to the sex-specific SMPL base model, transform it by the provided offsets, and normalize the resulting coordinates by dividing them by the size of the mattress. The resulting data frame has $116,000$ entries. In Figure 5.2a, we can see that mass and height follow a normal distribution and that the ratio of women to men is balanced.



(a) Histograms of the *Bodies at Rest* dataset. Mass and height follow a normal distribution, with a mean mass of 76 kilograms (values range from 17 to 222 kilograms) and a mean height of 1.73 meters (values range from 1.38 to 2.10 meters). The male/female distribution is 50%/50%.

(b) Histograms of the *SLP* dataset. The mean mass is 66 kilograms (values range from 47 to 104 kilograms) and the mean height 1.71 meters (values range from 1.51 to 1.84 meters). The male/female distribution is 64%/36%.

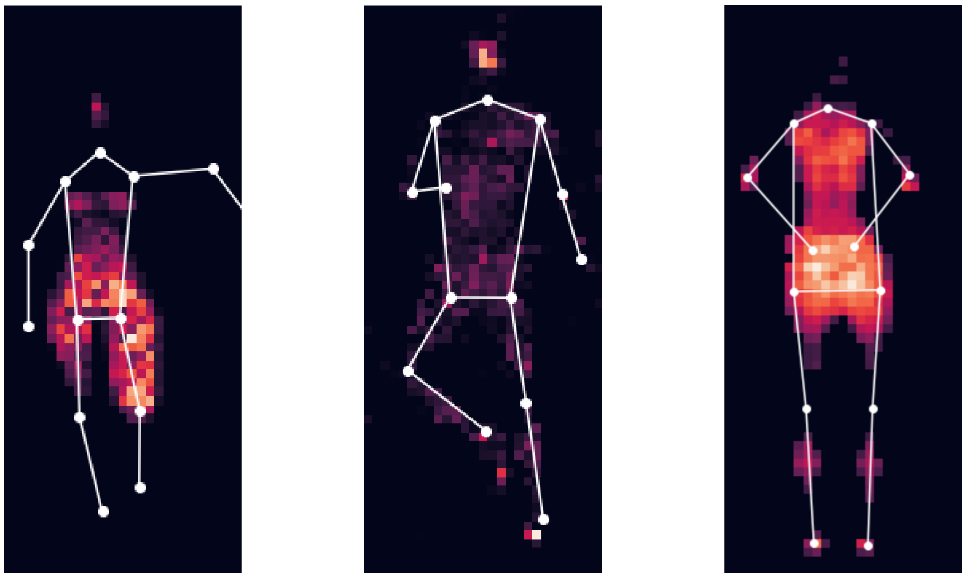
Figure 5.2.: Histograms for mass, height, and sex of the Bodies at Rest and SLP datasets

For the samples of the SLP dataset, we scale the provided 192×84 pressure recordings down to the target resolution of 64×27. The coordinates of the 14 annotated joints need to be aligned according to the calibration given as a homogenous transformation matrix. Furthermore, we need to normalize them by dividing them by the number of sensors in each dimension and flipping the result around the y-axis. The resulting data frame contains 13,770 rows. In Figure 5.2b, we can see that the distributions of mass and height do not follow standard distributions and that men are slightly overrepresented.

We proceed to randomly divide each of the two datasets into disjoint 80% training, 10% validation, and 10% test data; The Bodies at Rest set is split by samples, and the SLP set by patients. We then merge the two training sets. For that, we first need to convert the skeletons to a common format and, for this, decide to use the 13 joints present in both the 24-joint Bodies at Rest format and the 14-joint SLP format. In Figure 5.3, we can see that there are noticeable differences concerning intensity and noise in the Bodies at Rest and SLP datasets, which pose additional challenges to the model but also help it in generalizing (a topic further evaluated in chapter 6). We choose an oversampling factor of 9 for the SLP set to ensure that there are approximately the same amount of synthetic and real data points in the combined training set. For each set, the distribution of mass, height, and sex has been plotted and compared in order to ensure representativeness. Since synthetic data are not suitable for evaluation purposes, the validation and test samples from SLP will be in focus for this.

5.3.2. Annotation of self-recorded data

We want to use some self-recorded samples to quantify how well our model can be expected to function on the actual target hardware. Since the WiseMat mattress itself is still in a very prototypical state, we had to fall back on using the pressure mattress that was already present at Softline (see section 2.2). Pressure recordings



(a) Bodies at Rest **(b)** SLP **(c)** Softline

Figure 5.3.: Exemplary pressure recordings from the datasets *Bodies at Rest*, *SLP*, and *Softline*. We can see that the Bodies at Rest images are clearer, have a more balanced intensity distribution and include less noise than the SLP samples. The Softline samples actually look more similar to the Bodies at Rest data than to SLP.

of eight measurements with three people as well as video material recorded in parallel have been made available to our team. On randomly selected samples of these data, we annotated the 13 joint coordinates.

The recordings are given as *CSV* files. A sample can be found in Figure 5.3. Because they only have a resolution of 64×26, we pad them with a column of zeros. To annotate joint positions on these images, we implemented a Matplotlib-based labeling application, which you can see in Figure 5.4. We use this to label 50 samples and divide them evenly into validation and test sets. Because the video recordings are not always clear, the annotated joints may contain certain inaccuracies or mistakes. This applies in particular to the joints comprising the arm. It is also important to note that the amount of labeled data is too small to derive statistically reliable statements.

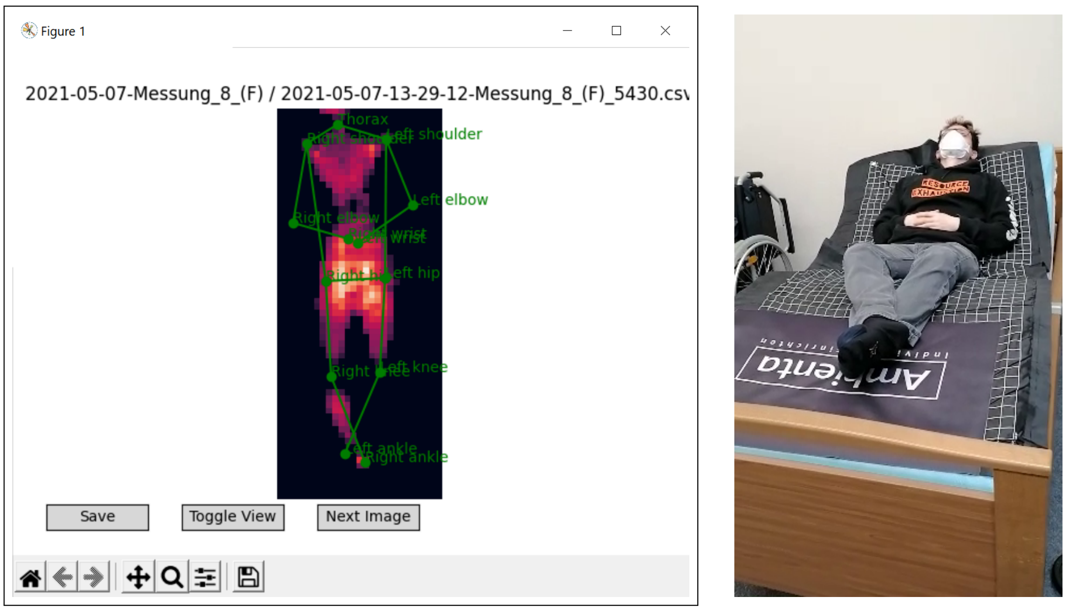


Figure 5.4.: Tools used to generate skeletal annotations for the Softline data: Basic Matplotlib-based labelling application (left) and the provided video recordings (right)

5.4. Baseline

In order to be able to assess the results better, we implement a simple baseline that, ignoring the input, always predicts a fixed placeholder skeleton. The position of each joint of this placeholder is calculated as the mean location of all the respective joints from the training dataset. The resulting skeleton can be seen in Figure A.2.

5.5. Direct regression approach using a CNN architecture

We start with the model architecture presented by Clever et al. [22], adapting the first layer to take a single pressure image of resolution 64×27 as input, and the final layer to regress the thirteen 2D joint coordinates of our dataset instead. The resulting detailed architecture diagram can be found in Figure 5.5.

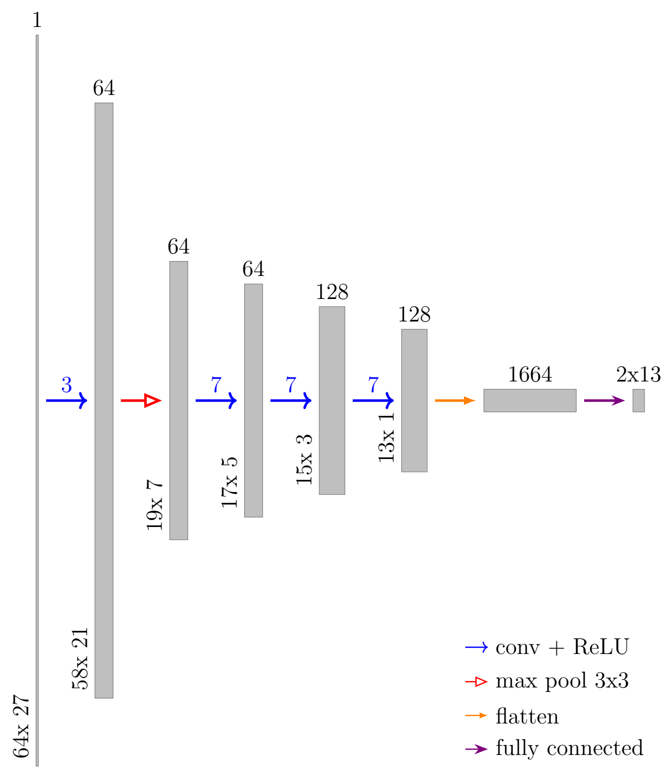


Figure 5.5.: Architecture used by Clever et al. [22], adapted to our input and output formats. Above the arrows are the convolutional kernel sizes. To mitigate the effects of overfitting, 10% dropout is applied before each convolution. We use this model as the starting point for our own experiments.

We found reading pressure images from the disk to be a bottleneck during training, even when increasing the number of worker threads. Therefore, we decided to evaluate a different approach, i.e., a new data frame was created that contains all pressure recordings. Since this file completely fits into our testing machine's main memory, we achieved a substantial speedup.

The Adam [42] algorithm is applied to perform the optimization of the MPJPE criterion. We use PyTorch Lightning's tuner functionality to determine an adequate learning rate automatically. Additionally, the *ReduceLROnPlateau* learning rate scheduler reduces the learning rate by a factor of 10 whenever the model no longer shows any improvements in validation loss for two epochs. Instead of a fixed number of epochs, *early stopping* is employed to train the model until the validation loss no longer improves by more than 0.2 mm for ten epochs. The batch size is set to 256. Across all epochs, the model that achieved the lowest validation loss is stored. This way, we protect ourselves from heavy overfitting that worsens the generalizability of our model.

We continue to conduct a series of experiments, iteratively building upon the result of the previous ones. Each experiment is evaluated in terms of several metrics on the SLP, Bodies at Rest and Softline validation datasets.

Experiment I: Training data

In a first experiment, the network is trained once each on the SLP training data, the Bodies at Rest training data, and the combined training data generated in subsection 5.3.1. Because the tuner did not work properly for both experiments based on only a single dataset, suggesting an extremely high learning rate on the order of 10^{-1}, we perform all three executions with the initial learning rate manually set to $2 \cdot 10^{-4}$.

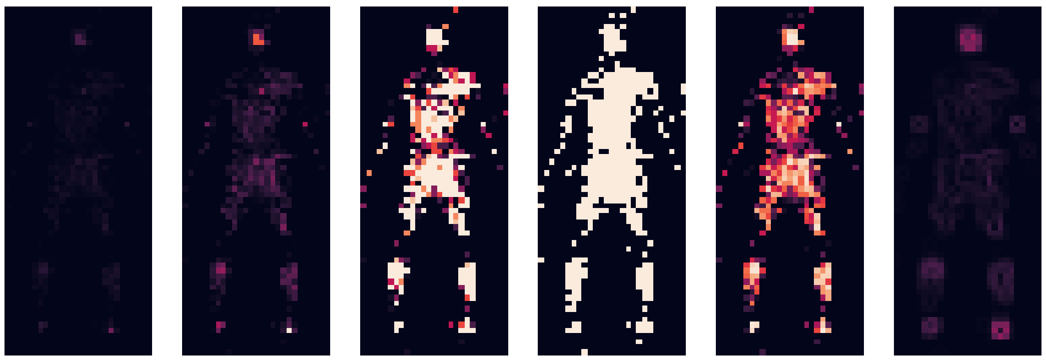
Experiment II: Preprocessing

Next, the impact of various preprocessing techniques on the model quality is evaluated. This includes no preprocessing, *min-max normalization*, *std-mean normalization*, a *boolean filter*, *histogram equalization*, and a *Sobel filter*. Figure 5.6 illustrates each of them with an example. For this experiment, we adapt the model to accept a stack of input images instead of just a single one. We start by training our model for each single preprocessing output and select the best-performing one. After this, we train our model adapted to take two inputs: The previously selected preprocessing output and each remaining one. We proceed analogously for three and more model inputs and repeat this process until we no longer find a subset of preprocessing methods that improves the model performance considerably.

Experiment III(A): Hyperparameter Optimization

Finally, we conduct a hyperparameter optimization using Optuna. The *Tree-structured Parzen Estimator* [11] sampling algorithm is employed for parameter selection. In 350 trials, various varying network architectures are trained and evaluated with the objective of minimizing the MPJPE. The search space comprises the following parameters:

- Size of the pooling applied in the first layer, varying between 1 (effectively no pooling) and 7
- Number of layers, sampled from the range 3 to 5
- For each of these layers:
  - A floating-point dropout probability, from the range 0% to 25%



(a) Original image **(b)** Min-max normalization **(c)** Std-mean normalization **(d)** Boolean filter **(e)** Histogram equalization **(f)** Sobel filter

Figure 5.6.: Exemplary illustrations of the preprocessing methods employed in experiment II. Brighter colors represent higher values. We can see that the normalizations and the histogram equalization help us to capture the content of the image, whereas the boolean filter generates a flat silhouette and the Sobel filter highlights edges.

