UTRECHT UNIVERSITY
Faculty of Science
Department of Information and Computing Sciences
MSc Artificial Intelligence

SYNTHESISING 2D IMAGES OF ADULT-CHILD INTERACTION FOR HUMAN POSE ESTIMATION

A THESIS BY
Martijn Dekker
6013368

Project supervisor Prof. dr. Albert Salah
Daily supervisor M. Doyran MSc
Second examiner Dr. ir. R.W. Poppe

Abstract

In the last few years, Human pose estimators have gotten better at predicting the pose of people, especially adults. Pose estimators still struggle with occlusions and the performance on children has not improved as much either due to a lack of child-specific pose data and child bodies having different proportions. Adult-child interaction is even more difficult as it has lots of occlusions and people with vastly different body sizes. This is unfortunate as it could potentially be really helpful. Human pose estimation can be applied in many areas, e.g., in human-computer interaction, healthcare, or behavioural sciences.

In this research, I try to improve pose estimators' performance on adult-child interactions by synthesising data of said interaction. Authors of other studies have tried to solve the a of pose data by synthesising pose data using motion capture data. I tried a different approach, synthesising data by adjusting 3D human models' poses in Unity. The adult and child models are adjusted such that they interact. In total I created 40 different interaction scenes, which I used to create 40,571 2D images of the interaction. During the synthesising I diversified the aesthetics to create a varied set of images. Unity automatically added precise annotations, 2D and 3D keypoint locations a.o., such that I was able to use these images to finetune four state-of-the-art human pose estimators (HigherHRNet-W32, HigherHRNet-W48, HRNet-W48 and Stacked Hourglass).

To finetune the models, I combined a subset of the synthesised images (27,042 images) with COCO training data and trained on the combination. I evaluated their performance on a Youth images test set for which I annotated 520 challenging images of adult-child interaction. These images are challenging due to occlusions, self-occlusions, people blending in with the background and keypoints falling outside of the camera bounds. I found that the models' AP improved by 1.81, 0.72, 1.14 and 1.55 after finetuning while the AR performance improvements are larger, 2.52, 2.17, 1.35 and 1.25 respectively. These improvement show that motion capture data is not necessary for synthesising images that can improve pose estimators' performance on adult-child interaction data.

Table of Contents

1. Introduction

The challenge of locating anatomical keypoints, or joints (e.g. hip, wrist, knee), on humans in combination with combining the keypoints correctly into a pose make up the task of human pose estimation (Sun et al., 2019). Human pose estimation has received significant amounts of attention as part of the artificial intelligence research field. Human pose estimation can be used for many applications such as animation, human-computer interaction, action recognition and medical applications (Sun et al., 2019) (Groos et al., 2021). Human pose estimation based on 2D images can be quite challenging, even for human annotators, due to several factors such as low-resolution, motion blur and occlusion (Toshev and Szegedy, 2014) (Xiao et al., 2018). Additionally, it can be difficult for pose estimation models due to the uncountable number of possible poses that a human can adopt and the large variety in people's appearances.

Human pose estimation belongs to the computer vision field which is a research field with the task of making computers gain an understanding of visual input, images and videos, and perform a task based on the input. The first pose estimator models relied on locating body parts, e.g. a lower arm or an upper leg, to estimate a pose (Josyula and Ostadabbas, 2021). These classical approaches struggled with expressiveness, as they reason about parts of the body and do not focus on the whole body. Nevertheless, there have consistently arrived newer and better-performing human pose estimation models (Xiao et al., 2018). This is partly a consequence of the adoption of Convolutional Neural Networks, or CNNs, to tackle the challenge (Xiao et al., 2018) (Toshev and Szegedy, 2014). CNNs often use supervised learning to gain an understanding of the visual input and this applies especially to CNNs used for human pose estimation. Supervised learning is an approach of machine learning where a computer is tasked with learning a certain task based on labelled training data. In the case of human pose estimation, the model has to learn to predict the location of keypoints based on an input image. Initially, the model will make poor predictions but the model will correct its internal weights using the ground-truth locations (labels). After some time, if everything goes well, the model will have adjusted its internal weights to make predictions that are far closer to the ground-truth than it did at the start. This training process is then continued till the model can no longer improve its performance with the training data or after a pre-determined time.

The datasets used to compare these models have changed as well and are able to put more focus on more difficult images and poses. Some of the first datasets consisted of movie frames (Sapp and Taskar, 2013) or images of athletes (Johnson and Everingham, 2010). Newer datasets contain images with more people, thus increasing the likelihood of occlusions (Li et al., 2019).

Although these newer datasets contain many more images (going from 2,000 images (Johnson and Everingham, 2010) to 82,000 images (Lin et al., 2022)), even more data can still help improve a model's performance (Shorten and Khoshgoftaar, 2019). However, the images in these datasets have to be selected, checked and annotated by people to generate the ground-truth annotations. This process is rather laborious and for some datasets the creators relied on crowdsourcing the annotations, for example, by using Amazon Mechanical Turk. There, people unaffiliated with the creators can annotate image in exchange for a small monetary reward (Johnson and Everingham, 2010) (Johnson and Everingham, 2011) (Sapp and Taskar, 2013). This can alleviate the immense work but comes with its own issues. Using crowdsourcing means that there are many different annotators and the inter-annotator differences can be large. Even one annotator isn't perfectly consistent and thus, these crowdsourced annotations need to be adjusted (for example averaging annotations from multiple annotators) to make them more consistent.

In general, annotating new images for a dataset is a demanding task and thus increasing the number of datapoints is rather difficult. Fortunately there is an alternative to artificially increase the size of the training data. With data augmentation we can, in a simple and straightforward manner, alter the images in the dataset to create images that are still realistic but different enough from the original. Examples of data augmentation methods are rotating and flipping. Data augmentation can be especially useful when there is not a lot of data for the task at hand (Shorten and Khoshgoftaar, 2019). However, data augmentation cannot create miracles and biases in the original data will also be present in the augmented data (Shorten and Khoshgoftaar, 2019).

Currently, a large proportion of images in datasets depict adult humans and not children (Sciortino et al., 2017). As children are not just adults at a smaller scale, they tend to have different body proportions, this can be an issue for pose estimation models (Sciortino et al., 2017). Most research is also focused on the pose estimation of adults with only sporadically an article specifically looking at the pose estimation of children (Sciortino et al., 2017). This is unfortunate as pose estimation of children can be useful for early detection of medical conditions such as cerebral palsy, autism spectrum disorder and Rett syndrome (Groos et al., 2021) (Sciortino et al., 2017). There are also use cases that are part of behavioural sciences (Salah, 2021).

The detection of medical conditions or certain behaviour relies on visual tracking and certain movements in infants or children. These tasks perfectly suited for pose estimation models. Moreover, the applications that benefit from human pose estimation such as action-recognition and human-computer interaction will also benefit from accurate pose estimation for children. Creating a publicly available dataset with images of children can be problematic due to privacy concerns and using data augmentation cannot fill the gap in available children pose data. As mentioned before, data augmentation cannot resolve or correct biases in a dataset. Thus, there needs to be more pose data of children. Taking the

possible applications into consideration, this data should preferably contain images with both adults and children because once these applications are used in real-time they will encounter such data as well.
The synthesising of data could help close the data gap and has already been used for certain goals within the field of human pose estimation research. The synthesising of data allows us to obtain data with a large variety of poses and environments (Ludl et al., 2018) (Covre et al., 2019). Although this data is not a perfect replacement due to the synthetic humans looking less realistic than real ones, it can still help improve the performance of human pose estimation models on real data (Ludl et al., 2018) (Covre et al., 2019) (Wang et al., 2019).
One advantage of synthesising data is that the annotations can be generated by the computer software that renders the 3D human models and thus results in precise, accurate and consistent annotations. Moreover, it allows us to control certain aspects of the data and prevent certain biases. Biases in terms of skin colour, body shape or age can occur but, fortunately, these can also be prevented rather easily.
Most synthesising relies on motion capture data to adjust the poses of 3D human models. But there is no motion capture data of adult-child interactions. Using other motion capture data is not an option as the data is body-specific, meaning that using a model with a different body size would not result in good poses. Moreover, there is not enough motion capture data of close interactions to use.
Therefore, I will research if it is possible to synthesise pose data of adult-child interactions without the use of motion capture data. This will require me to research what the synthesised pose data should look like in terms of poses as well as aesthetic aspects such as clothing and models. For this data to be truly useful, training a pose estimator on this data should also improve that pose estimator's performance.

1.1 Research question

This thesis researches the synthesising of adult-child interaction data for human pose estimation without utilising motion capture data. The thesis will aim to answer the following research question:

> *Is it possible to synthesise adult-child interaction pose data to improve the performance of a human pose estimation model without using motion capture data?*

In order to answer this research question, three sub-questions need to be answered:

- **Question 1**: What type of poses ought to be present when synthesising adult-child

interaction pose in order to improve the performance?

- **Question 2**: How can this type of adult-child interaction pose data be synthesised without using motion capture data?
- **Question 3**: What data aspects, besides the poses, are important to improve a pose estimator's performance on adult-child interactions?