- Size of the convolutional kernel, varying between 1 (effectively a linear transformation) and 7
- Number of channels this layer outputs, selected as a power of two between 2^4 and 2^8

5.6. Heatmap-based approach using the U-Net architecture

Related work found that direct regression of coordinates is not the optimal format for neural networks to learn. Instead, many scientific papers suggest that the prediction of heatmaps for each joint outperforms the direct regression [58, 80, 84].

In order to evaluate this solution for our problem, we start with the *U-Net* model architecture presented by Ronneberger et al. The architecture diagram of this network is given in Figure 5.7.

For this approach, we need to convert the thirteen 2D joint coordinates of the ground truth skeletons into 13 heatmap images with a resolution of 64×27 pixels, each showing a *Gaussian peak* at the ground truth joint location. We compute all the heatmaps in advance and serialize them to disk because we found this to be quicker than doing the conversion just-in-time. It is impossible to hold all the data with all its heatmap images simultaneously in the main memory of our testing machine. However, because the training of the much bigger U-Net architecture is slower than that of our direct regression approach, data loading becomes no bottleneck.

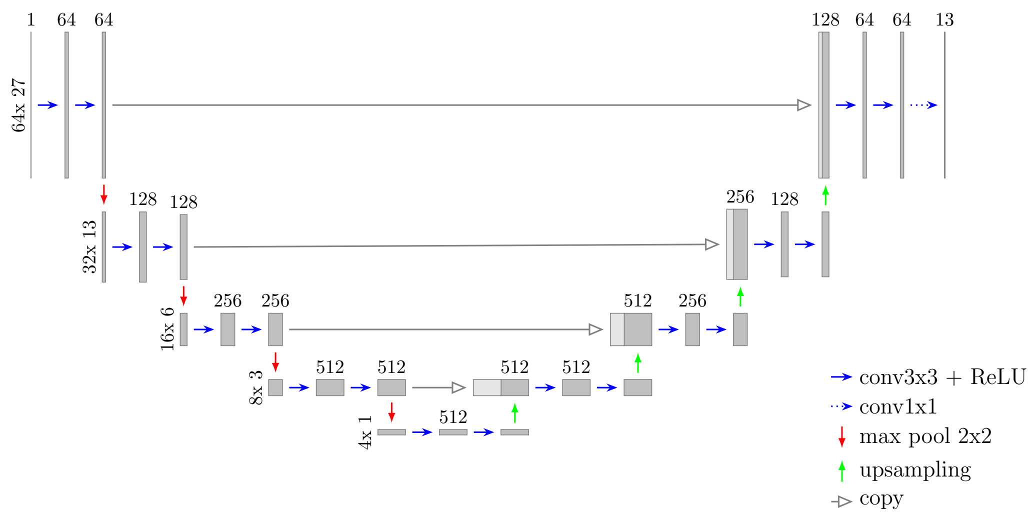


Figure 5.7.: *U-Net* architecture presented by Ronneberger et al. [68]. The first stage is a coarse-to-fine *Convolutional Neural Network* (CNN) that derives some high-level features. For this, each layer consists of two 3×3 convolutions plus ReLU followed by a 2×2 max pooling layer. In each second stage layer, additional double convolutions are applied to the upsampled output of the previous layer stacked on the respective output from the first stage's layer. Here, *bilinear upsampling* is employed to double the spatial resolution while maintaining the number of channels until the next convolution. Alternatively, *deconvolutions*, the inverse of convolutions, can be employed to perform the increase in resolution and reduction of channels all at once.

To obtain a skeletal prediction from a list of heatmaps returned by the network, we determine the pixel with the highest value, i.e. the argmax, for each joint's heatmap. Interestingly, this approach enables us to easily assign to each predicted joint a confidence score.

We train the model using the MSE optimization criterion introduced in Equation 2.1. That is, for each pixel of each heatmap, the squared difference between the predicted and the ground truth pixel values is summed up and the mean is calculated. Similar to the direct regression approach, the Adam [42] algorithm is applied to perform the optimization, and the batch size is kept at 256. We also continue to use PyTorch Lightning's learning rate tuner functionality and the ReduceLROnPlateau learning rate scheduler monitoring MPJPE with patience set to two epochs. Likewise, early stopping is employed to train the model until the MPJPE no longer improves by more than 0.2 mm for ten epochs. Once more, the intermediate model that achieved the lowest MPJPE is stored.

We continue to conduct another series of experiments, with each experiment building upon the previous findings. Whereas the procedures for experiments I and II remain as before (see section 5.5), the configuration of the hyperparameter optimization is different.

Experiment III(B): Hyperparameter Optimization

Here, 50 trials with the objective of minimizing the MPJPE are conducted over the search space comprising the following parameters:

- Whether to use *bilinear upsampling* or *deconvolutions* in the second stage
- Number of layers, sampled from the range 3 to 5
- For each of these layers: The number of output channels, selected as a power of two between 2^5 and 2^{11}

5.7. Hybrid approach

As evaluated in chapter 6, the direct regression approach suffers from a higher MPJPE, while the heatmap-based approach makes more mistakes that are not anatomically conclusive. Therefore, we decided to experiment with a hybrid approach, training a CNN to post-process the pressure image in combination with the heatmaps generated by the pre-trained U-Net into joint coordinates. This way, the post-processing network acts as a substitute for the argmax function.

6. Results

This chapter gives an overview of the results obtained from the experiments explained in chapter 5, and summarizes the test performance of our final model. The detailed documentation of all experiment results can be found in the appendix (Tables A.1, A.2, A.3).

6.1. Baseline

Our baseline achieves an MPJPE of 28.99 cm for the Bodies at Rest data set and 19.08 cm for the SLP data set. The worst results are obtained for the extremities, especially for the arms, i.e., elbows and wrists.

6.2. Direct regression approach

From the results of experiment I summarized in Figure 6.1, we can see that the models trained on only a single set perform poorly on the other. In contrast, the model trained on the combined set makes practicable predictions on all validation sets.

We conclude that the model trained on the combined dataset generalizes better because it achieves the best performance on the SLP and Softline data. Training the model based on the Bodies at Rest data takes more than ten times as long as training based on the SLP dataset, but using the combined set increases the training time only by another 15%.

From the various preprocessing methods examined in experiment II, we conclude that the best performance is achieved by the network that works on the images obtained from min-max normalization and the boolean filter stacked together. Overall, this improved our MPJPE by 4 mm on the Bodies at Rest and by 1 mm on the SLP dataset compared to employing no preprocessing at all. We could not identify considerable implications of the preprocessing methods employed on the duration of the training. Also, they do not have a considerable impact on the number of trainable parameters and model size: Using two input channels instead of just one added 3K new trainable parameters to the model and 13 KB to the model size (plus 1%).

Lastly, experiment III(A) suggested an alternative architecture that outperforms the original one. This adds 251K new trainable parameters to the model and 1.00 MB to the model size (plus 81%), but the training time remains in the same range

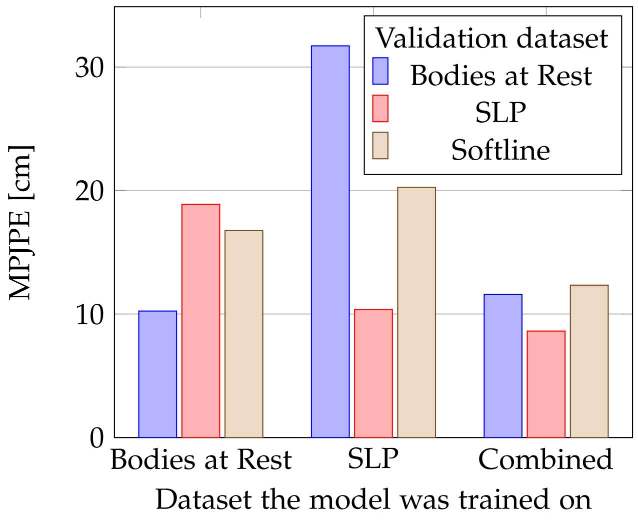


Figure 6.1.: *Mean Per Joint Position Error* (MPJPE) in centimeters of the direct regression models trained in experiment I on the *SLP*, *Bodies at Rest*, and *Softline* validation sets. We find that whereas the models trained on only a single set perform poorly on the others, the model trained on the combined set makes not only practicable predictions on all validation sets, but even achieves the best MPJPE for SLP and Softline data.

as the original architecture. This final architecture shown in Figure 6.2 consists of five convolutional layers, but already reaches its minimal spatial resolution after the third.

Figure 6.3 summarizes the quality of this model for the Bodies at Rest, SLP and Softline test datasets in terms of the metrics introduced in section 5.2. Training its 558K parameters took 11m 22s, resulting in a model that is 2.23 MB large. PyTorch Lightning determined an initial learning rate of $5 \cdot 10^{-4}$.

It is common practice for many regression problems to optimize the mean squared error instead of the mean absolute error in order to penalize higher errors more heavily. However, we experienced that this worsens our metrics. On the contrary, we found that already omitting arms during training can help improve the model's MPJPE by another 8 mm for "Bodies at rest" and 2 mm for SLP validation data.

6.3. Heatmap-based approach

Similar to the direct regression approach, the first experiment shows that training using the combined dataset leads to a higher model generalization, resulting in practicable metrics on all three validation datasets. However, it no longer achieves the very best performance on SLP and Softline data. Training with the Bodies at Rest dataset took once again longer than with the SLP set, but now only approxi-

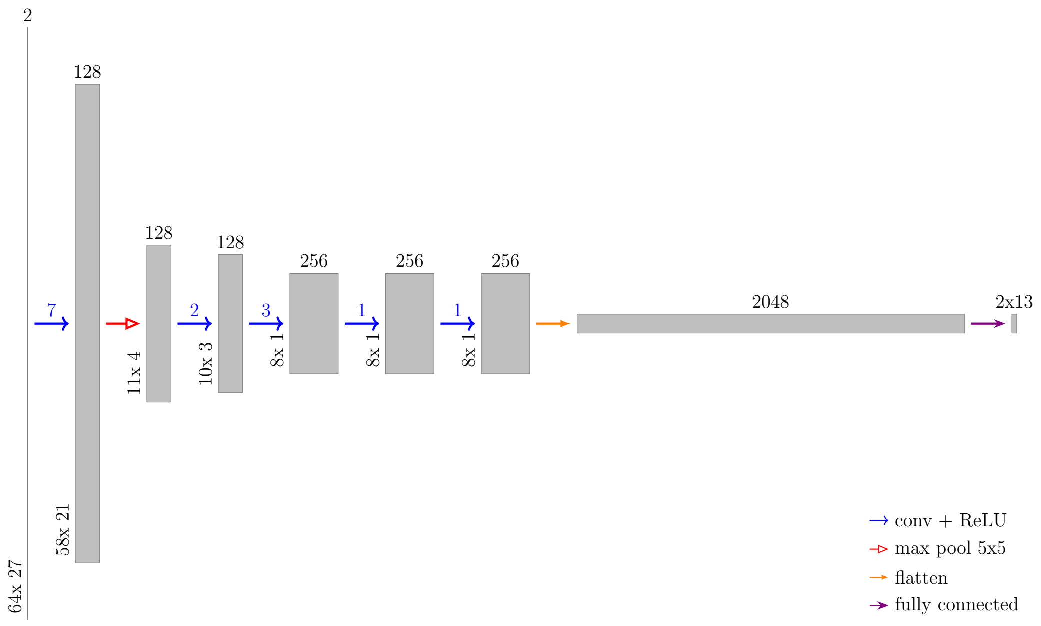


Figure 6.2.: Final architecture for the direct regression approach that evolved from the three experiments conducted. Above the arrow is the convolutional kernel size. Dropout is applied before each convolution, with approximately 20% before the first layer and only between 2 and 4% before each other.

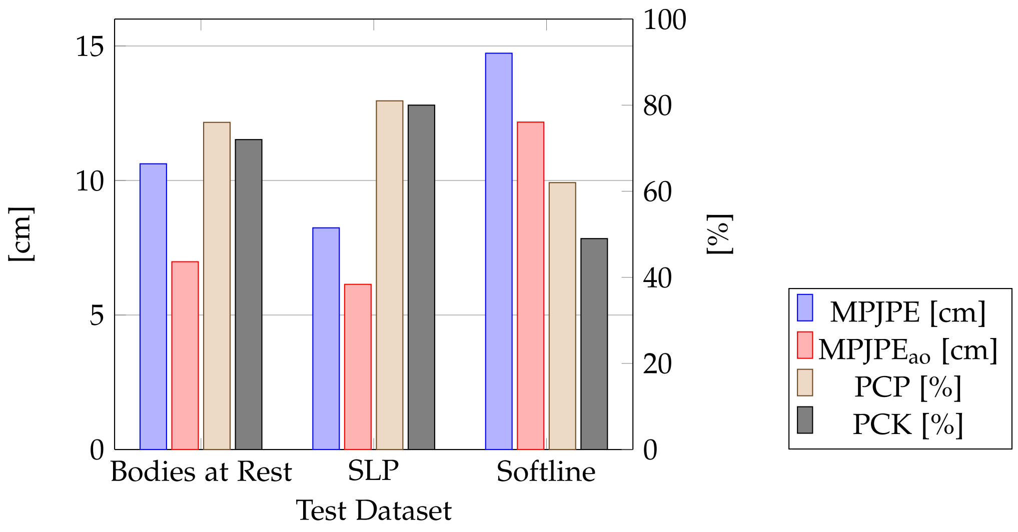


Figure 6.3.: Metrics of the final network for the direct regression approach on various test datasets. *Mean Per Joint Position Error* (MPJPE) in centimeters, *Percentage of Correct Keypoints* (PCK) and *Percentage of Correct Parts* (PCP) in percent. Tables A.4, A.5 and A.6 show the detailed numbers.

mately three times as long. Interestingly, the model trained on the combined data converged about 17% quicker than the one trained on Bodies at Rest exclusively.