To answer the first question I will research what type of images depicting adult-child interactions cause problems. I will look at poses of all the people and investigate whether, e.g., lower performance is due to the people in the image having different heights, a rare pose of a single person or occlusions. For the second question I will research how to synthesise (adult-child interaction) data and what the absence of motion capture data requires from the software used to synthesise the image. Moreover, I will investigate what things need to be present in the annotation files. Lastly, to answer the third sub-question, I will research what data aspects, e.g, camera position and materials are important.

My contributions can be summarised as follows:

- I address the lack of public adult-child interaction pose data by synthesising 2D images without utilising motion capture data. In total I have created 40 different interaction scenes in Unity, which resulted in 40,571 with significant amounts of aesthetic diversity. This diversity was accomplished by randomising certain aspects such as clothing, clothing materials, lighting, camera viewpoint and backgrounds. The images were automatically and precisely annotated by Unity and consist of the 2D and 3D locations, the area and bounding box of the people in the scene.
- I annotated a challenging new test set of adult-child interaction consisting of 520 images of 130 interactions. This Youth images test set is challenging due to occlusions, self-occlusions, people blending in with the background and keypoints falling outside of the camera bounds.
- I finetuned four state-of-the-art pose estimator models (HigherHRNet-W32, HigherHRNet-W48, HRNet-W48 and Stacked Hourglass) by training on a combination of COCO training data and a subset of the synthesised images. The finetuning of the models resulted in higher performance on the Youth images test set. The AP improvements are 1.81, 0.72, 1.14, 1.15 while the AR improvements are 2.52, 2.17, 1.35 and 1.25 respectively.
- I show that it is not necessary to use motion capture data to synthesise pose data which can result in a performance improvement for state-of-the-art pose estimators after training.
- I show that, to achieve good performance, one should not only look at the training and training data, but also the test data. Simple changes to the test data, such as

splitting image files with multiple images, can improve the performance of the pose estimators significantly.

2. Related Work

The related works section discusses articles that are related to my research topic. First, I dive into some of the classical approaches to human pose estimation which did not yet rely on machine learning (Section 2.1). Approximately nine years ago the neural model called DeepPose was created by Toshev and Szegedy (2014) which is discussed in the second section (Section 2.2.2). DeepPose started the transition from the classical approaches towards neural models which use machine learning to achieve higher performances than the classical approaches at that time could. The first neural models used to predict the location directly but Tompson et al. (2014, 2015) showed that models predicting heatmaps could achieve higher performances. Some of the first neural models used refinement base models as these can combine local and global features.

With the increase in computing power and research that brought more powerful and more accurate models, there started to appear two different approaches that the models could use. The first approach, using top-down models that use a person detector to detect people after which the model predicts the keypoint locations, is discussed in the third section (Section 2.3). The second approach, bottom-up models, predict the keypoints without using a person detector. Instead, it tries to find all keypoints and predict their locations and then combines the keypoints into poses (Section 2.4). Most models using either approach predict a heatmap to predict the keypoint locations. This requires the correct location to be encoded into ground-truth heatmaps which can be used to teach the model and the predicted heatmaps have to be decoded to get the keypoint location. This encoding and decoding process is not easy and there are multiple strategies for doing this. As it turned out, different strategies are worse than others as they result in lower performance for the pose estimator. These strategies are explained in the fifth section (Section 2.5).

To get the performance of certain models, there are several different metrics which can be used. These performance metrics, which are discussed in Section 2.6, have also evolved to better account for the huge variation in possible poses. The metrics are used to compute the performance of an estimator which is trained and tested on one of the available datasets such as FLIC and COCO. I explain what these datasets entail in Section 2.7.

As these datasets only have a limited number of images to train the pose estimation models on, some method of artificially extending this would be useful. This is exactly what can be achieved with the use of data augmentation methods which I explain in Section2.8. These methods slightly alter the images with a simple strategy such that the result is an image that is still useful to train on and is different enough from the original such that the model can learn more from it. These methods are surely useful but also have their own limits because they cannot create as much novelty as just adding more images would.

Unfortunately, adding more images can be rather laborious as they also require a human to annotate the images. A possible alternative is having a computer synthesise images and letting the computer directly and precisely annotate these images. I discuss some articles that have used this approach to improve a model's performance in Section 2.9.
As most pose datasets have few images of children, models tend to struggle with estimating the pose of children. There has been some research done specifically on child pose estimation but models still have not breached the gap in performance between child pose estimation and adult pose estimation. This is unfortunate, as pose estimation can be useful for certain healthcare applications and behavioural science applications. The problems plaguing pose estimators with child pose estimation and the useful applications are discussed in the last section (Section 2.10).

2.1 Classical approaches to human pose estimation

The classical approaches to human pose estimation did not rely on deep learning as used by the current state-of-the-art models, rather they relied on other methods such as a pictorial structure framework (PSF) or a flexible mixture of parts (FMP) (Josyula and Ostadabbas, 2021). The pictorial structure framework was originally developed for recognising and representing objects and faces by Fischler and Elschlager (Fischler and Elschlager, 1973). Later on, this framework was adapted to work with human bodies as well. The idea behind the pictorial structure framework, for human pose estimation, is that human bodies are made up of rigid parts such as part of the limbs and the head, which are connected to one another via joints. These joints allow humans to move body parts (upper legs, lower arms etc.) around and therefore, the flexibility of body parts needs to be represented in the PSF as well. Using a separate part, which can be visualised as a rectangle, for each body part allows us to detect human body parts in various poses. In a similar manner to human bodies, where some body parts are connected, connections between parts are present in the representation of the body parts and keypoints in the image, often visualised as springs (Yang and Ramanan, 2011) (Fischler and Elschlager, 1973). The combination of parts, representing the body parts, and springs, representing the joints, allows for an immense degree of freedom in poses, which is a requirement for estimating human poses, as humans can take on numerous different poses as well.
Bourdev et al. (2009) introduced a new approach to human pose estimation where a model detects a poselet. The authors define a poselet as "a part of one's pose", so a poselet is similar to a part representing a body part in PSF, except that a poselet is able to cover multiple body parts as well. Additionally, a poselet must be clustered compactly in terms of configuration space and appearance space. Hence, a poselet can represent a body part (just like a part in PSF) or even a whole human, as well as everything in between. Identifying a lower leg might be difficult due to all the variation in proximity between body parts and

the variation in orientation possible while identifying "two crossed legs" might be easier. The poselets can be detected in an image using unsupervised clustering before they are classified (Holt et al., 2011). However, the downside of this approach is that the input images need to contain depth information and this means that it cannot easily be applied to commonly used input images (for human pose estimation).

Flexible mixture of parts uses the core idea behind PSF, a human body has both rigid (e.g. upper leg and lower arm) and flexible parts(joints), and was proposed by Y. Yang and D. Ramanan (2013) (Zhu et al., 2015). With FMP the pose is represented as a mixture of pictorial structures with small parts where, e.g., a lower arm is represented by two small parts. The model learns prior probabilities for what combinations of parts are likely to occur, as not all possibilities are likely. When looking at body parts, the two parts that represent the body part must have consistent orientations due to the rigidity of the body part. If the part representing the upper half of the lower arm is at a 45-degree angle, then the part representing the lower half of the lower arm should have a very similar angle.

Flexible mixture of parts models create a tree structure to describe the spatial and physical relationships between the parts. This is because a lower arm is connected to a hand, generally speaking, and an upper arm but not to any other body parts and this also means that an upper arm should be spatially near the lower arm that it is connected to. Some advantages of FMP are that it offers a compact way of representing human poses and allows for efficient inference (Zhu et al., 2015). One issue that FMP tries to tackle is the foreshortening of limbs, where a body part appears shorter in the image in comparison to other parts of the body. This is due to the angle between the aforementioned body part and the camera and can occur to various degrees.

Table 1 contains the performance of a few classical approaches on the Leeds Sports Pose

	Johnson and Everingham (2010)	Yang and Ramanan (2011)	Johnson and Everingham (2011)	Zhu et al. (2015)
PCP total	55.1	58.9	62.7	64.6

Table 1. Performance of classical approaches on the LSP dataset using the PCP metric, 0.5 radius, (see sections 2.7.1 and 2.6.1). The scores are from the article by Zhu et al. (2015).

dataset (see Section 2.7.1) using the PCP metric with a 0.5 radius (see Section 2.6.1). This dataset was proposed by Johnson and Everingham (2010, 2011) and was one of the first datasets used to compare the classical approaches described above.

2.2 First neural models

What unites the classical approaches, and in turn distinguishes them from the more recently learned features, is that they rely on man-made features such as HoG features, colour histograms and contours, which limit the models' ability to discriminate between keypoints

and generalise to new inputs (Tompson et al., 2015). In the second decade of the twenty-first century, there was a shift away from these man-made features toward human pose estimation models based on deep learning, with the DeepPose model, proposed by Toshev and Szegedy (2014), as one of the first[1] of its kind.

The DeepPose model is based on the AlexNet model as described by Krizhevsky et al., which is a slightly bigger model (Krizhevsky et al., 2012). The model is considered holistic which allows for predicting the location of joints that are not visible in the image. A holistic model looks at the whole input and not just a part of it in order to predict the task at hand, in this case, the estimates of the joint locations are based on the whole input image. The DeepPose model consists of seven layers, five of which are convolutional and two fully connected, as can be seen in Figure 1. In between the convolutional layers, there are some pooling layers and a local response normalisation(LRN) layer (Krizhevsky et al., 2012) after the first two convolutional layers. Both the pooling layers and the LRN layers are parameter-free. The input layer has a predefined input size of 220×220 pixels. As the pose estimation is formulated as a regression problem towards the joints in the image, the model outputs the x and y coordinates directly. In total, the model has about 40 million parameters, which is significantly less than the AlexNet model (Krizhevsky et al., 2012) which has 60 million and has one convolutional layer more than DeepPose.