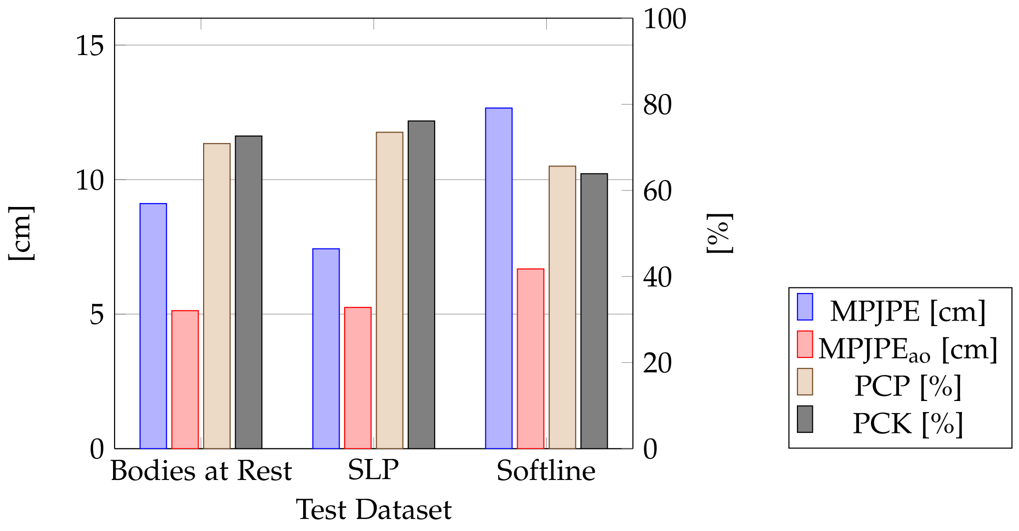


Figure 6.4.: Metrics of the final network for the heatmap-based approach on various test datasets. ***M**ean **P**er **J**oint **P**osition **E**rror* (MPJPE) in centimeters, ***P**ercentage of **C**orrect **K**eypoints* (PCK) and ***P**ercentage of **C**orrect **P**arts* (PCP) in percent. Tables A.4, A.5 and A.6 show the detailed numbers.

Experiment II leads us to use the three images obtained from standard-mean normalization, the boolean filter, and the histogram equalization. Overall, this improved our MPJPE by 11 mm on the Bodies at Rest and by 27 mm on the SLP dataset compared to employing no preprocessing at all. Even more so than for the analog experiment performed for the direct regression approach, we conclude that preprocessing methods have no considerable impact on the number of trainable parameters and model size (plus 0.005% for one additional input channel). For the duration of the training, on the other hand, we do see a trend towards an increase with the number of helpful inputs supplied.

For this approach, the hyperparameter search could not discover an architecture that outperformed the original one. We also found that the architecture based on deconvolutions is inferior to that based on upsampling, and that omitting arms during training actually worsens the metrics. Figure 6.4 summarizes the quality of the final model for the Bodies at Rest, SLP and Softline test datasets in terms of the metrics introduced in section 5.2. Training its 17.3M parameters took 1h 32m, resulting in a model that is 69.1 MB large. PyTorch Lightning determined a learning rate of $5 \cdot 10^{-4}$.

6.4. Hybrid approach

The complete model for this approach had a total size of 71.6 MB. Overall, we were able to lower the MPJPE on the Bodies at Rest data, but this in turn increased the MPJPE on the SLP data. Neither training only the 558K parameters of the coarse-to-fine CNN nor subsequently training the 17.8M parameters of the whole model led to a considerable improvement for both datasets. This is also due to severe overfitting; However, increasing the dropout probability of each layer of the CNN to 20% did not lead to an improvement either.

6.5. Comparison

In Table A.4, we can see that both the direct regression and the heatmap-based approach were able to outperform the baseline on every single joint in terms of mean error. Furthermore, it shows that the heatmap-based approach achieves better overall results than the direct regression. This applies in particular to the thorax, shoulders, hips, and knees. For ankles, elbows, and wrists, the standard deviation of the direct regression is lower, although the heatmap-based model remains largely superior in terms of MPJPE.

From the difference between MPJPE_{ao} and MPJPE in all results, as well as from the large standard deviation circles in Figure 6.6 and flat curves in figures A.9 and A.10, we conclude that the detection of the extremities, especially the arms, is particularly difficult. This is reasonable because the arms and especially the wrists are quite light. Because of this, however, they are also less prone to develop pressure ulcers in the first place [79]. One may therefore consider leaving them out entirely.

The heatmap-based approach regularly makes suggestions like the one visualized in Figure 6.5 that are not anatomically coherent, leading us to conclude that the U-Net finds it difficult to learn the interdependencies of human joints. We have not been able to observe such behavior for the direct regression. Rather, the arms poses predicted by it look quite reasonable, and we conclude that it must have learned some sort of geometric constraint model of the human body.

Concerning the fact that both models have never seen a single pressure image recorded using the Softline mattress during training, the results both models achieve on this dataset are decent. From the fact that the heatmap-based model's MPJPE_{ao} on Softline data is less than 30% worse than that achieved on Bodies at Rest and SLP, we conclude that the model generalizes well.

Overall, we can, in contrast to Li et al.'s findings, conclude that a heatmap-based approach is actually more suited for our problem [49]. Still, there are situations when the simpler direct regression would be preferable, for instance, when targeting embedded hardware with limited computing capabilities.

Our heatmap-based approach achieves a lower MPJPE on the SLP dataset than the model of Clever et al. (2018) on the "hrl-ros" dataset [22]. However, note that this comparison is of little significance not just because the underlying datasets differ but also because they perform 3D-joint regression where we only consider two dimensions, we use a 13-joint model where they use only ten joints, and we trained a model that aims to work well on various mattresses and not just a single one.

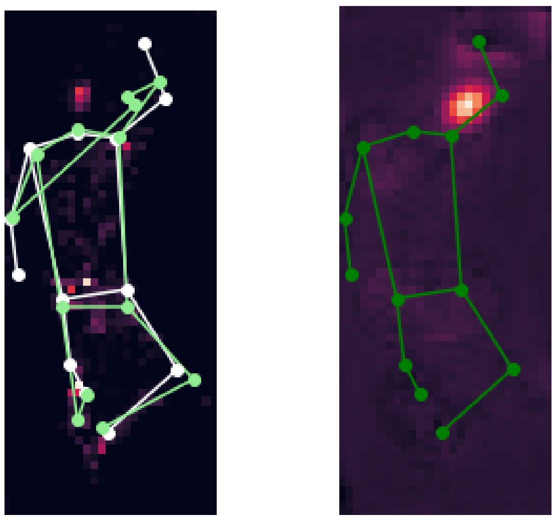
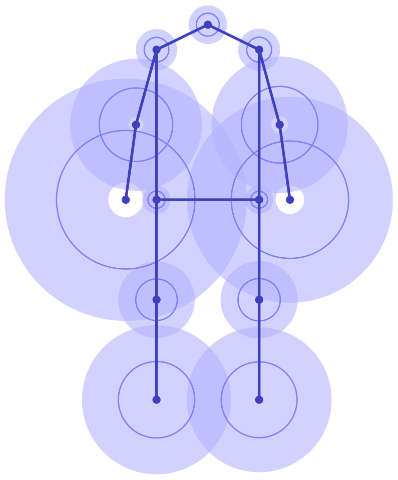
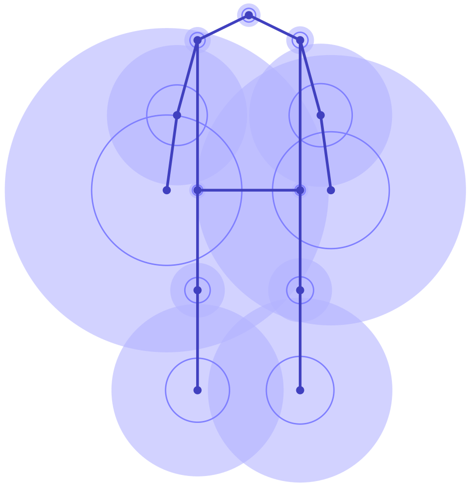
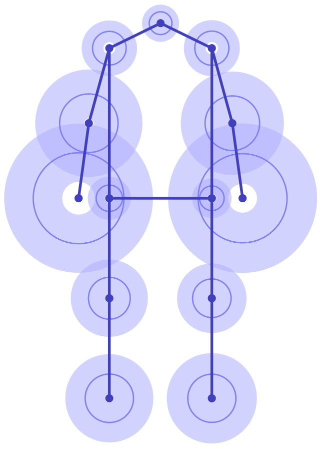


Figure 6.5.: The heatmap-based model occasionally produces output like the one visualized here, i.e., where predictions for one joint are very far off. Left: Pressure recording with the ground truth (white) and predicted (green) skeletons. Right: Heatmap predicted for the right wrist. For the direct regression approach, something like this hardly ever happens. We conclude that the *U-Net* finds it more difficult to learn generic anatomical constraints of the human body.

Similarly, we outperformed the work of Clever et al. (2020) in terms of MPJPE on the Bodies at Rest synthetic dataset. However, they predicted entire 3D model parameterizations and based their MPJPE calculation on 24 joints. We can not be certain whether this worsens their performance (because more joints in more dimensions bear the potential for more errors) or even rather improves it (because most of the additional joints lay within the torso area, which we found the easiest to predict).


(a) Direct regression approach evaluated on Bodies at Rest


(b) Heatmap-based approach evaluated on Bodies at Rest


(c) Direct regression approach evaluated on SLP

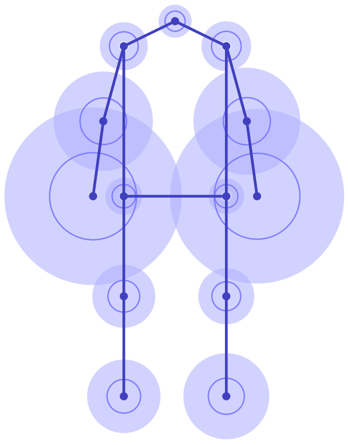

(d) Heatmap-based approach evaluated on SLP

Figure 6.6.: Visualization of mean and standard deviation of per-joint position errors for the direct regression and heatmap-based approaches on *Bodies at Rest* and *SLP* test data. One circle is drawn around each joint with a radius set to the mean error, and another circle is filled around the latter in the range of the standard deviation. We can see that the heatmap-based approach achieves better results for hips, shoulders, thorax, knees, and ankles, but suffers from a high variance of the arm predictions, especially on the Bodies at Rest dataset.

7. Conclusion

This thesis aimed to investigate the problem of 2D Human Pose Estimation, i.e., the derivation of skeletal joints, from single-frame pressure recordings. A solid solution to this is a basis for a pressure mattress-supported decubitus prophylaxis and can thus, in the future, help improve the quality of life for many bed-bound patients.

We explored two Deep Learning-based approaches, namely the direct regression using a coarse-to-fine CNN and the heatmap-based approximation using a U-Net, and found that these Machine Learning techniques are well-suited to solve the problem at hand, exceeding our baseline by far. The final direct regression model achieves an MPJPE of 10.65 cm for the Bodies at Rest data and 8.70 cm for the SLP data. The heatmap-based method outperforms the direct regression in all metrics, achieving an overall MPJPE of 9.11 cm for the Bodies at Rest data and 7.43 cm for the SLP data. This led us to draw the conclusion that this approach is the better option.

We found that it is often hardly possible to reconstruct arms and, most specifically, wrists. Because they are also less prone to pressure sores anyways [79], we recommend omitting them. By evaluating our model on self-recorded and self-labeled data, we showed that especially the heatmap-based approach generalizes well on different pressure mattresses, achieving an $MPJPE_{ao}$ that is only 30% worse than that achieved on the mattresses seen during training.

We conducted three sets of experiments for both models to improve our results by using different training datasets, employing several preprocessing techniques, and performing a hyperparameter optimization. We concluded that training data created by combining different individual datasets could help to build a more generalizing model. Furthermore, it is beneficial to explore image preprocessing methods, and hyperparameter optimizers can help discover enhanced architectures.

We found it difficult to compare our results to previous work because there is not one common dataset, skeleton format, or evaluation metric. Nevertheless, if we did so to the best of our ability, we found that we achieved state-of-the-art results and outperformed the work of Clever et al.

At the beginning of our work, we used *TensorDatasets* to load all our training data into the main memory in advance. With the introduction of the heatmap-based approach and various preprocessing methods, this approach was no longer feasible due to excessive memory requirements, and we had to switch to custom datasets based on an index referencing pressure recordings and heatmaps stored sample-wise in separate files. This, however, worsened the training time of the direct regression by a factor of five, presumably primarily due to a lot of random disk access. We, therefore, decided also to implement a preloading option for our custom dataset, which achieves a similar performance. We also faced several GPU-related issues during our work, which we solved by upgrading to the latest versions of PyTorch and the CUDA driver. Because our containers became incompatible after an update of the DE-Lab, we could not execute additional hyperparameter optimization runs without disproportionate effort.

We also gave an overview of annoto, a web-based medical data labeling tool we created during the bachelor project phase. We experienced that the Scrum framework and agile practices employed to manage our development workflow supported us in efficiently building robust, well-documented, easily maintainable, and expandable software. Following the open-source movement, we published our work (that is, the annotation tool as well as the Pose Estimation models) under the MIT license on Github.[1][2]

Models like the ones presented in this thesis could be combined with the annotation tool to simplify and accelerate the labeling process. Future work may explore other approaches, e.g., Convolutional Pose Machines or Stacked Hourglass models, which also come from the domain of RGB camera images, or simply a coarse-to-fine CNN with a higher number of trainable parameters or layers. Data augmentation can be employed, and other metrics can be examined, for example, introducing a joint weighting similar to OKS. Additionally, confidence scores may be used to omit joints our network is so uncertain about that it is tantamount to guessing.

Further important steps towards the practical use of pressure mattresses for decubitus prophylaxis concern the consideration of temporal consistency, pressure accumulation, and, based on this, the risk assessment itself. When the target hardware is ready, we suggest generating a complete dataset with a wide variety of subjects in various poses and including this data in training and validation.

[1]*Annoto*. URL: https://github.com/team-folivora/annoto.
[2]*WiseMat Pose Estimation*. URL: https://github.com/bewee/wisemat-pose-estimation.

References

[1] M. Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. Mar. 16, 2016. DOI: 10.48550/arXiv.1603.04467. arXiv: 1603.04467[cs].

[2] F. Achilles, A.-E. Ichim, H. Coskun, F. Tombari, S. Noachtar, and N. Navab. "Patient MoCap: Human Pose Estimation Under Blanket Occlusion for Hospital Monitoring Applications". In: *MICCAI*. 2016. DOI: 10.1007/978-3-319-46720-7_57.

[3] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. "Optuna: A next-generation hyperparameter optimization framework". In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, pages 2623–2631.

[4] F. Anders. "Überprüfung: Schwere Mängel in der Altenpflege". In: *DIE WELT* (Aug. 13, 2007).

[5] A. Ard. *If not Scrum, then what?* Medium. Dec. 13, 2020. URL: https://medium.com/@ard_adam/the-nature-of-computer-programming-7526789b3af1 (visited on 2022-07-20).

[6] B. Artacho and A. Savakis. "UniPose: Unified Human Pose Estimation in Single Images and Videos". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020, pages 7035–7044.

[7] N. Aylward-Wotton and B. Kent. *Improving pressure damage detection in the community using continuous pressure monitoring of patients*. Cornwall Partnership Foundation NHS Trust.

[8] S. C. Babu. *A 2019 guide to Human Pose Estimation with Deep Learning*. AI & Machine Learning Blog. Apr. 12, 2019. URL: https://nanonets.com/blog/human-pose-estimation-2d-guide/ (visited on 2022-03-22).

[9] M. Baran Pouyan, J. Birjandtalab, M. Heydarzadeh, S. Ostadabbas, R. Yousefi, M. Farshbaf, M. Nourani, and M. Pompeo. *A Pressure Map Dataset for In-bed Posture Classification (pmd)*. 2018. DOI: 10.13026/C2W09K.

[10] T. Bauknecht. "Optimization of Machine Learning Models on Embedded Devices for the Analysis of ECG Data". Bachelor's Thesis. Potsdam: Hasso Plattner Institute, 2022.

[11] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. "Algorithms for hyper-parameter optimization". In: *Advances in neural information processing systems* 24 (2011).

[12] J. Berndt. “The effect of image preprocessing on the accuracy of neural networks using the example of real-time posture classification”. Bachelor’s Thesis. Potsdam: Hasso Plattner Institute, 2022.

[13] L. A. Borojeny, A. N. Albatineh, A. H. Dehkordi, and R. G. Gheshlagh. “The Incidence of Pressure Ulcers and its Associations in Different Wards of the Hospital: A Systematic Review and Meta-Analysis”. In: *International Journal of Preventive Medicine* 11 (Oct. 2020). ISSN: 2008-7802. DOI: 10.4103/ijpvm.IJPVM_182_19.

[14] J. Brownlee. *A Gentle Introduction to k-fold Cross-Validation*. Machine Learning Mastery. May 22, 2018. URL: https://machinelearningmastery.com/k-fold-cross-validation/ (visited on 2022-07-18).

[15] L. Böhlke. *Continuous Deployment*. Nov. 4, 2021. URL: https://mindsquare.de/knowhow/continuous-deployment/ (visited on 2022-06-29).

[16] L. Casas, N. Navab, and S. Demirci. “Patient 3D body pose estimation from pressure imaging”. In: *International Journal of Computer Assisted Radiology and Surgery* 14.3 (Mar. 2019), pages 517–524. ISSN: 1861-6410, 1861-6429. DOI: 10.1007/s11548-018-1895-3.

[17] M. Clark, M. Romanelli, S. I. Reger, V. K. Ranganathan, J. Black, and C. Dealey. “Microclimate in Context”. In: *International Review Pressure Ulcer Prevention: Pressure, Shear, Friction and Microclimate in Context – a Consensus Document* (2010), pages 19–25.

[18] H. Clever. *Bodies at Rest – PressurePose Real Dataset*. Mar. 31, 2020. DOI: 10.7910/DVN/KOA4ML.

[19] H. M. Clever. *Bodies at Rest – PressurePose Synthetic Dataset*. Mar. 30, 2020. DOI: 10.7910/DVN/IAPI0X.

[20] H. M. Clever, Z. Erickson, A. Kapusta, G. Turk, C. Liu, and C. C. Kemp. “Bodies at Rest: 3D Human Pose and Shape Estimation From a Pressure Image Using Synthetic Data”. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). June 2020, pages 6214–6223. DOI: 10.1109/CVPR42600.2020.00625.

[21] H. M. Clever, P. Grady, G. Turk, and C. C. Kemp. “BodyPressure – Inferring Body Pose and Contact Pressure from a Depth Image”. In: *arXiv preprint arXiv:2105.09936* (May 20, 2021).

[22] H. M. Clever, A. Kapusta, D. Park, Z. Erickson, Y. Chitalia, and C. C. Kemp. “3D Human Pose Estimation on a Configurable Bed from a Pressure Image”. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Oct. 2018, pages 54–61. DOI: 10.1109/IROS.2018.8593545.

[23] P. Cunningham and S. Delany. “k-Nearest neighbour classifiers”. In: *Mult Classif Syst* 54 (Apr. 27, 2007). DOI: 10.1145/3459665.

[24] M. Dabas, D. Schwartz, D. Beeckman, and A. Gefen. "Application of Artificial Intelligence Methodologies to Chronic Wound Care and Management: A Scoping Review". In: *Advances in Wound Care* (Apr. 19, 2022). ISSN: 2162-1918. DOI: 10.1089/wound.2021.0144.

[25] M. Dantone, J. Gall, C. Leistner, and L. Van Gool. "Human Pose Estimation Using Body Parts Dependent Joint Regressors". In: 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Portland, OR, USA: IEEE, June 2013, pages 3041–3048. ISBN: 978-0-7695-4989-7. DOI: 10.1109/CVPR.2013.391.

[26] V. Davoodnia, S. Ghorbani, and A. Etemad. "In-bed Pressure-based Pose Estimation using Image Space Representation Learning". In: *ICASSP 2021 – 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. Sept. 2020, pages 3965–3969.

[27] European Pressure Ulcer Advisory Panel & National Pressure Ulcer Advisory Panel. "Prevention and Treatment of Pressure Ulcers: Quick Reference Guide". In: *Natl. Press. Ulcer Advis. Panel* (2014), pages 1–75.

[28] M. Farshbaf, S. Ostadabbas, R. Yousefi, M. Nourani, and M. Pompeo. "Pressure ulcer monitoring and intervention: A software platform". In: 2011 IEEE International Conference on Bioinformatics and Biomedicine Workshops (BIBMW). Atlanta, GA: IEEE, Nov. 2011, pages 897–898. ISBN: 978-1-4577-1613-3, 978-1-4577-1612-6. DOI: 10.1109/BIBMW.2011.6112493.

[29] R. T. Fielding. "Architectural Styles and the Design of Network-based Software Architectures". Doctoral dissertation. Irvine: University of California, 2000.

[30] M. A. Fischler and R. Elschlager. "The Representation and Matching of Pictorial Structures". In: *IEEE Transactions on Computers* C-22.1 (Jan. 1973), pages 67–92. ISSN: 0018-9340. DOI: 10.1109/T-C.1973.223602.

[31] M. Fowler. *Continuous Integration*. May 1, 2006. URL: https://martinfowler.com/articles/continuousIntegration.html (visited on 2022-06-27).

[32] S. Gallinger, N. Jankowski, M. Bister, S. Korge, A. Trachterna, S. Hildebrand, T. Oruc, J. Niewöhner, U. Heitman, R. Downes, and M. Kraft. "Development of a modular Decubitus Prophylaxis System: DekuProSys". In: *Current Directions in Biomedical Engineering* 5.1 (Sept. 2019), pages 277–279. ISSN: 2364-5504. DOI: 10.1515/cdbme-2019-0070.

[33] A. Gray. *A Criticism of Scrum*. Oct. 24, 2015. URL: https://www.aaron-gray.com/a-criticism-of-scrum/ (visited on 2022-07-20).

[34] R. Grimm, S. Bauer, J. Sukkau, J. Hornegger, and G. Greiner. "Markerless estimation of patient orientation, posture and pose using range and pressure imaging : for automatic patient setup and scanner initialization in tomographic imaging". In: *International Journal of Computer Assisted Radiology and Surgery* 7.6 (Nov. 2012), pages 921–929. ISSN: 1861-6429. DOI: 10.1007/s11548-012-0694-5.

[35] A. Hammer and E. Kobbert. “Dekubitusprophylaxe”. In: *verstehen & pflegen 4 – Prävention und Rehabilitation*. Stuttgart: Thieme, 2017, pages 250–276.

[36] C. R. Harris et al. “Array programming with NumPy”. In: *Nature* 585.7825 (Sept. 2020), pages 357–362. DOI: 10.1038/s41586-020-2649-2.

[37] K. Hornik. “Approximation capabilities of multilayer feedforward networks”. In: *Neural Networks* 4.2 (Jan. 1, 1991), pages 251–257. ISSN: 0893-6080. DOI: 10.1016/0893-6080(91)90009-T.

[38] J. D. Hunter. “Matplotlib: A 2D graphics environment”. In: volume 9. 3, pages 90–95.

[39] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. “Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.7 (July 2014), pages 1325–1339. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2013.248.

[40] P. D. C. Johner, C. Molnar, F. Hohlwegler, S. Piechottka, and M. Klinger-Ji. *AI Guideline*. June 21, 2022. URL: https://github.com/johner-institut/ai-guideline/blob/defff8c13a59912c0e77667711b66972b656dfb9/Guideline-AI-Medical-Devices_EN.md (visited on 2022-06-27).

[41] A. Jung. *Machine Learning: The Basics*. Singapore: Springer, 2022. ISBN: 981-16-8192-9, 981-16-8193-7.

[42] D. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations* (Dec. 22, 2014).

[43] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas. “Data Preprocessing for Supervised Learning”. In: *International Journal of Computer Science* 1 (Jan. 2006), pages 111–117.

[44] J. Kottner and T. Dassen. “Pressure ulcer risk assessment in critical care: interrater reliability and validity studies of the Braden and Waterlow scales and subjective ratings in two intensive care units”. In: *International Journal of Nursing Studies* 47.6 (June 2010), pages 671–677. ISSN: 1873-491X. DOI: 10.1016/j.ijnurstu.2009.11.005.

[45] N. König. “Measuring energy efficiency of artificial intelligence”. Bachelor’s Thesis. Potsdam: Hasso Plattner Institute, 2022.

[46] A. Labach, H. Salehinejad, and S. Valaee. *Survey of Dropout Methods for Deep Neural Networks*. Oct. 25, 2019. arXiv: 1904.13310[cs].

[47] R.-M. Lehmann (FDP) and Abgeordnetenhaus Berlin. *Kleine Anfrage und Antwort: Häusliche Pflege in Berlin*. Drucksache 16 / 10 937. July 23, 2007.

[48] K. LeViet and Y.-h. Chen. *Pose estimation and classification on edge devices with MoveNet and TensorFlow Lite*. Aug. 16, 2021. URL: https://blog.tensorflow.org/2021/08/pose-estimation-and-classification-on-edge-devices-with-MoveNet-and-TensorFlow-Lite.html (visited on 2022-01-08).

[49] Y. Li, S. Yang, S. Zhang, Z. Wang, W. Yang, S.-T. Xia, and E. Zhou. *Is 2D Heatmap Representation Even Necessary for Human Pose Estimation?* July 11, 2021. arXiv: 2107.03332[cs].

[50] A. Lichterfeld-Kottner, N. Lahmann, and J. Kottner. "Sex-specific differences in prevention and treatment of institutional-acquired pressure ulcers in hospitals and nursing homes". In: *Journal of Tissue Viability*. SI: Leg Ulceration 29.3 (Aug. 1, 2020), pages 204–210. ISSN: 0965-206X. DOI: 10.1016/j.jtv.2020.05.001.

[51] J. Liu, M.-C. Huang, W. Xu, and M. Sarrafzadeh. "Bodypart localization for pressure ulcer prevention". In: *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC 2014* (Aug. 2014), pages 766–769. DOI: 10.1109/EMBC.2014.6943703.

[52] S. Liu, X. Huang, N. Fu, C. Li, Z. Su, and S. Ostadabbas. "Simultaneously-Collected Multimodal Lying Pose Dataset: Enabling In-Bed Human Pose Monitoring". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022). ISSN: 1939-3539. DOI: 10.1109/TPAMI.2022.3155712.

[53] C. Ma and J. Wu. *Data Clustering: Theory, Algorithms, and Applications*. Volume 20. Jan. 1, 2007. DOI: 10.1137/1.9780898718348.

[54] Wes McKinney. "Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference*. Edited by Stéfan van der Walt and Jarrod Millman. 2010, pages 56 –61. DOI: 10.25080/Majora-92bf1922-00a.

[55] D. Merkel. "Docker: lightweight linux containers for consistent development and deployment". In: *Linux journal* 2014.239 (2014), page 2.

[56] V. Metsis, G. Galatas, A. Papangelis, D. Kosmopoulos, and F. Makedon. "Recognition of sleep patterns using a bed pressure mat". In: ACM International Conference Proceeding Series. 2011, page 9. DOI: 10.1145/2141622.2141633.

[57] P. Murugan and S. Durairaj. *Regularization and Optimization strategies in Deep Convolutional Neural Network*. Dec. 13, 2017. arXiv: 1712.04711[cs].