For roughly estimating the location of the joints this coarse view (a small input size) is necessary, but it is inadequate for precise positioning. While increasing the input size could circumvent this, it would result in many more parameters. Therefore, the authors propose to train a number of pose regressors, see Figure 1. The model first processes the full image and roughly estimates the location of each keypoint. Next, for each keypoint, the model processes a crop, centred on the previous rougher estimate, of the input to estimate the locations more precisely. This step is repeated a total of S stages. Each next processing stage will have a smaller field of view with a higher resolution of the original one, which can be thought of as each stage zooming in further to the predicted keypoint location. For each stage $s \in \{1,...,S\}$, the same model is used but the parameters differ.

DeepPose manages to achieve an average PCP score of 61.0 at a radius size of 0.5 (see

Figure 1. The architecture of the DeepPose model, only layers with parameters are visible (Toshev and Szegedy, 2014).

[1]The first one that I was able to find

Section 2.6.1 for PCP definition). This average score is the average PCP on the upper and lower legs and arms. DeepPose outperforms classical models such as the model from the article by Johnson and Everingham (2011). DeepPose also works well on the FLIC dataset (see Section 2.7.2 and is able to handle foreshortening well (Toshev and Szegedy, 2014).

2.2.1 The origin of heatmap utilisation

In the first article by Tompson et al. (2014) that is discussed here, the authors argue that although DeepPose (Toshev and Szegedy, 2014) is able to achieve state-of-the-art performance, DeepPose suffers from the regression of coordinates from input images as this kind of mapping is quite difficult. Tompson et al. wrote some of the first articles that showed the merit of heatmaps (Tompson et al., 2014) (Tompson et al., 2015). This heatmap describes the probability of the joint in question to be located for each pixel using intensity. If the intensity is high then the model estimates that the probability of the joint being located there is high. Some examples of heatmaps layed over a regular image can be found in Figure 2.

With DeepPose the input image is processed multiple times to end up with a more precise location of the joint. Tompson et al. note that this causes an inefficient direct regression which is difficult to learn the correct parameters for (Tompson et al., 2014). The authors suspected that a new method for describing the estimation of the joints' location could improve performance. They propose a hybrid architecture of a deep convolutional network in combination with a spatial model. The convolutional network, or detector, is used to localise the body parts and produces a heatmap per joint. The convolutional network uses a sliding window that slowly moves over the image. The network combines two branches with overlapping receptive fields, one with a size of 128×128 (down-sampled to 64×64) and one with 64×64. As a result of this sliding window approach, the network becomes invariant to translation. The spatial model's purpose is to reduce false positives. A false positive can occur when, e.g., the detector's predicted head and shoulder keypoints have an impossibly high distance between them. This spatial model will ensure anatomically correct predictions based on the detector's predicted locations of the keypoints.

In the second article (Tompson et al., 2015) they continue their work on creating an architecture that produces heatmaps which they started in 2014 (Tompson et al., 2014). The authors use a similar multi-resolution sliding window detector. Because the network, in both articles, has some pooling layers, the output heatmap has a lower resolution than the input image. The new model processes an image by first using the coarse heatmap model to produce a heatmap for a joint with a coarse location prediction. Then this coarse location is used to crop the image, centred at the coarse location, for the fine heatmap model to predict the final location for each joint. This works in a similar manner to the DeepPose model, but unlike the DeepPose model, Tompson et al. reuse the exact same model and weights for their next processing step through the detector.

The models from both articles (Tompson et al., 2014) (Tompson et al., 2015) outperform state-of-the-art models at that time such as DeepPose (Toshev and Szegedy, 2014), but the newer model (Tompson et al., 2015) shows the largest improvement. On the FLIC dataset (see Section 2.7.2) the performance improvement for high precision regions is 30.4, 29.9 and 34.0 percentage points for the shoulder, elbow and wrist respectively (achieving a PCK@0.05 score of 73.0, 57.1 and 60.4 with the 4x pooling model). For the MPII dataset, the PCKh@0.5 (see Section 2.6.3) is 82.0 when the scale is normalized and 66.0 when the scale is not normalized. This is an improvement of 37.5 and 21.5 percentage points respectively over other models, such as the model by Yang and Ramanan (2013). Although the model is not compared to many other state-of-the-art models, the performance increase is extensive. These results indicate the advantages of using heatmaps for location prediction rather than predicting the coordinates directly (Tompson et al., 2015).

2.2.2 Refinement based pose estimators

The neural models described above, DeepPose by Toshev and Szegedy (2014) and the models in the articles by Tompson et al. (2014,2015), started using refinement to improve the model's prediction to mitigate the restricted input size. The utilisation of such a refinement process was continued by some models as it was the best option at that time to achieve high performance without increasing the input resolution of the model. Similar to the article by Tompson et al. (2015) and DeepPose (Toshev and Szegedy, 2014), Carreira et al. (2016) process the input multiple times to adjust the first coarse estimation to a finer final estimation. Their model predicts a keypoint location directly, just like DeepPose did, as well as an estimation for the corresponding error. The model repeats this process for two steps (or "stages", as Toshev & Szegedy call them), three steps in total, to improve the performance; they found that an additional fourth step brings little additional performance. The authors call this process Iterative Error Feedback (IEF) because they use this predicted error to iteratively increase the accuracy of the keypoint predictions.
The IEF model does significantly improve upon older models on the MPII dataset (see Section 2.7.3) such as the model by Yang and Ramanan (2013) but when it is compared to the model by Tompson et al. (2015), the results are more similar if the ground-truth scale is provided during test time as IEF scores 81.3 versus 82.0 by Topmson et al. (2015). However, when the ground truth scale is absent during testing, IEF still manages to obtain a PCKh@0.5 score of 81.3 while the model by Tompson et al. (2015) only achieves a score of 66.0.

The article by Wei et al. (2016) introduces Convolutional Pose Machines (CPMs) which are able to learn long-range spatial dependencies. The architecture, which consists of convolutional networks, continuously refines the estimations for the locations of the joints, in the form of heatmaps, during the processing of an image. The authors argue

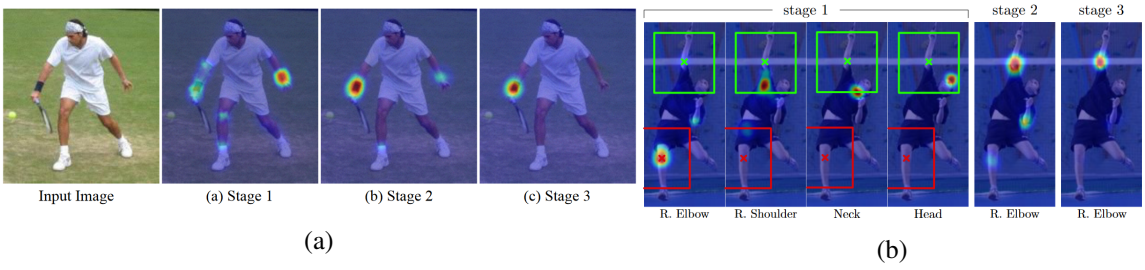


Figure 2. (a) The predictions of the 'right elbow' at different stages of a CPM, where the location gets refined at each stage (Wei et al., 2016). (b) The spatial information of easy-to-predict keypoints can help with the prediction of more difficult ones (Wei et al., 2016).

that although Carreira et al. (2016) iteratively refine the locations of the keypoints, their model will have reduced accuracy in high precision predictions due to their model directly predicting the coordinates (just like with DeepPose (Toshev and Szegedy, 2014)). In a similar manner to the model by Tompson et al. (2015), Wei et al. (2016) use a coarse estimate to create a fine and final estimate of the joint location, where the estimates get refined in each stage of the model. Where convolutional models can struggle with keypoints farther away from the core of the body, such as wrists and ankles, the CPM is able to use the heatmap for nearby keypoints to better predict the location of these more difficult keypoints. This allows the model to learn long-range spatial dependencies, as the location of, e.g., the wrist is dependent on the location of the elbow and shoulder.