[58] A. Newell, K. Yang, and J. Deng. "Stacked Hourglass Networks for Human Pose Estimation". In: *Computer Vision – ECCV 2016*. Edited by B. Leibe, J. Matas, N. Sebe, and M. Welling. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pages 483–499. ISBN: 978-3-319-46484-8. DOI: 10.1007/978-3-319-46484-8_29.

[59] F. Nightingale. *Notes on nursing for the labouring classes*. Harrison, 1861. ISBN: 978-0-343-66013-0.

[60] F. Osterloh. "Pflegekräfte: Den Personalbedarf messen". In: volume 116 (35 - 36). Deutsches Ärzteblatt, Sept. 2, 2019.

[61] G. Papandreou, T. Zhu, L.-C. Chen, S. Gidaris, J. Tompson, and K. Murphy. "PersonLab: Person Pose Estimation and Instance Segmentation with a Bottom-Up, Part-Based, Geometric Embedding Model". In: *Computer Vision – ECCV 2018*. Edited by V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pages 282–299. ISBN: 978-3-030-01264-9. DOI: 10.1007/978-3-030-01264-9_17.

[62] A. Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf. Curran Associates, Inc., 2019, pages 8024–8035.

[63] P. Ploton, F. Mortier, M. Réjou-Méchain, N. Barbier, N. Picard, V. Rossi, C. Dormann, G. Cornu, G. Viennois, N. Bayol, A. Lyapustin, S. Gourlet-Fleury, and R. Pélissier. "Spatial validation reveals poor predictive performance of large-scale ecological mapping models". In: *Nature Communications* 11.1 (Sept. 11, 2020), page 4540. ISSN: 2041-1723. DOI: 10.1038/s41467-020-18321-y.

[64] M. Q. Pompeo. "The role of "wound burden" in determining the costs associated with wound care". In: *Ostomy/Wound Management* 47.3 (Mar. 2001), pages 65–71. ISSN: 0889-5899.

[65] B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, and S. Ganguli. "Exponential expressivity in deep neural networks through transient chaos". In: *Advances in Neural Information Processing Systems*. Edited by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Volume 29. Curran Associates, Inc., 2016.

[66] *posture, n.* In: *Oxford English Dictionary Online*. Oxford University Press, Mar. 2022.

[67] H. Robbins and S. Monro. "A Stochastic Approximation Method". In: *The Annals of Mathematical Statistics* 22.3 (Sept. 1951), pages 400–407. ISSN: 0003-4851, 2168-8990. DOI: 10.1214/aoms/1177729586.

[68] O. Ronneberger, P. Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Edited by N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi. Volume 9351. Cham: Springer International Publishing, 2015, pages 234–241. ISBN: 978-3-319-24573-7, 978-3-319-24574-4. DOI: 10.1007/978-3-319-24574-4_28.

[69] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (Oct. 1986), pages 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0.

[70] J. Sauer. "Gamification in healthcare - investigating attention and motivation enhancement". Bachelor's Thesis. Potsdam: Hasso Plattner Institute, 2022.

[71] K. Schwaber and J. Sutherland. *The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game*. Nov. 2020. URL: https://scrumguides.org/ (visited on 2022-07-01).

[72] W. O. Seiler. *Immobilität als Hauptrisikofaktor. Hightech Mobility-Monitoring in der modernen Dekubitusprophylaxe*. Volume 48. 2. Jan. 23, 2015.

[73] C. Shorten and T. M. Khoshgoftaar. "A survey on image data augmentation for deep learning". In: *Journal of big data* 6.1 (2019), pages 1–48.

[74] D. M. Smith, D. E. Snow, E. Rees, A. M. Zischkau, J. D. Hanson, R. D. Wolcott, Y. Sun, J. White, S. Kumar, and S. E. Dowd. "Evaluation of the bacterial diversity of Pressure ulcers using bTEFAP pyrosequencing". In: *BMC Medical Genomics* 3.1 (Sept. 21, 2010), page 41. ISSN: 1755-8794. DOI: 10.1186/1755-8794-3-41.

[75] A. Stasiuk. *Pose Estimation. Metrics.* Medium. Nov. 11, 2020. URL: https://stasiuk.medium.com/pose-estimation-metrics-844c07ba0a78 (visited on 2022-01-14).

[76] M. Sun and S. Savarese. "Articulated part-based model for joint object detection and pose estimation". In: *2011 International Conference on Computer Vision*. 2011 IEEE International Conference on Computer Vision (ICCV). Barcelona, Spain: IEEE, Nov. 2011, pages 723–730. ISBN: 978-1-4577-1102-2, 978-1-4577-1101-5, 978-1-4577-1100-8. DOI: 10.1109/ICCV.2011.6126309.

[77] The Hague: Health Council of the Netherlands. *Pressure ulcers*. Apr. 5, 2010. URL: www.gezondheidsraad.nl/en/publications/pressure-ulcers (visited on 2022-04-12).

[78] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134.

[79] T. Tomova-Simitchieva, A. Lichterfeld-Kottner, U. Blume-Peytavi, and J. Kottner. "Comparing the effects of 3 different pressure ulcer prevention support surfaces on the structure and function of heel and sacral skin: An exploratory cross-over trial". In: *International Wound Journal* 15.3 (June 1, 2018), pages 429–437. ISSN: 1742-481X. DOI: 10.1111/iwj.12883.

[80] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. "Efficient object localization using Convolutional Networks". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Boston, MA, USA: IEEE, June 2015, pages 648–656. ISBN: 978-1-4673-6964-0. DOI: 10.1109/CVPR.2015.7298664.

[81] A. Toshev and C. Szegedy. "DeepPose: Human Pose Estimation via Deep Neural Networks". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Columbus, OH, USA: IEEE, June 2014, pages 1653–1660. ISBN: 978-1-4799-5118-5. DOI: 10.1109/CVPR.2014.214.

[82] W. T. Wake. "Pressure Ulcers: What Clinicians Need to Know". In: *The Permanente Journal* 14.2 (2010), pages 56–60. ISSN: 1552-5767.

[83] M. Waskom. "seaborn: statistical data visualization". In: *Journal of Open Source Software* 6.60 (Apr. 6, 2021), page 3021. ISSN: 2475-9066. DOI: 10.21105/joss.03021.

[84] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. "Convolutional Pose Machines". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV, USA: IEEE, June 2016, pages 4724–4732. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.511.

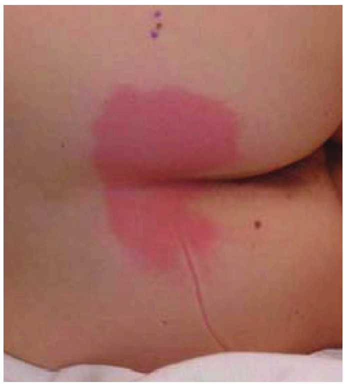
[85] J. Weng. *Data Splitting for Model Evaluation*. Medium. Oct. 15, 2021. URL: https://towardsdatascience.com/data-splitting-for-model-evaluation-d9545cd04a99 (visited on 2022-07-18).

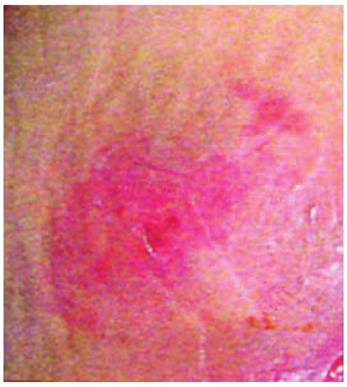
[86] *White paper. New Technology Significantly Reduces Rates of Hospital Acquired Pressure Injuries*. Leaf Healthcare, Inc.

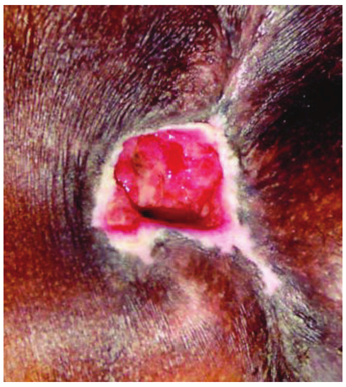
[87] F. William and T. P. L. team. "PyTorch Lightning". In: Mar. 30, 2019. DOI: 10.5281/zenodo.3828935.

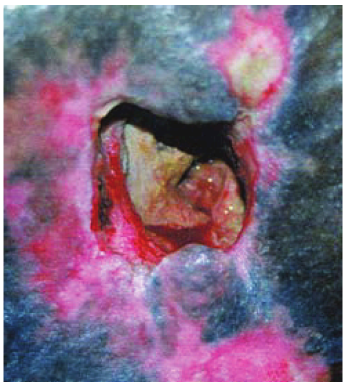
[88] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola. "Dive into Deep Learning". In: *arXiv preprint arXiv:2106.11342* (2021).

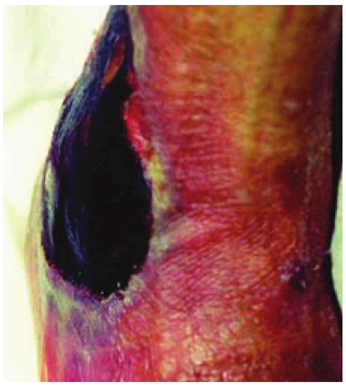
A. Appendix


(a) Stage 1 pressure ulcer: Redness of a localized area, which may be painful, firm, soft, warmer or cooler. Skin remains intact.


(b) Stage 2 pressure ulcer: Shiny or dry shallow open ulcer with a red pink wound bed or serum-filled blister. The dermis suffers a partial thickness loss.


(c) Stage 3 pressure ulcer: Subcutaneous fat may be visible. Depth varies greatly by anatomical location. May include undermining or tunneling. Full thickness tissue loss.


(d) Stage 4 pressure ulcer: Exposed bone, tendon or muscle is visible. Depth varies greatly by anatomical location. Often include undermining or tunneling.


(e) *Unstageable* pressure ulcer: Base of the ulcer is covered by slough or eschar, thus true depth and thus stage cannot be determined.

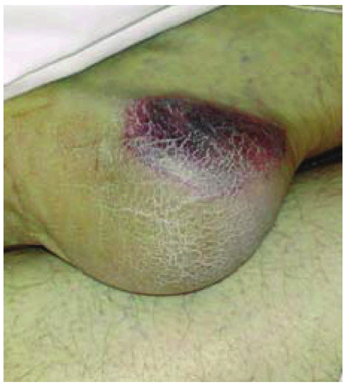

(f) *Deep tissue injury*: Purple or maroon localized area of discolored intact skin or blood-filled blister. Damage to underlying soft tissue.

Figure A.1.: *Pressure ulcers* of the four stages as well as examples of an unstagable ulcer and a deep tissue injury, with characteristics used for their classification. [27]

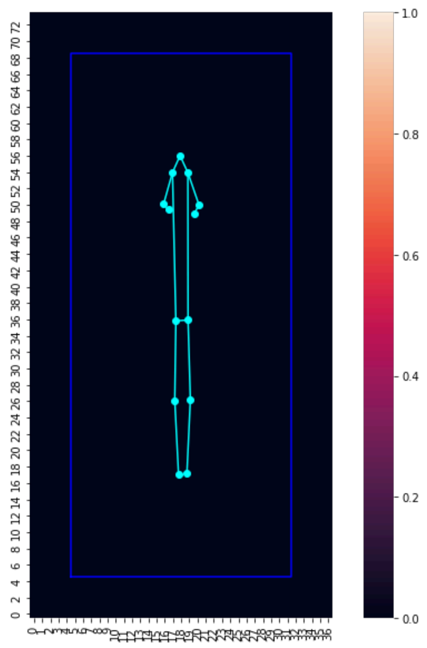


Figure A.2.: Mean skeleton calculated from the training data. Our baseline algorithm ignores the input and always predicts this. In doing so, it achieves an MPJPE of 28.99 cm for the *Bodies at Rest* dataset and 19.08 cm for the *SLP* dataset.