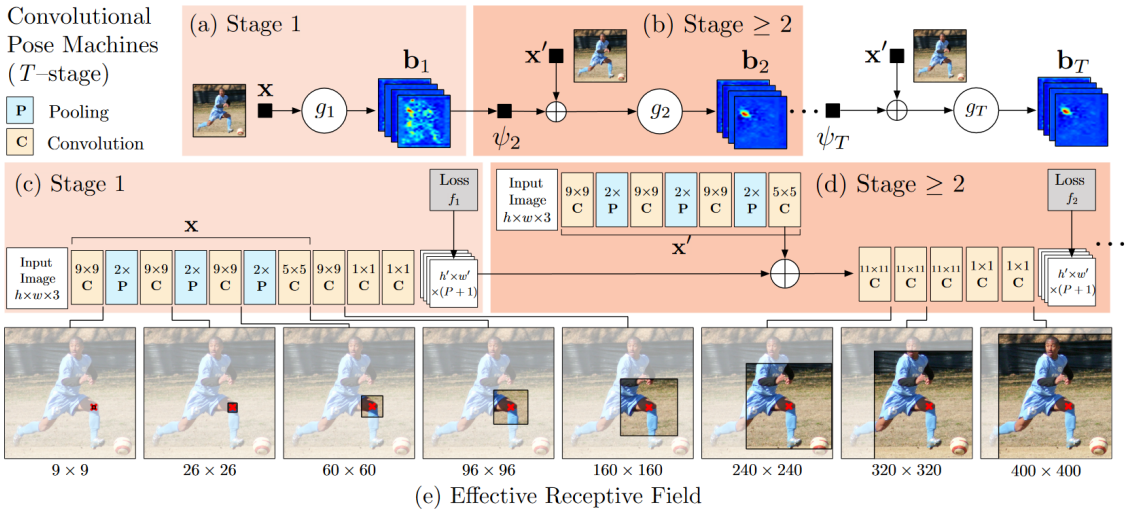
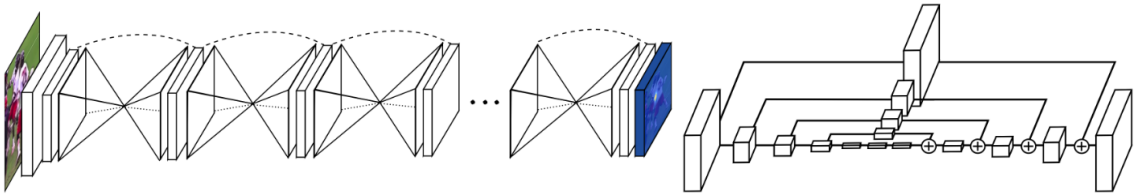


Figure 3. The CPM's architecture and receptive fields (Wei et al., 2016).

Before the input image can be fed to the CPM, it has to be resized to fit the input layer which has a resolution of 368×368 pixels. This is already significantly larger than DeepPose (Toshev and Szegedy, 2014) which has an input size of 220×220. The input image then passes through stage 1 (see Figure 3a and c) which predicts an initial heatmap mainly based on local information due to the small receptive field (see Figure 3c). In the subsequent stages the image is passed through several convolutional and pooling layers

(see x' in Figure 3d) and combined with the initially predicted heatmap from the previous stage. After this, a few more convolutional layers follow before a new heatmap is predicted. In the refinement stages (stage ≤ 2) the receptive field is increased to allow for more global information to be used. Having multiple stages also allows for computing intermediate losses, which can improve the model's performance.

CPM outperforms state-of-the-art models on the LSP, MPII and FLIC datasets (see Section 2.7 for information about the datasets) such as the model proposed by Tompson et al. (2015) and IEF by Carreira et al. (2016). The PCKh@0.5 score of CPM is 87.95 on the MPII dataset with a 10.8 percentage point increase for the ankle, which is one of the most challenging joints (Wei et al., 2016). This shows that CPM is able to learn long-range spatial dependencies quite well. On the FLIC dataset the PCK@0.05 performance for high precision regions, such as wrists and elbows, improves upon other models by 14.8 and 12.7 percentage points respectively. Although CPM outperforms all other models on the aforementioned datasets, it does struggle more with crowded images (Wei et al., 2016).



(a) (b)

Figure 4. (a) The Stacked Hourglass model (Newell et al., 2016). (b) A single hourglass module. Each box corresponds to a residual module, see Figure 5b (Newell et al., 2016).

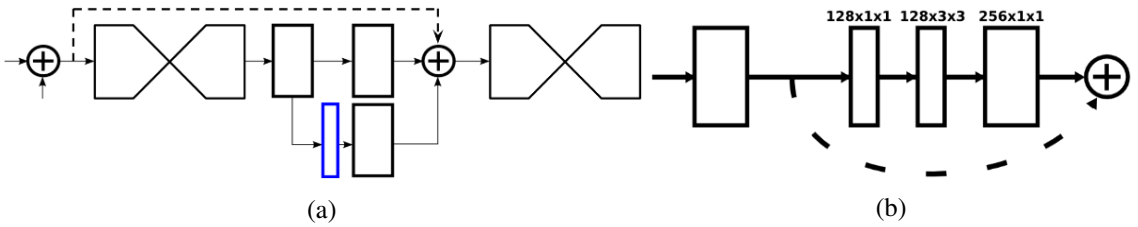
In 2016 Newell et al. introduced the Stacked Hourglass model (2016) which consists of a stack of hourglass modules (see Figure 4a). As local information is necessary for identifying joints, a model still needs an understanding of the pose as a whole via spatial information in the image. Models with a wider field of view, and often lower resolution, are better at extracting the spatial information, while smaller views, and higher resolution, are advantageous for identifying joints. Hence, the authors argue that a model must be able to process the image at different scales and make the model process the image at multiple resolutions in combination with skip layers to prevent spatial information loss (see Figure 4b).

At the lowest resolution, the Stacked Hourglass processes a 4×4 scaled representation of the input image. The hourglass module consists of a number of convolutional and pooling layers to downsample the resolution followed by the upsampling of the resolution after reaching the lowest resolution. Just before each pooling layer, one branch applies more convolution and is not downsampled further to be combined with a down and then upsampled version later (the branches in the back of Figure 4b) using elementwise addition. As can be seen in Figure 4b, the hourglass module is symmetric which means that for

every downsampling layer there is a corresponding upsampling layer that ensures that the input resolution is the same as the output resolution for the whole module. After reaching this highest resolution again, two 1×1 convolutional layers are applied (see Figure 5a). For the convolutional layers in the hourglass module, the authors experimented with either using one layer with a larger filter size or using a residual learning module (see Figure 5b), from He et al. (2016), and found that using the residual module improved performance. In the hourglass modules there are many residual models and standard convolutional layers, but none with a filter size larger than 3×3.

Stacking multiple hourglasses allows for refinement of the pose estimation, which is also used by the CPMs (Wei et al., 2016) and the model by Tompson et al. (2014; 2015). But just down- and upsampling the representation does not achieve the ability for refinement by itself, the model has to be able to compute a loss at intermediary stages. That is why the authors added an intermediary heatmap layer after each hourglass module (see Figure 5a, where the blue rectangle is the intermediary heatmap). The reason for putting these intermediary heatmaps after each hourglass module, as opposed to within the module, is that then some information will be 'missed'. If you put it in the first part, before it reached the lowest resolution, then the prediction will miss out on more global spatial information. On the other hand, if you put it in the second part, in the upsampling part but before it reached the original size, then the prediction will not be able to use all the local information and thus not be as precise as it can be. Therefore, to allow for refinement, there have to be multiple stages where the representation is down- and upsampled. Ergo, the model requires multiple hourglass modules (as can be seein in Figure 5a) to allow for the refinement.

After passing an hourglass module, the output is combined with the output of the previous hourglass module (see the dotted line in Figure 5a). The final Stacked Hourglass model has eight hourglass modules in total, where the weights within each hourglass model are not shared. Due to computing constraints, the authors added some initial layers before the hourglass modules to reduce the input size from 256×256 to 64×64, which is the heatmap resolution as well. After the final heatmap is generated, the authors use an encoder-decoder to go from the heatmap locations to the locations in the original image with a larger resolution. The Stacked Hourglass outperformed other state-of-the-art models on the MPII



(a) (b)

Figure 5. (a) Hourglass modules, 1×1 conv. layers (black rectangles) and a intermediary heatmap (blue) (Newell et al., 2016). (b) A Residual Module used by Newell et al., originally from He et al. (2016).

dataset as can be seen in Table 2. With a PCKh@0.5 of 90.9, it managed to significantly outperforms models such as the one by Tompson et al. (2015) which already saw vast improvement compared to classical approaches such as the one by Yang and Ramanan (2013).

	IEF (2016)	Tompson et al. (2015)	CPM (2016)	Stacked Hourglass (2016)
PCKh@0.5	81.3	82.0	88.0	90.9

Table 2. Performance of some of the first neural pose estimators on the MPII (see Section 2.7.3) dataset using the PCKh@0.5 (Newell et al., 2016).

2.3 Top-down pose estimators

The most recent human pose estimation models that can be considered state-of-the-art can generally be divided in two main groups, top-down and bottom-up models (Cheng et al., 2020). There are, of course, some exceptions such a models combining bottom-up and top-down approaches (Cheng et al., 2021). Top-down models use a human detector to detect people in an image and predict a tightly fitting bounding box around each detected person. After this step, the image is cropped to each bounding box and the resulting cropped images are fed to the pose estimators. The pose estimator will then predict the location of the keypoints one by one in this cropped image (Cheng et al., 2020). By using this bounding box it makes the assumption that there is only one of each keypoint and thus only predicts one location for each keypoint. The correctness of this assumption is not paramount as the pose estimator can still achieve a high performance if there are keypoints of another person in the bounding box. In general, top-down models achieve higher performances than bottom-up models except for crowded scenes.

2.3.1 R-CNN models

The R-CNN, regions with CNN features, model was proposed by Girshick et al. (2014) in 2014 and was developed for object detection. When the R-CNN model was developed it had state-of-the-art performance for object detection but lacks in speed. The model first extracts 2,000 region proposals (bounding boxes) from the image using selective search, which are then processed through the Convolutional Neural Network (CNN). The CNN, based on AlexNet (Krizhevsky et al., 2012), creates a feature vector that can be used for the classification and bounding box output.

Both the Fast R-CNN model by Girshick et al. (2015) and the Faster R-CNN by Ren et al. (2015) are inspired by the original R-CNN model. The Fast R-CNN is able to increase performance and significantly increase the speed simultaneously. Compared to R-CNN, Fast R-CNN is nine times faster at training time and 213 times faster at testing time while

it also does not need any storage space, as it does not cache the features.

The Faster R-CNN (Ren et al., 2015) improves both the speed and performance over the Fast R-CNN network and is able to process between 5 and 17 images per second. The model no longer uses Selective Search to generate region proposals. Instead, they use a Region Proposal Network (RPN) to generate the proposals. He et al. (2017) propose the Mask R-CNN model which is based on Faster R-CNN. It adds a branch to the faster R-CNN framework in order to predict the mask of an object in combination with predicting the bounding box. The simplicity of mask R-CNN allows it to be used for other tasks as well, such as human pose estimation.

The Faster R-CNN framework predicts a class label and bounding box for each object that it detects. For each detected object the Mask R-CNN also predicts a (binary) mask for each class of the Faster R-CNN. In order to adapt it to human pose estimation, they trained the model to predict a mask for each keypoint (e.g. right elbow) that indicates the location of the keypoint. In these masks, only a single pixel is supposed to have a value of 1 (the keypoint) and they have a resolution of 56×56. Overall, the model is able to adapt to various tasks, such as human pose estimation, while maintaining a speed of 5 images per second (similar to the low bound of Faster R-CNN) and achieving good performance as well.

The mask R-CNN model, trained on the COCO training set (see Section 2.7.4), achieves an AP (average precision, see Section 2.6.4) score of 63.1 on the COCO test-dev set, which is higher than the state-of-the-art models discussed in their article (He et al., 2017). When they use a different backbone for their model as well as use a few more updates, they manage to boost the mask R-CNN model's performance on the COCO minival set but they do not mention a performance improvement on the COCO test-dev dataset.

2.3.2 Cascaded Pyramid Network

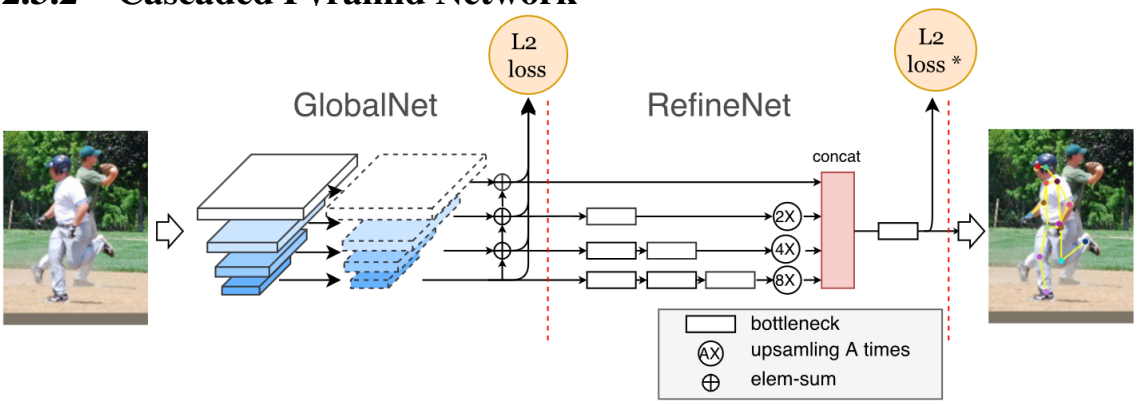


Figure 6. The architecture of Cascaded Pyramid Network (Chen et al., 2018).

The Cascaded Pyramid Network(CPN) was proposed by Chen et al. (2018) in 2017 and combines the global pyramid network, GlobalNet, with a pyramid refined network based on online hard keypoints mining, RefineNet. CPN, like any other top-down model, uses a

human detector to generate the bounding boxes of the people present in the image. With these bounding boxes, the image gets cropped to a single person, for each person in the image, and this cropped image is then fed to the model. As can be seen in Figure 6, the cropped image is first given to the GlobalNet network.

The GlobalNet structure was inspired by ResNet (He et al., 2016) and has four different layers in its pyramid structure (see Figure 6). These four layers have a decreasing resolution, from top to bottom, and capture different information. The top two layers capture more spatial information for localisation, while the bottom two layers capture semantic information for the joint recognition. Then, before the results of the convolutions in GlobalNet are summed element-wise, a 1×1 convolution is applied.

GlobalNet allows the model to find the easy keypoints but may fail at the harder occluded keypoints, but RefineNet will be able to help with these difficult keypoints. The RefineNet consists of some bottlenecks before it upsamples the information and then concatenates it before the final heatmap is generated. The deeper layers get more bottlenecks (see Figure 6). The authors found that the network tends to focus more on the simple keypoints as the training progresses. To balance this out, they use online hard keypoint mining to shift some of the focus to the more difficult keypoints instead.

CPN achieved state-of-the-art performance on the COCO dataset. CPN achieved an AP score of 72.1 on the COCO test-dev set, which is higher than the second best model used for comparison, the bottom-up model G-RMI (Papandreou et al., 2017), which achieved an AP of 64.9. The CPN+ model, which used ensembled models, achieved a higher AP score of 73.0, while G-RMI trained with extra data achieved an AP of 68.5. Even with the extra data for G-RMI, CPN without extra data achieves a 3.6 percentage point increase. Other models such as mask R-CNN (He et al., 2017) and the (old) OpenPose (Cao et al., 2017)[2] had significantly lower APs of 63.1 and 61.8 respectively.

2.3.3 Simple Baseline model

The simple baseline model, proposed by Xiao et al. (2018), has a very simple design, yet is able to achieve good performance. The idea was to research how well a very basic model could perform and took some inspiration from the Mask R-CNN model, where a branch was added to a pre-existing network. The model is constructed by appending a few deconvolutional layers to the backbone network, the ResNet model (He et al., 2016). Although the ResNet-50 network is the default choice, they experimented with two others, ResNet-101 and ResNet-152.

There are, in the standard model, three deconvolutional layers added at the end of the backbone network. These deconvolutional layers increase the resolution of the model's internal representation of the input image and allow the model to have a higher receptive

[2]Sometimes referred to as CMU-Pose

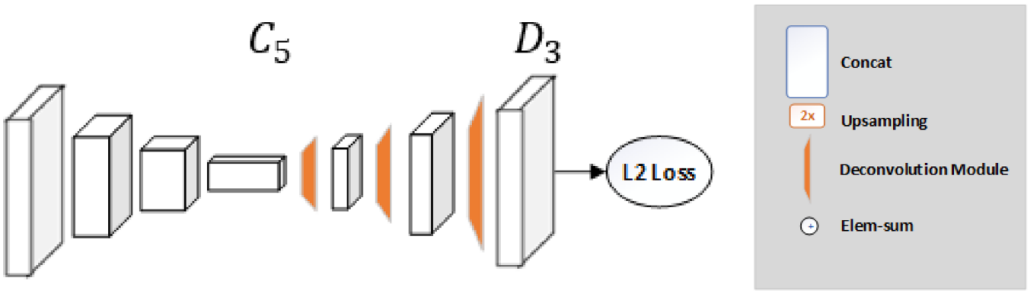


Figure 7. The architecture of the Simple Baseline model (Xiao et al., 2018).

field while still being able to output a relatively high-resolution prediction of the joints' locations. Unlike the Hourglass (Newell et al., 2016) and CPN (Chen et al., 2018) models, where up-sampling, to increase the feature map resolution, and convolutional parameters are kept separate, in the Simple Baseline model they are combined in the deconvolutional layers. These deconvolutional layers also do not use any connections that skip layers which makes them simpler while still being able to increase the resolution.
The authors looked into the effect of the input size and found that when the input size is decreased from 384x288 to 128x96, the AP on the COCO val2017 set decreases from 72.2 to 60.6. This is a significant drop in performance, but a drop in performance should not come as a surprise as having a larger input size means more information for the model. They also compared which ResNet model (He et al., 2016) would result in the highest AP on the COCO val2017 set, with an input size of 256x192. They found that using the ResNet-50 has the lowest performance with an AP of 70.4, while the larger ResNet-101 and ResNet-152 have APs of 71.4 and 72.0 respectively. As having a deeper network and a higher input size resulted in higher performance, this was also why they chose to use the ResNet-152 with 384x288 input size as their final model. This simple baseline version achieves an AP score of 73.7 on the COCO test-dev set and performs well compare to other top-down models as can be seen in Table 3. Meanwhile, the CPN (Chen et al., 2018) which uses the same input size and the ResNet-Inception as their backbone achieves an AP of 72.1.