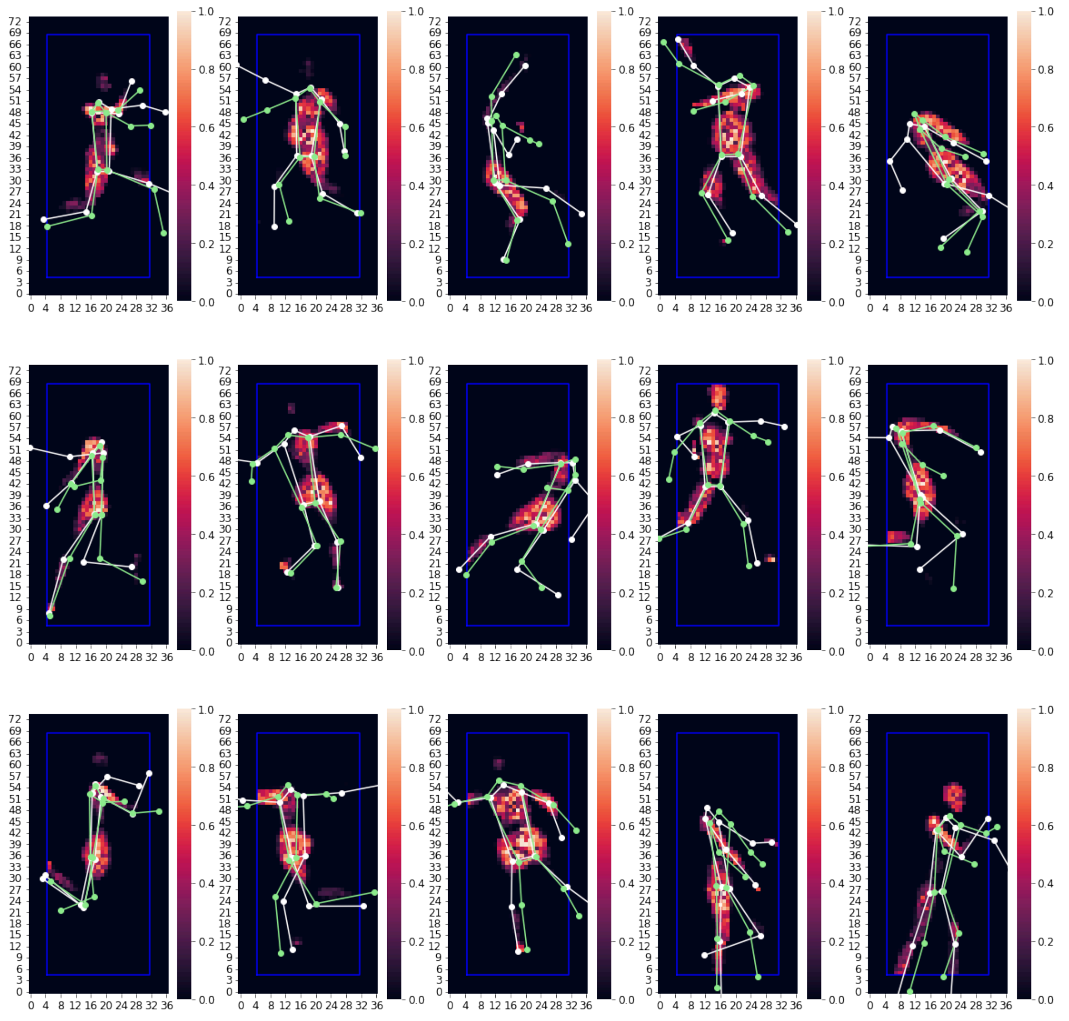


Figure A.3.: Exemplary results of the direct regression approach for the *Bodies at Rest* data. Ground truth in white, prediction in green

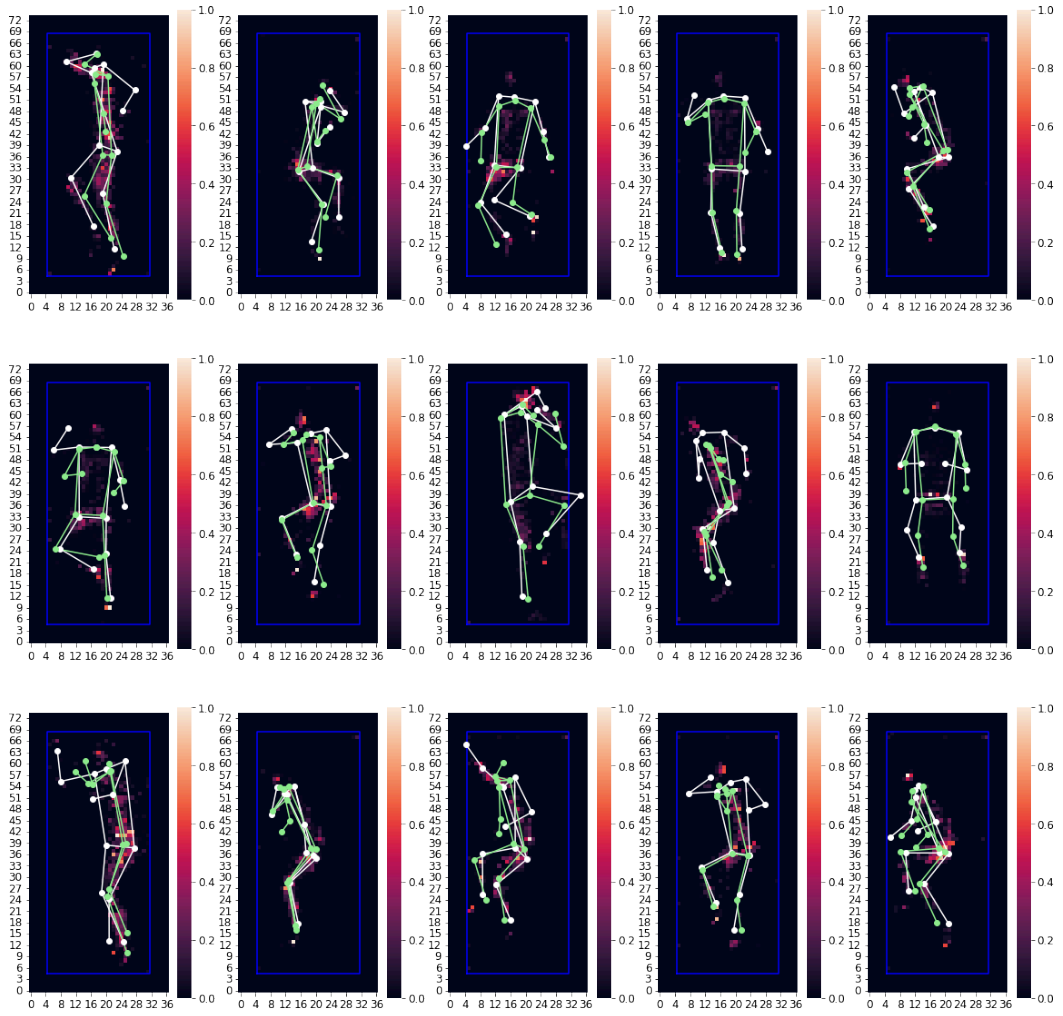


Figure A.4.: Exemplary results of the direct regression approach for the *SLP* data. Ground truth in white, prediction in green

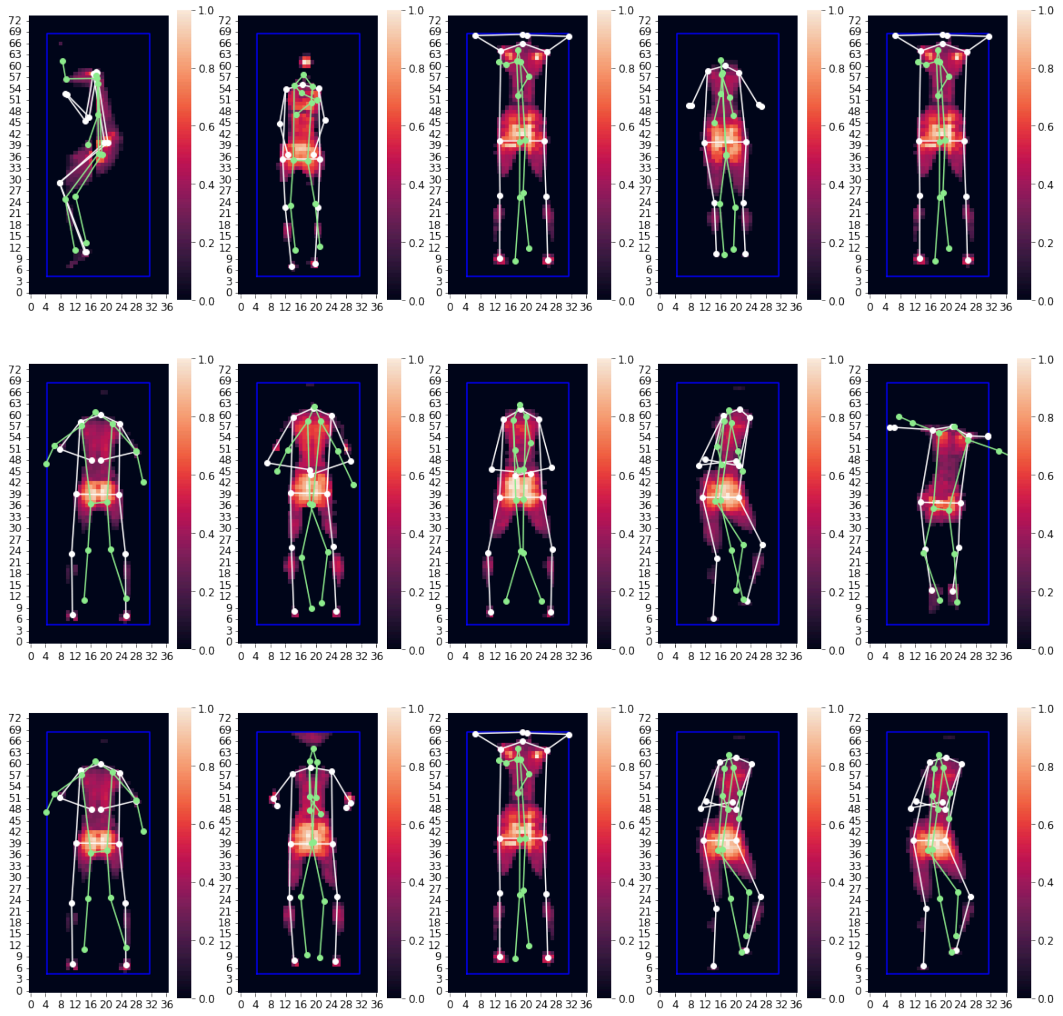


Figure A.5.: Exemplary results of the direct regression approach for the *Softline* data. Ground truth in white, prediction in green

Table A.1.: Complete results for the experiments performed for the direct regression approach. ***M**ean **P**er **J**oint **P**osition **E**rror* (MPJPE) and MPJPE with **a**rms **o**mitted (MPJPE$_{ao}$) in centimeters, ***P**ercentage of **C**orrect **K**eypoints* (PCK) and ***P**ercentage of **C**orrect **P**arts* (PCP) in percent

	Bodies at Rest				SLP				Softline						
	MPJPE	MPJPE$_{ao}$	PCP	PCK	MPJPE	MPJPE$_{ao}$	PCP	PCK	MPJPE	MPJPE$_{ao}$	PCP	PCK	train loss	train time	epochs
Experiment I															
- SLP	31.72	26.63	14	12	10.37	7.73	71	71	15.72	11.35	67	51	8.21	02m 04s	84
- Bodies at Rest	10.24	6.75	77	73	18.88	14.92	38	33	16.76	14.23	48	36	9.93	17m 38s	94
- Combined	11.60	7.72	72	69	8.62	6.47	80	78	12.34	9.66	79	60	8.17	20m 20s	60
Experiment II: Original (O), Min-max normalization (MM), Std-mean-normalization (SM), Boolean (B), Histogram Equalization (HE), Sobel Filter (SF)															
- O	11.64	7.76	72	68	8.82	6.57	79	78	14.70	12.18	64	46	7.84	09m 04s	27
- MM	11.38	7.52	73	69	8.73	6.45	78	77	13.48	11.00	70	55	8.06	12m 19s	37
- SM	11.85	7.85	71	68	9.04	6.69	77	77	15.13	12.13	59	51	7.74	11m 10s	33
- B	11.71	7.74	72	68	8.77	6.54	79	78	12.74	9.86	76	59	7.55	10m 51s	32
- HE	11.44	7.64	73	69	8.84	6.61	79	77	14.87	12.51	61	48	7.97	11m 08s	33
- SF	12.12	8.18	71	67	9.01	6.69	78	77	13.18	11.28	80	60	8.50	09m 49s	29
- MM + O	11.40	7.60	73	69	8.70	6.57	79	78	14.51	11.21	72	51	7.83	11m 26s	32
- MM + SM	10.92	7.21	74	71	9.00	6.63	78	77	14.82	12.28	64	48	7.29	14m 54s	41
- MM + B	11.25	7.50	73	70	8.70	6.51	79	78	13.67	11.13	72	52	7.86	10m 05s	28
- MM + HE	11.39	7.59	73	69	8.78	6.55	79	78	15.13	13.12	59	42	8.12	10m 42s	30
- MM + SF	11.89	8.01	71	67	8.77	6.52	79	78	13.55	9.79	73	56	8.63	07m 57s	22
- MM + B + O	10.73	7.12	75	72	8.37	6.27	81	80	14.41	11.47	63	50	7.81	15m 34s	39
- MM + B + SM	10.60	6.97	75	72	8.59	6.37	80	79	14.54	12.02	62	48	7.30	13m 56s	35
- MM + B + HE	11.23	7.47	73	70	8.55	6.38	80	79	14.86	12.07	64	46	7.77	09m 38s	24
- MM + B + SF	11.35	7.55	73	69	8.50	6.39	80	79	13.21	9.98	74	57	8.19	10m 24s	26
Experiment III(A)															
Optuna Result	10.62	6.99	76	72	8.29	6.15	81	80	14.25	12.06	67	49	7.44	11m 22s	29
Squared loss	10.72	7.32	75	71	8.66	6.49	80	78	14.60	11.31	68	50	139.16	10m 54s	28
Arms omitted	6.19	6.19	93	88	5.97	5.97	97	91	10.13	10.13	93	72	4.24	11m 05s	28