2.3.4 High-Resolution Net (HRNet)

The High-Resolution Net (HRNet) by Sun et al. (2019) is an inventive, top-down method that is different from other models due to it maintaining a high-resolution representation throughout the whole network, while many other networks only recover a high-resolution from a lower resolution. The network has four parallel branches, one that keeps the original high resolution and three that have the resolution of the branch above it halved (in Figure 8

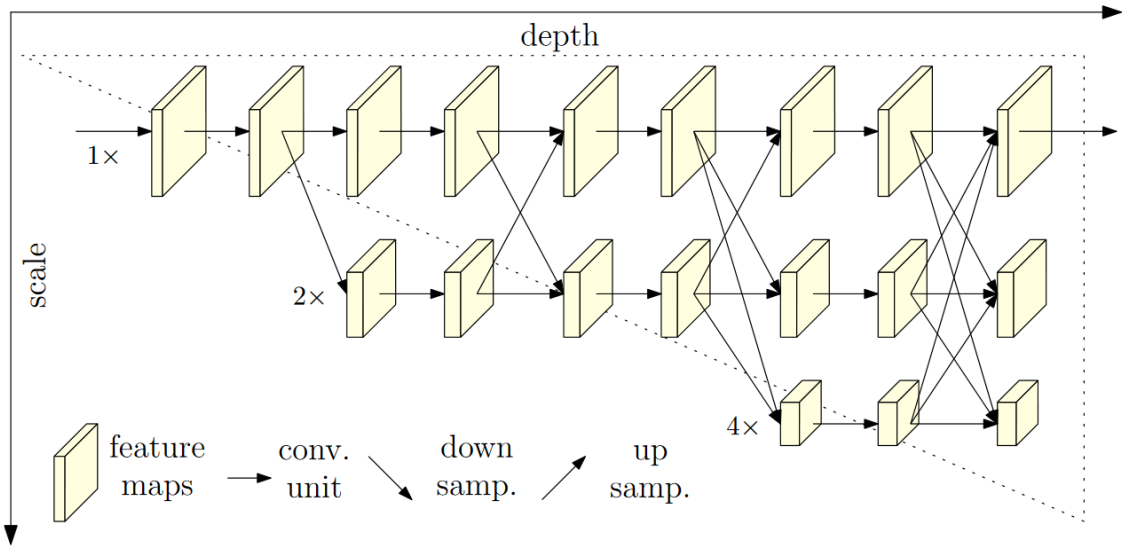


Figure 8. The architecture of the HRNet model (Sun et al., 2019).

only the three highest branches are shown). On top of the halving of the resolution, the width (the number of channels) is doubled.

Having one branch that does not decrease the resolution allows the model to output heatmaps with the same resolution as the input. In between, and at the end of these lower resolution branches, they are also connected to the high-resolution branch. This is done via a so-called exchange unit in order to allow the branches to receive information from other branches and combine the information.

They created two main versions, HRNet-W32 and HRNet-W48, where W32 and W48 stand for the width of the high-resolution branch in the last three stages. The HRNet-W32 is smaller in terms of parameters and is not able to achieve the same performance as the larger HRNet-W48 version, but is faster to train, roughly two times as fast.

When the authors tested it on the COCO validation set they found that HRNet-W32 and HRNet-W48 outperform Simple Baseline (Xiao et al., 2018) (with either ResNet-50, ResNet-101 or ResNet-152 as backbone) as well as other state-of-the-art models such as CPN (Chen et al., 2018) and 8-stage Stacked Hourglass (Newell et al., 2016). On an input size of 256x192, Simple Baseline achieves an AP of 70.4, 71.4 and 72.0 for the three backbones respectively while HRNet-W32 achieved 74.4 and HRNet-W48 achieved 75.1. On the COCO test-dev dataset with an input size of 384x288, the HRNet-W32 and HRNet-W48 receive AP scores of 74.9 and 75.5 respectively, as can be seen in Table 3. HRNet-W48 obtains higher scores but also has more parameters to train, 63.6 million versus 28.5 million for W32, as well as more GFLOPs, 32.9 versus 16.0. When the correlation between the number of GFLOPs (and input size) and AP between Simple Baseline and HRNet-W32 are compared, we see that when the number of GFLOPs is very close between the models, HRNet-W32 achieves higher AP.

	Mask R-CNN (2017)	CPN$^+$ (2018)	Simple baseline (2018)	HRNet-W32 (2019)	HRNet-W48 (2019)
AP	63.1	73.0	73.7	74.9	75.5

Table 3. Highest achieved performance of top-down models on the COCO test-dev set (see Section 2.7.4) using the AP (see Section 2.6.4). $^+$ indicates ensembled models.

2.4 Bottom-up pose estimators

Bottom-up models work differently in that they process a whole image at once. The model predicts the location of all keypoints per type (e.g. left knee, right hip etc.), next these keypoints need to be combined and connected to form a pose. The model has to predict which keypoints belong to the same person and thus should be grouped together (Cheng et al., 2020). Often the model predicts more poses than there are people present in the image and then uses the poses with the highest confidence as its final prediction. Most models use the top-down approach while the PersonLab, OpenPose and HigherHRNet models are some exceptions (Cheng et al., 2020). The main reason behind the preference for the top-down models is their performance, as they tend to outperform bottom-up models in all but crowded images. However, they do come with their own disadvantages. One such disadvantage is the need for either a human detector or the ground-truth bounding box. Bottom-up models do not need this detector and are generally faster as well.

2.4.1 PersonLab

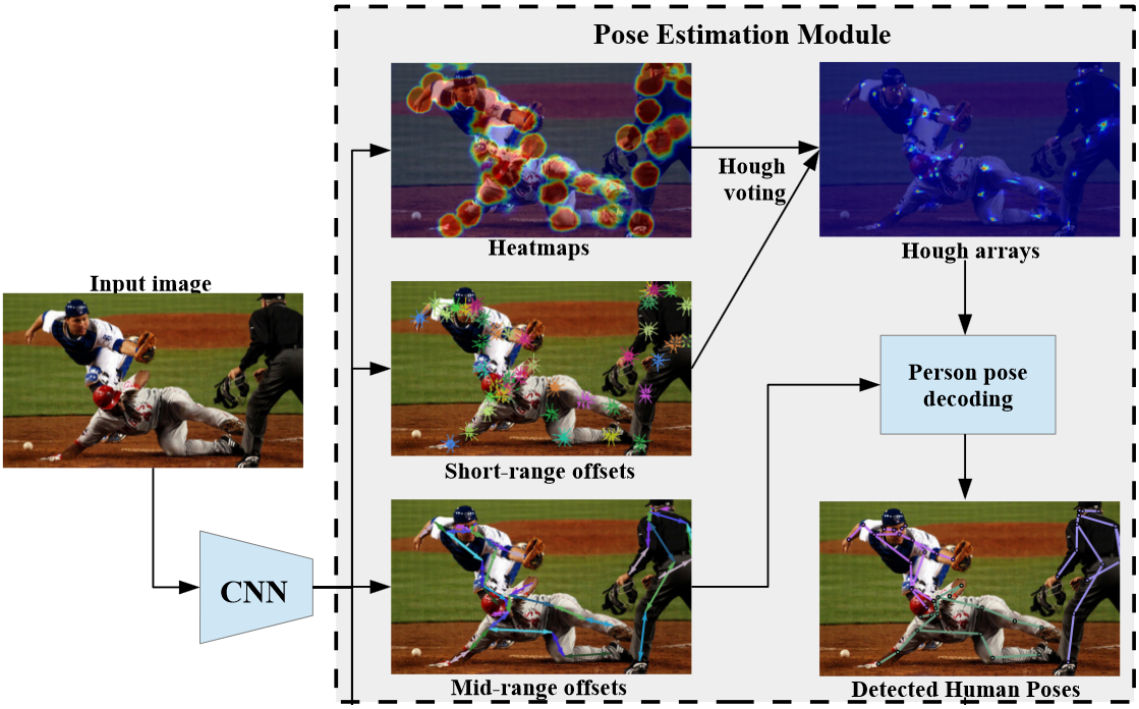


Figure 9. The architecture of the PersonLab model, where the CNN is the ResNet backbone (Papandreou et al., 2018).

The PersonLab model is a bottom-up model developed by Papandreou et al. (2018) for

both human pose estimation and instance segmentation. The model is based on the ResNet model (He et al., 2016), specifically the ResNet-101 and ResNet-152 models. Since these models were created for classification on the ImageNet (Krizhevsky et al., 2012), the last layer of the ResNet model is removed for PersonLab and the authors have added a 1×1 convolutional layer for each branch. The branches for the human pose estimation part serve the prediction of the heatmaps, the short-range offsets and the mid-range offsets (see Figure 9). For each estimated keypoint, the model also predicts a circle with a radius of 32 pixels around the keypoint location which acts as the Gaussian distribution. This radius is independent of the person instance's scale and was chosen to equally weigh all the people (instances) in the classification loss.

The reason for the short-range offsets' inclusion is to improve the keypoint localisation accuracy. Within each disk (see Figure 9, heatmaps branch), for each pixel a vector is generated such that it points from the pixel's location towards the closest keypoint. Next, the short-range offsets and heatmaps are aggregated via Hough voting to create Hough arrays or score maps. The pixels in the image all 'cast a vote' for each keypoint in the image where a pixel outside of the keypoint circles 'vote' zero and the pixels inside the circles 'vote' based on their location, intensity in the heatmap and the short-range offset. In the score maps, there will be local maxima that serve as keypoint candidates for constructing the poses.

The third branch, the mid-range offsets, are designed to couple different keypoints of the same person. The model computes two offset vectors per pair of keypoints that ought to be connected in the final pose, resulting in $2 \cdot (17 - 1) = 32$ vectors per person (17 indicates the number of keypoints per person). Person instances at large scales can have joints that are hundreds of pixels apart, which can cause some issues when they require a connection via the mid-range offsets. To tackle this issue the authors refine the mid-range offsets two times using the short-range offsets. They note that this offset helps PersonLab to achieve higher performance compared to other bottom-up models.

Lastly, the model uses a greedy algorithm to find the most likely connection between different keypoints based on the mid-range offsets and the Hough arrays (see Figure 9). The authors trained PersonLab on the COCO training data (see Section 2.7.4) but they did extend the training data annotations. People at small scales in the images part of the COCO dataset are not annotated, but the authors note that they do need these annotations for the instance segmentation part of the model. Therefore, they annotated these people at small scales with a different top-down model and used its predictions as ground-truth annotation during the training of PersonLab.

When the authors tested PersonLab on the COCO test-dev set they found that PersonLab achieved an AP of 65.5 using the ResNet101 backbone (He et al., 2016) and 66.5 using the ResNet152 backbone. PersonLab manages to outperform the (old) OpenPose model (Cao et al., 2017) by 3.7 and 4.7 percentage points respectively. When they utilise multi-

scale, PersonLab achieves AP scores of 67.8 and 68.7 respectively and outperforms the Associative Embbedding (Newell et al., 2017) model by 2.3 and 3.2 points respectively.

2.4.2 OpenPose

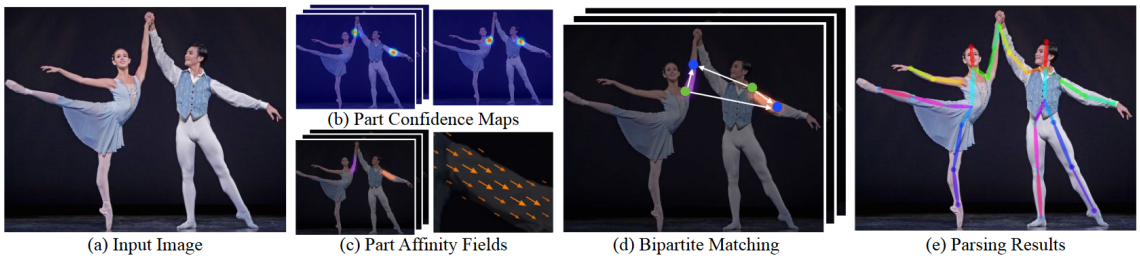


Figure 10. The pipeline of the OpenPose model (Cao et al., 2019).

In the article by Cao et al. (2019), the authors propose a new model, OpenPose (based on the original OpenPose model by Cao et al. (Cao et al., 2017)), which uses part affinity fields to estimate the pose in a bottom-up manner. Part affinity fields consist of 2D vector (flow) fields to encode the orientation and location of limbs and in turn the relationship between body parts of a number of people.

First, the image is processed by a CNN that generates a number of feature maps F (see Figure 11) that is used as input for the first stage. The model then generates a set (named L) of 2D vector fields of part affinity fields (see *c* in Figure 10) by the first part of the model. This first part is shown in blue in Figure 11 and generates one part affinity field per limb. Next, the model generates a set (named S) of 2D confidence maps (see *b* in Figure 10), one per part, by the the second part of the model, which is shown in beige in Figure 11. Lastly, the maps and the part affinity fields are parsed in a greedy manner to predict the final keypoints in the image (see *d* in Figure 10).

Inspired by the convolutional pose machine by (Wei et al., 2016), the OpenPose model refines the predictions over multiple stages (stage t $\in$ 1,...,T). After the first stage, the original feature maps F and predictions from the previous stage are concatenated and

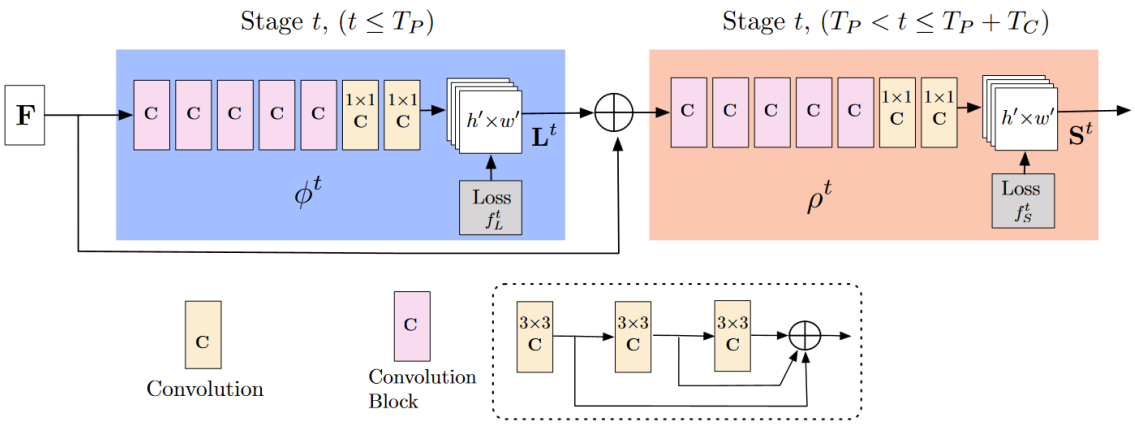


Figure 11. The architecture of the OpenPose model (Cao et al., 2019). The convolution block is defined as what is visible in the dotted rectangle at the bottom.

processed again to achieve refinement. After T_p stages. the final feature map and the part affinity fields are concatenated and forwarded to the confidence map part of the model (see the beige part in Figure 11) where the same refinement process is used for T_C stages. In the original OpenPose model by Cao et al. (2017) there were 7×7 convolutional layers instead of the convolution blocks (see Figure 11) but they found that the convolution blocks resulted in a speed increase of approximately 200% and an accuracy improvement of 7% as well.

Given a number of detected body parts, in an image with multiple people, it is not always so evident for the model to whom those parts belong. It becomes especially difficult if the people in the image are very close to one another or even create occlusion. This is where the part affinity fields come into play, as they encode the location and orientation, they can help with informing the model to whom a certain part of the image belongs, see *c* in Figure 10. The 2D vectors of the part affinity fields point from one side of the limb to the other side. Therefore it becomes easier to determine to which elbow a certain wrist belongs.

Lastly, the body parts have to be grouped to form a complete human pose, see *d* in Figure 10. The edges in the Figure are possible connections between the parts with each edge having a weight. This weight expresses the confidence that two nodes belong to the same person. This weight is based on the part affinity fields. Then, the model will match the bipartite graph, which is made up of the nodes and edges, and the model will greedily select the edges to maximise the overall weight for a limb, and that for all the limbs. During this maximisation process the model makes sure to not connect one elbow to multiple wrists, this is to ensure that the final poses are anatomically correct.

The authors show that they can achieve good performance using these part affinity fields. They achieve an AP score of 64.2 on the COCO test-dev set and, as the authors show, this does not outperform state-of-the-art models, neither top-down nor bottom-up. However, top-down models that are able to achieve higher performance than bottom-up methods also tend to be slower. OpenPose can process images faster than any other model tested or that provided runtime measurements, the authors were not able to achieve runtime measurements for all models. Additionally, OpenPose performs well on crowded images, while top-down methods generally struggle more with multi-person pose estimation.

2.4.3 HigherHRNet

The HigherHRNet is proposed by Cheng et al. (2020) and uses the HRNet as a backbone to good performance for bottom-up models. The authors adopted HRNet to work as a bottom-up method and extended the network by adding a few extra layers at the end of HRNet. They also add a deconvolution module that allows them to create feature maps that have a resolution two times larger than the input (see bottom right of Figure 12).

The first step of the model, the stem in Figure 12, divides the input image resolution by

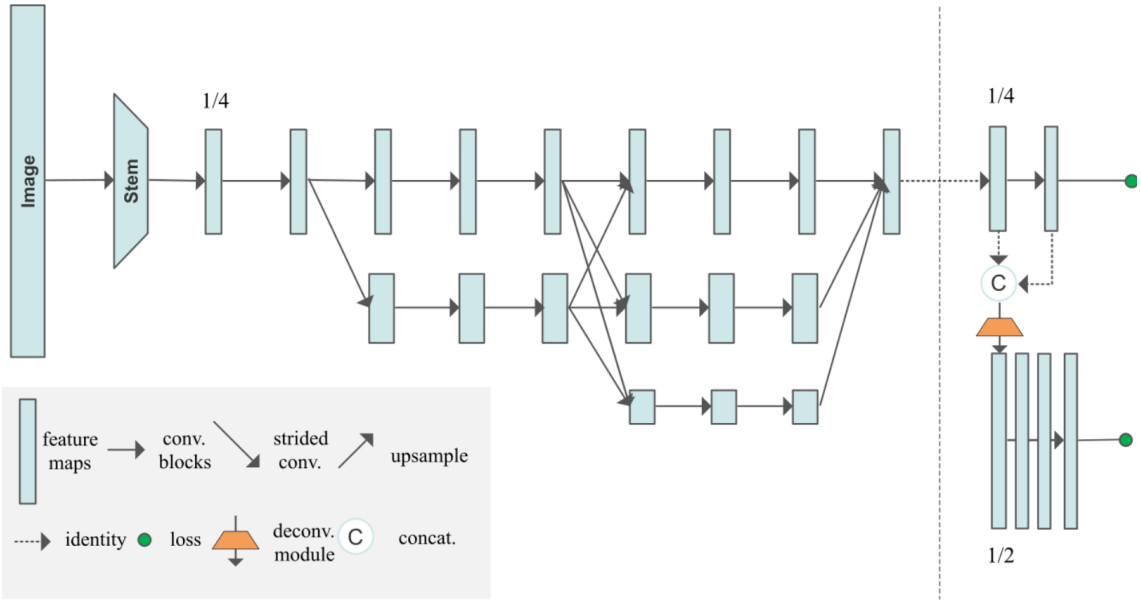


Figure 12. The architecture of the HigherHRNet model, having the HRNet as backbone on the left of the dotted vertical line (Cheng et al., 2020).

four and uses this resolution for the output heatmap as well. Just like with HRNet, this is possible thanks to the model maintaining the resolution throughout the whole network. In order to make the HRNet work as a bottom-up network, they added a 1x1 convolution after the backbone network (HRNet) as can be seen in Figure 12.