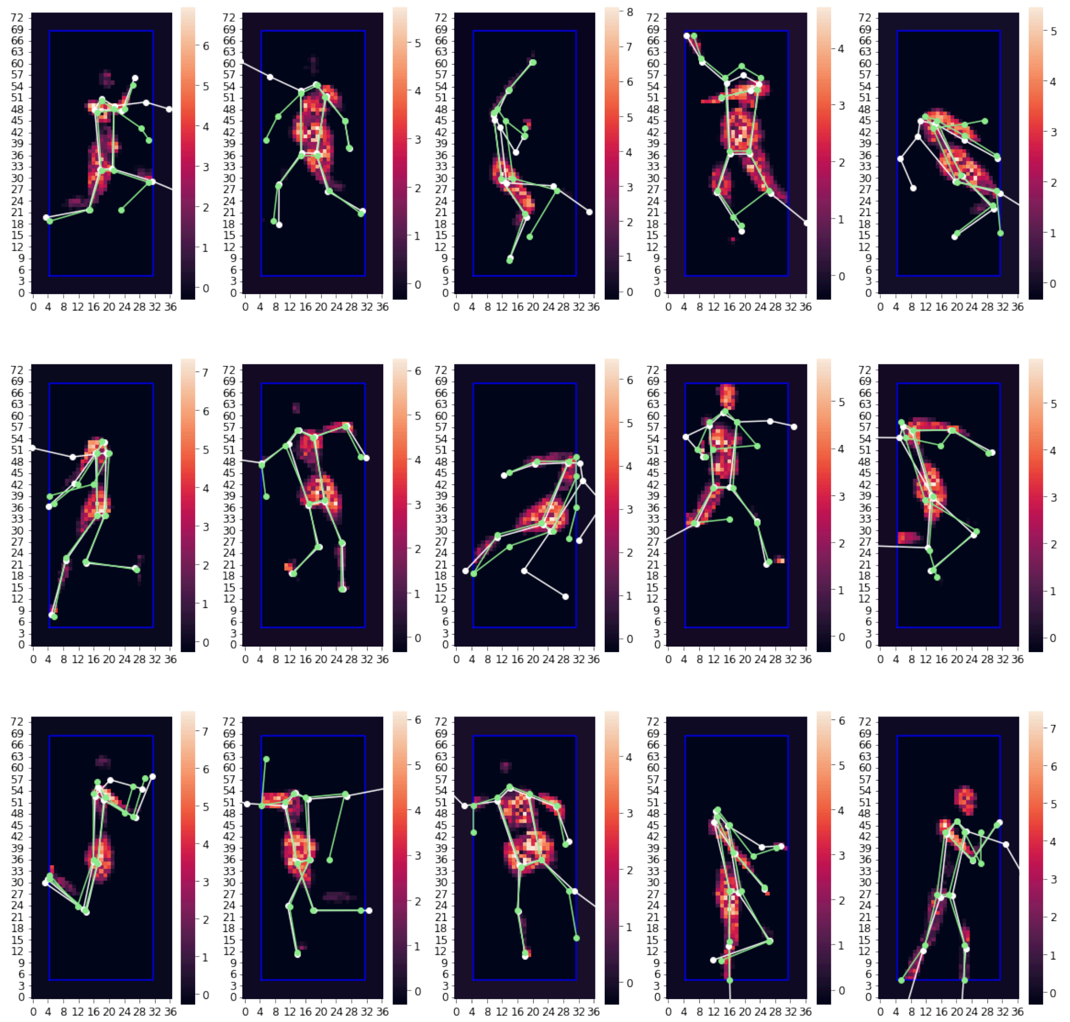


Figure A.6.: Exemplary results of the heatmap-based approach for the *Bodies at Rest* data. Ground truth in white, prediction in green

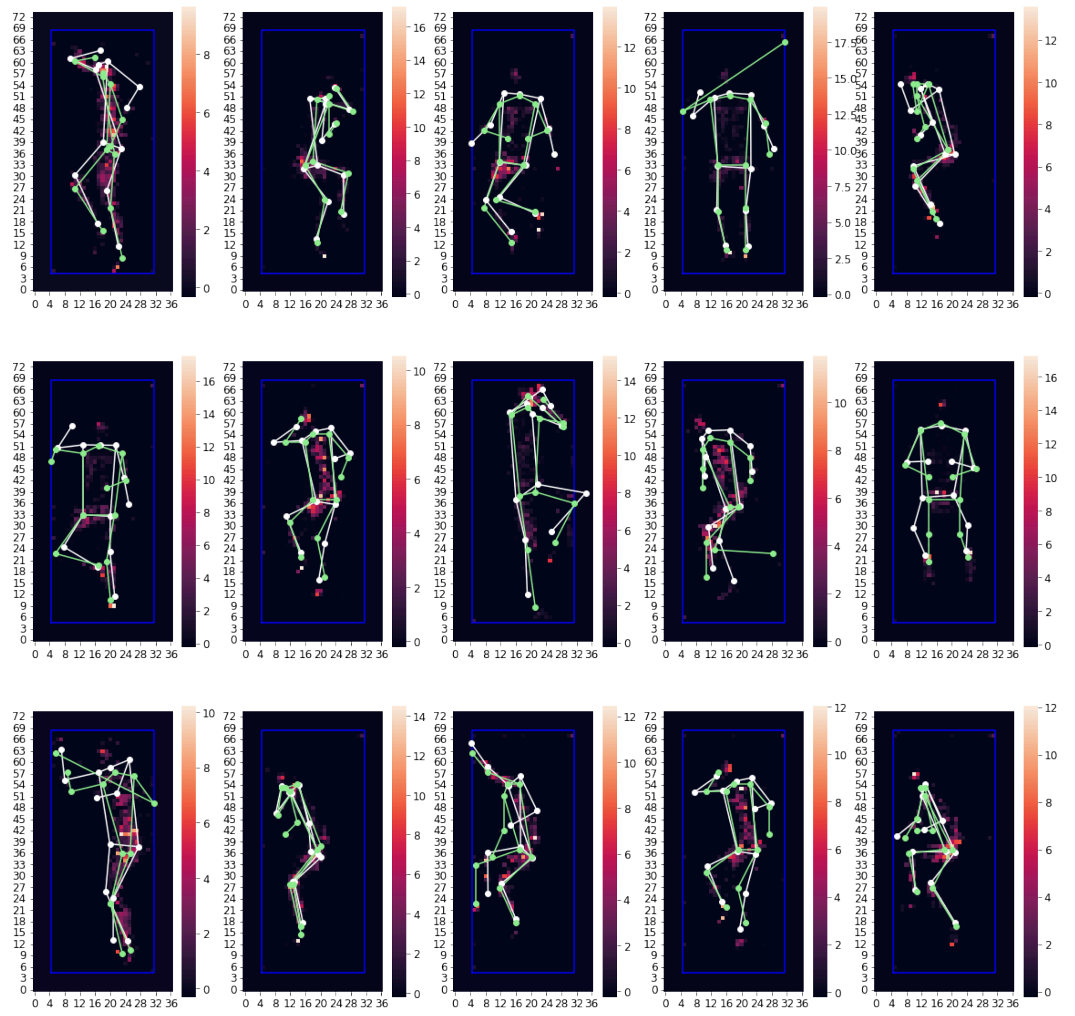


Figure A.7.: Exemplary results of the heatmap-based approach for the *SLP* data. Ground truth in white, prediction in green

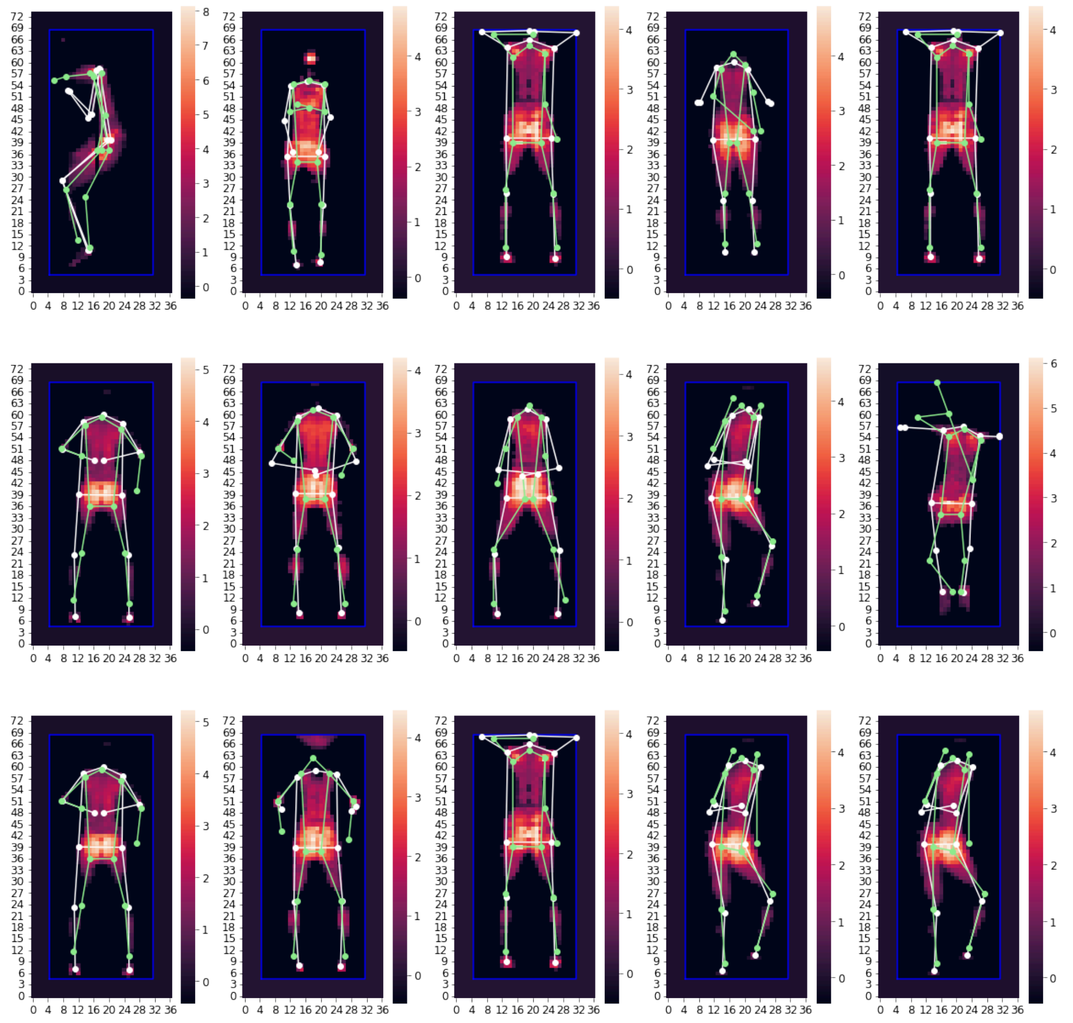


Figure A.8.: Exemplary results of the heatmap-based approach for the *Softline* data. Ground truth in white, prediction in green

Table A.2.: Complete results for the experiments performed for the heatmap-based approach. ***M**ean **P**er **J**oint **P**osition **E**rror* (MPJPE) and MPJPE with **a**rms **o**mitted (MPJPE$_{ao}$) in centimeters, ***P**ercentage of **C**orrect **K**eypoints* (PCK) and ***P**ercentage of **C**orrect **P**arts* (PCP) in percent

	Bodies at Rest				SLP				Softline						
	MPJPE	MPJPE$_{ao}$	PCP	PCK	MPJPE	MPJPE$_{ao}$	PCP	PCK	MPJPE	MPJPE$_{ao}$	PCP	PCK	train loss	train time	epochs
Experiment I															
- SLP	24.52	19.10	35	34	7.57	5.42	85	86	13.90	10.83	71	63	$1.00e-03$	00h 20m	34
- Bodies at Rest	8.10	4.42	83	84	23.79	18.64	32	31	16.25	10.92	64	56	5.30e−04	01h 14m	30
- Combined	10.14	5.68	79	81	9.70	6.97	79	80	12.62	8.29	75	70	5.41e−04	01h 03m	17
Experiment II: Original (O), Min-max normalization (MM), Std-mean-normalization (SM), Boolean (B), Histogram Equalization (HE), Sobel Filter (SF)															
- O	10.14	5.68	79	81	9.70	6.97	79	80	12.62	8.29	75	70	5.41e−04	01h 03m	17
- MM	9.89	5.43	79	82	8.46	6.23	83	84	11.51	8.08	80	71	3.75e−04	01h 07m	17
- SM	9.65	5.31	80	82	7.18	5.16	87	88	12.22	8.63	74	70	2.77e−04	01h 15m	19
- B	10.10	5.95	78	80	7.64	5.64	85	87	12.21	9.11	77	68	2.75e−04	01h 04m	17
- HE	9.70	5.40	79	82	7.69	5.62	85	86	13.00	9.75	68	62	3.79e−04	01h 02m	17
- SF	10.38	5.85	78	81	7.85	5.74	85	86	13.85	9.64	68	63	4.81e−04	01h 03m	17
- SM + O	9.64	5.50	80	82	7.19	5.27	86	88	11.67	8.30	73	67	3.48e−04	01h 04m	17
- SM + MM	9.64	5.40	80	82	7.43	5.31	86	87	10.86	7.49	79	71	2.97e−04	01h 12m	19
- SM + B	9.28	5.10	81	83	7.17	5.26	86	88	12.44	8.46	73	66	2.42e−04	01h 34m	25
- SM + HE	10.35	5.93	78	80	7.47	5.37	85	87	12.59	9.00	75	67	3.72e−04	01h 11m	19
- SM + SF	15.52	10.01	64	67	8.30	6.22	83	83	13.03	10.76	75	67	6.68e−04	00h 48m	13
- SM + B + O	9.80	5.41	80	82	7.22	5.23	86	88	10.66	7.66	79	74	2.93e−04	01h 20m	21
- SM + B + MM	9.40	5.24	81	83	7.06	5.23	87	88	11.03	7.64	77	69	2.02e−04	01h 31m	24
- SM + B + HE	9.09	5.10	81	83	6.99	5.16	87	89	11.48	7.50	77	73	2.09e−04	01h 32m	23
- SM + B + SF	13.19	7.99	71	74	8.04	6.11	84	84	13.36	10.05	69	65	6.42e−04	00h 45m	12
- SM + B + HE + O	10.59	6.25	77	80	7.49	5.55	85	87	11.21	9.17	77	68	3.49e−04	01h 01m	16
- SM + B + HE + MM	9.98	5.71	79	81	7.31	5.36	86	88	11.81	7.93	77	72	3.72e−04	01h 11m	18
- SM + B + HE + SF	10.52	6.06	78	80	7.13	5.32	86	88	10.35	7.71	81	73	2.91e−04	01h 06m	17
Up-Conv	11.54	7.14	75	77	7.51	5.50	86	87	12.20	8.93	76	69	3.40e−04	00h 59m	14
Arms omitted	5.30	5.30	93	92	5.19	5.19	98	95	7.78	7.78	96	84	1.34e−04	01h 20m	21

Table A.3.: Complete results for the experiments performed for the hybrid approach. ***M**ean **P**er **J**oint **P**osition **E**rror* (MPJPE) and MPJPE with **a**rms **o**mitted (MPJPE$_{ao}$) in centimeters, ***P**ercentage of **C**orrect **K**eypoints* (PCK) and ***P**ercentage of **C**orrect **P**arts* (PCP) in percent

	Bodies at Rest				SLP				Softline						
	MPJPE	MPJPE$_{ao}$	PCP	PCK	MPJPE	MPJPE$_{ao}$	PCP	PCK	MPJPE	MPJPE$_{ao}$	PCP	PCK	train loss	train time	epochs
U-Net freezed	10.00	6.78	79	76	8.31	6.29	82	82	12.89	8.74	73	66	2.92	32m 01s	15
U-Net unfreezed	7.68	4.70	84	84	7.28	5.39	85	86	11.80	7.60	71	70	2.46	44m 27s	11
20% droupout, U-Net freezed	8.64	5.55	83	81	7.52	5.67	85	85	12.29	8.42	76	66	4.29	33m 54s	16
20% droupout, U-Net unfreezed	8.53	5.50	83	81	7.49	5.61	85	85	11.92	7.82	75	68	4.66	43m 48s	11

Table A.4.: Mean and standard deviation of per-joint position errors in centimeters of the final models on the test datasets in centimeters

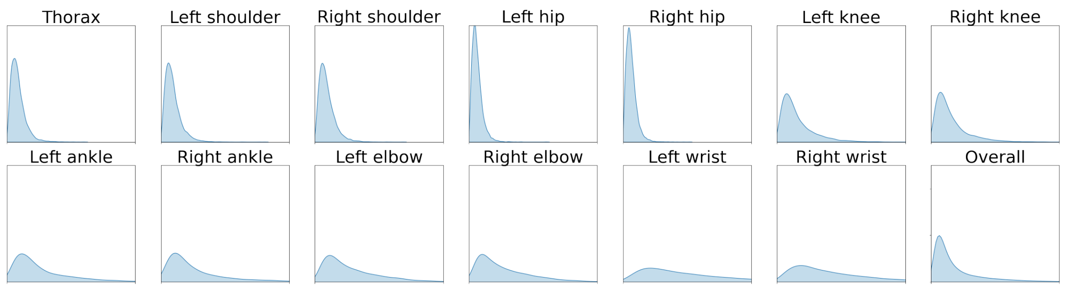
	Thorax	Left Shoulder	Right Shoulder	Left Hip	Right Hip	Left Knee	Right Knee	Left Ankle	Right Ankle	Left Elbow	Right Elbow	Left Wrist	Right Wrist	Mean	Mean$_{ao}$
Bodies at Rest															
Baseline	19.59 ±9.47	20.65 ±10.19	20.87 ±10.10	16.65 ±7.16	16.69 ±7.04	24.41 ±11.93	24.28 ±11.82	34.49 ±16.51	34.30 ±16.66	33.20 ±14.61	32.89 ±14.79	52.31 ±18.27	46.52 ±17.19	28.99 ±17.10	23.55 ±13.30
Direct regression	4.23 ±2.92	4.60 ±3.18	4.55 ±3.10	3.11 ±2.08	3.17 ±2.05	7.82 ±6.47	7.68 ±6.48	14.02 ±12.81	14.13 ±13.46	14.17 ±11.07	13.59 ±10.81	25.63 ±19.19	21.71 ±16.42	10.65 ±12.34	7.03 ±8.28
Heatmap-based	2.53 ±1.80	2.92 ±2.21	2.81 ±2.06	1.92 ±1.11	1.92 ±1.13	4.97 ±6.18	4.70 ±5.83	12.52 ±21.62	11.84 ±20.02	11.68 ±14.77	11.21 ±14.72	27.82 ±32.14	21.62 ±28.37	9.11 ±17.60	5.13 ±11.02
SLP															
Baseline	9.95 ±6.26	13.30 ±6.57	13.30 ±6.13	10.27 ±4.24	11.73 ±4.75	20.07 ±12.44	21.35 ±11.62	20.49 ±10.52	20.84 ±9.96	22.98 ±9.52	23.27 ±9.17	29.98 ±11.25	30.56 ±11.58	19.08 ±11.34	15.70 ±9.72
Direct regression	4.21 ±2.64	6.32 ±4.04	6.23 ±4.02	4.55 ±2.74	4.88 ±3.10	7.14 ±5.72	7.74 ±6.53	8.91 ±7.78	8.96 ±7.34	10.10 ±9.01	10.83 ±9.03	16.81 ±10.76	16.48 ±11.14	8.70 ±8.08	6.55 ±5.50
Heatmap-based	3.72 ±2.39	5.49 ±3.72	5.33 ±3.55	4.30 ±2.25	4.33 ±2.49	5.34 ±5.03	5.88 ±5.79	6.67 ±9.23	6.23 ±7.35	8.73 ±11.05	8.62 ±9.74	16.11 ±16.74	15.89 ±16.38	7.43 ±9.63	5.25 ±5.26
Softline															
Baseline	12.67 ±5.43	16.59 ±6.21	15.72 ±4.68	13.82 ±3.71	15.42 ±3.38	16.68 ±5.65	14.78 ±4.86	26.90 ±7.67	24.85 ±9.16	20.73 ±5.42	19.17 ±5.39	21.76 ±13.03	23.33 ±11.27	18.65 ±8.27	17.49 ±7.44
Direct regression	4.87 ±3.26	9.31 ±5.24	7.27 ±4.28	11.44 ±3.71	10.67 ±4.55	14.41 ±7.62	13.46 ±8.35	11.82 ±6.20	14.51 ±8.04	17.28 ±11.03	22.32 ±12.45	27.50 ±12.31	29.00 ±15.69	14.91 ±11.13	10.86 ±6.63
Heatmap-based	4.73 ±2.88	5.85 ±5.31	4.51 ±2.89	8.40 ±3.18	8.62 ±2.95	6.51 ±6.26	4.60 ±2.85	8.20 ±4.95	8.73 ±3.61	18.80 ±17.67	16.33 ±15.63	37.75 ±30.23	31.49 ±19.40	12.66 ±15.96	6.68 ±4.35

Table A.5.: *Percentage of Correct Parts* of the final models on the test dataset

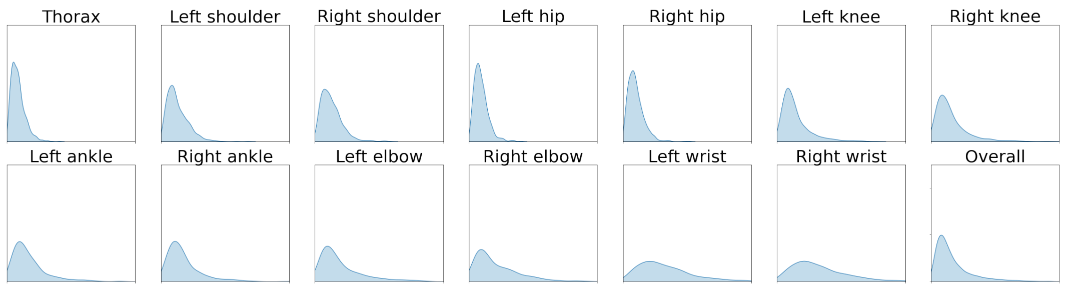
	Left upper leg	Right upper leg	Left lower leg	Right lower leg	Left upper arm	Right upper arm	Left lower arm	Right lower arm	Mean	Mean_{ao}
Bodies at Rest										
Baseline	40	41	18	18	7	7	1	0	17	21
Direct regression	99	99	83	84	75	76	46	41	75	80
Heatmap-based	99	99	87	87	81	82	57	58	81	86
SLP										
Baseline	68	58	46	38	21	30	3	2	33	40
Direct regression	98	98	93	90	82	80	43	42	78	83
Heatmap-based	99	98	95	95	86	87	58	57	84	89
Softline										
Baseline	100	96	48	64	12	44	4	8	47	60
Direct regression	96	88	96	88	72	52	4	8	63	71
Heatmap-based	100	100	100	100	72	76	20	28	74	84

Table A.6.: *Percentage of Correct Keypoints* of the final models on the test datasets

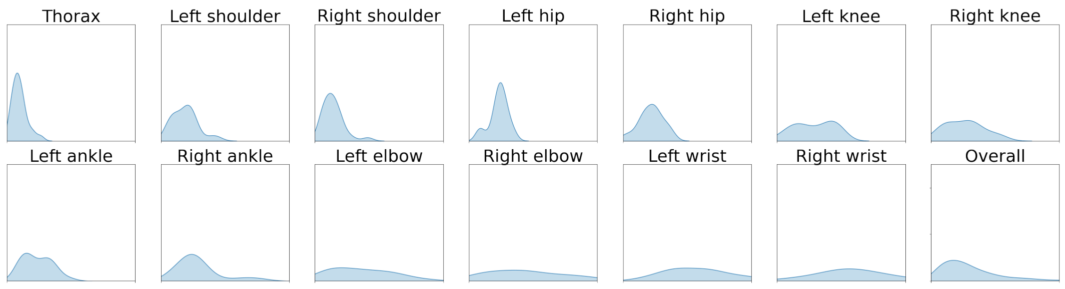
	Thorax	Left Shoulder	Right Shoulder	Left Hip	Right Hip	Left Knee	Right Knee	Left Ankle	Right Ankle	Left Elbow	Right Elbow	Left Wrist	Right Wrist	Mean	Mean_{ao}
Bodies at Rest															
Baseline	23	22	21	27	27	16	16	7	7	7	8	1	2	14	18
Direct regression	97	96	97	99	99	80	81	58	59	53	56	27	33	72	85
Heatmap-based	100	99	100	100	100	91	92	75	77	70	71	48	56	83	93
SLP															
Baseline	67	42	38	64	48	26	21	20	18	11	10	9	7	29	38
Direct regression	99	90	92	98	97	85	82	78	78	72	66	38	40	78	89
Heatmap-based	99	94	95	99	99	93	93	91	92	81	78	58	57	87	95
Softline															
Baseline	28	20	16	28	24	28	36	0	16	4	12	28	16	20	22
Direct regression	96	72	92	48	68	44	48	44	44	36	28	16	16	50	62
Heatmap-based	96	88	96	92	84	88	100	88	88	44	52	12	16	73	91



(a) *Bodies at Rest* test data

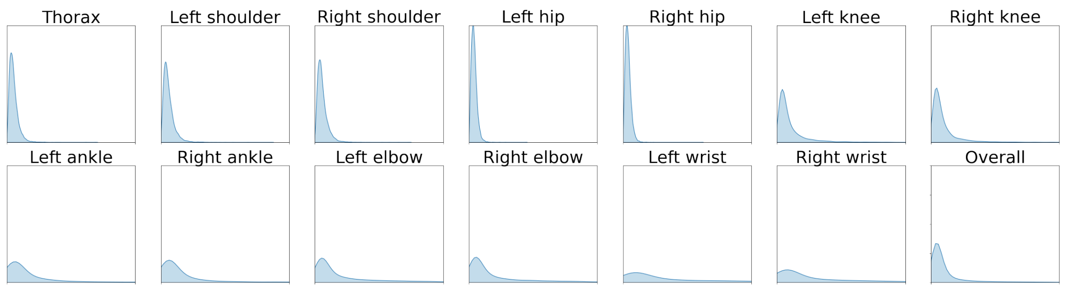


(b) *SLP* test data

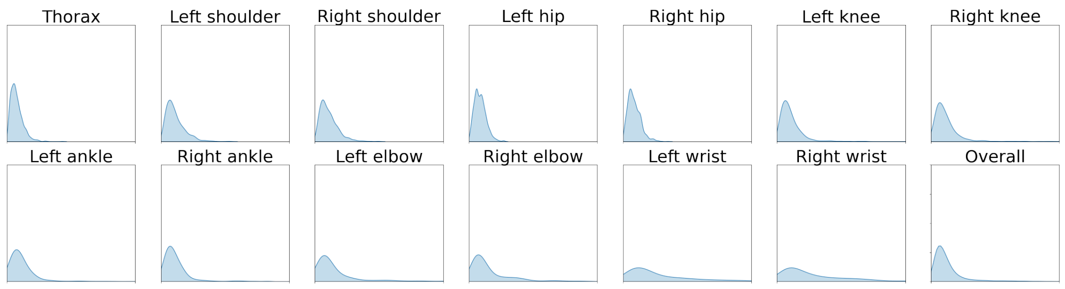


(c) *Softline* test data

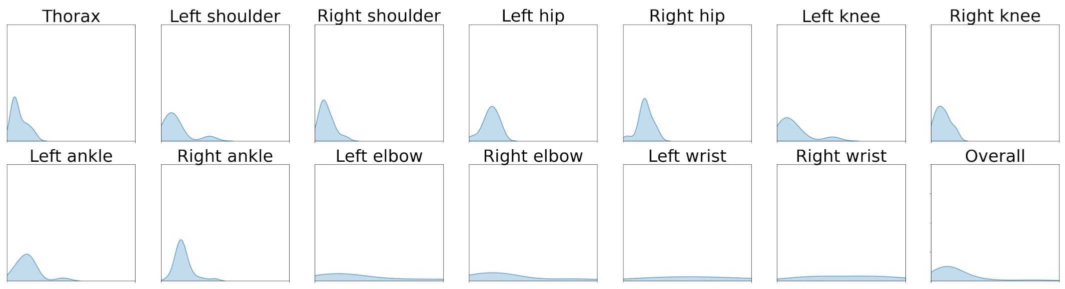
Figure A.9.: Density function of the distribution of the per-joint and overall errors for the final direct regression model. The x-axis represents the absolute error ranging from 0 to 50 centimeters, and the y-axis shows the associated value of the density function between 0 and 0.25. We can see that the model performs well for the thorax, hips and shoulders, but has difficulties when it comes to the extremities, especially the arms.



(a) *Bodies at Rest* test data



(b) *SLP* test data



(c) *Softline* test data

Figure A.10.: Density function of the distribution of the per-joint and overall errors for the final heatmap-based model. The x-axis represents the absolute error ranging from 0 to 50 centimeters, and the y-axis shows the associated value of the density function between 0 and 0.4. We can see that the model performs well for the thorax, hips and shoulders, but has difficulties when it comes to the extremities, especially the arms.

ISBN 978-3-86956-551-4
ISSN 1613-5652