The resolution of the heatmap is especially important in the case of bottom-up networks because the humans can have varying sizes (due to top-down models cropping around a human and thus making sure that each human processed has roughly the same size, while bottom-up models do not crop and humans are processed at the size as they are present in the input image).

The authors argue that with bottom-up methods adding a Gaussian kernel to the heatmap is suboptimal, especially for people who appear smaller on the input image. Normally, with top-down models, one standard deviation is used for applying the Gaussian kernel to the heatmap for the input image at hand but this can blur the predicted location too much. Using a smaller standard deviation could be a solution, but they found that this makes optimisation more difficult. Instead, they opt to increase the heatmap's resolution using a deconvolution layer at the end of the model (the orange trapezoid in Figure 12), which was inspired by Xiao et al. (2018). This is what allows the model to have a heatmap with a resolution two times larger than the input image (after the stem).

It is possible to add multiple of these deconvolution layers to increase the heatmap resolution even further. However, the authors found that using only one such layer resulted in the best performance on the Coco dataset (see Section 2.7.4). On top of the increased heatmap resolution, they use heatmap aggregation for scale-aware pose estimation. Scale-aware pose estimation is useful as not all people on the images take up the same amount of pixels.

For the heatmap aggregation, they average heatmaps from all the different scales in the network.

The authors tested the performance on the COCO test-dev set and found that the W32 variant (using the HRNet-W32 as backbone) achieves an AP of 66.4, while the larger W48 variant achieves 68.4. HigherHRNet outperforms state-of-the-art bottom-up models with an AP of 70.5 using the HRNet-W48 as backbone with an input size of 640 and multi-scale testing, as can be seen in Table 4.

The HigherHRNet model also performs relatively well on the CrowdPose dataset (Li et al., 2019), see Section 2.7.5 for the details, with an AP of 65.9 and 67.6 for the HigherHRNet-W48 variant using single and multi-scale testing respectively. They outperform Mask-RCNN (He et al., 2017) and Simple Baseline (Xiao et al., 2018), which have APs of 57.2 and 60.8 respectively. HigherHRNet also outperforms the model by Li et al. (2019), by 1.6 percentage points for the multi-scale variant, who are the creators of the CrowdPose dataset. Cheng et al. mention that due to the crowded images of CrowdPose, top-down models no longer have an advantage over bottom-up methods.

	OpenPose (2017)	OpenPose (2019)	G-RMI (2017)	Associative Embedding* (2017)	Personlab* (2018)	HigherHRNet-W48* (2020)
AP	61.8	64.2	64.9	65.5	68.7	70.5

Table 4. Highest achieved (single-scale) performance of bottom-up models on the COCO test-dev set (see Section 2.7.4) using the AP (see Section 2.6.4). * indicates using multi-scale.

2.5 Model extensions and improvements

Based on the modern models for human pose estimation there have been some articles that look into improving the performance of one or multiple modern models, mostly top-down models, such as Simple Baseline and HRNet. These articles do not necessarily propose new models, instead, they propose manners for improving the performance of other models or try to resolve issues that these modern models encounter.

2.5.1 Distribution-aware coordinate representation of keypoints

In the article by Zhang et al. (2020), the authors propose the distribution-aware coordinate representation of keypoints (DARK) to improve the performance of state-of-the-art models by addressing the issue regarding inaccurate encoding and decoding. The motivation for addressing this in particular, instead of focusing on creating a new and better-performing model, is the potential improvements that resolving the issue would bring. The authors mention that resolving the issue can in fact improve the performance of a model more than a completely new model potentially could. The issue is that when ground-truth heatmaps are created for top-down models, the conversion of keypoint location to heatmap and from heatmap to keypoint location are done inaccurately and this negatively affects the

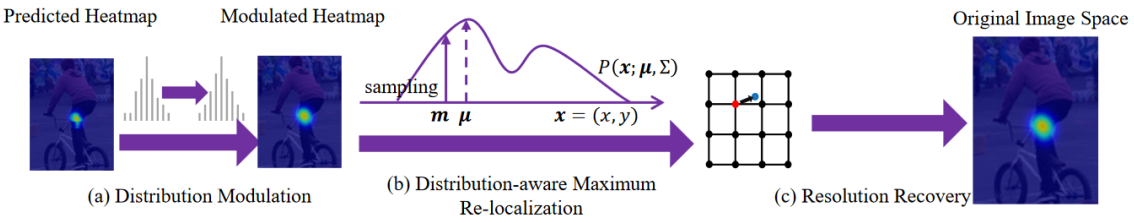


Figure 13. Overview of the DARK decoding process (Zhang et al., 2020).

performance of the model. These inaccurate encoding and decoding conversions have a larger impact when the input resolution of the model is smaller. The current state-of-the-art methods do try to mitigate the impact of this issue by shifting the heatmap location after decoding slightly.

The authors propose DARK which has two key components: (1) efficient and accurate coordinate decoding using Taylor-expansion and (2) unbiased encoding that is sub-pixel centred. DARK can be easily added to existing models without the need for algorithmic modifications.

Decoding

Decoding, as the authors describe, is the conversion of the heatmap, for each individual joint, into a coordinate location in the original image space. For this task, one needs to locate the maximum point in the heatmap, the point with the highest intensity, which is rather straightforward. However, as heatmaps often have a different resolution and/or spatial size than the original image, this requires us to upsample the heatmap or correctly convert between coordinate systems. The original method for doing this is as follows: we have a heatmap h, the coordinate of the maximum pixel m and the coordinate of the second maximum pixel s, the predicted location is then:

$$p = m + 0.25 \frac{s - m}{||s - m||_2} \tag{2.1}$$

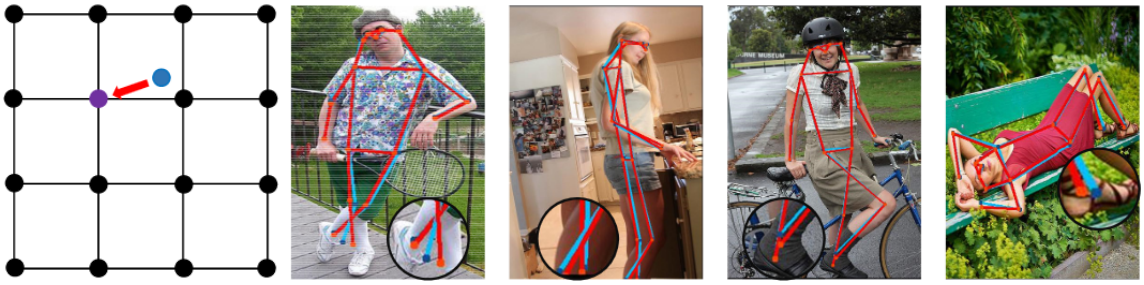
where $|| \cdot ||_2$ is the magnitude of the vector. When this equation is applied, the predicted location is shifted 0.25 pixels (a sub-pixel, as the authors call it) from the maximum m towards the second maximum s. The final prediction then becomes

$$\hat{p} = \lambda p \tag{2.2}$$

where λ is the resolution reduction ratio. The shifting achieved by equation 2.1 is to compensate for the inaccuracy due to the conversion between coordinate systems. The authors note that although this shifting helps increase performance and decrease inaccuracy, it is not perfect and thus they propose their own decoding method. The proposed method looks at the distribution of the intensity in the predicted heatmap and assumes that the distribution follows a Gaussian distribution and tries to find the maximum, μ, based on

that. Their method is computationally efficient as it only has a few computations it has to perform to find the prediction. The first step, modulating the distribution (see Figure 13), consists of creating a Gaussian kernel with the same variation as the training data, in order to smooth it. This smoothing is necessary as they found that there are usually multiple peaks around the maximum activation. This modulation ensures that the new heatmap has a smoother distribution, more resembling a true Gaussian distribution. Next, the maximum is relocated based on the distribution and this new maximum is then computed using Taylor-expansion. This relocating can be seen in Figure 13, between the second and third purple arrow, where the location is updated from the red dot to the blue one. Lastly, the model input resolution is recovered by converting to the input coordinate system.

Encoding



(a) (b)

Figure 14. (a) The standard encoding process' error (Zhang et al., 2020). (b) The difference in results between DARK(red) and HRNet(cyan) (Zhang et al., 2020).

Similar to the issue with decoding, the issue with encoding stems from the resolution reduction of the cropped image to fit the model's input resolution. This means that the ground-truth keypoint locations have to be converted as well, after which the ground-truth heatmaps can be generated. $g = (u, v)$ is the ground-truth keypoint coordinate and this is updated by the resolution reduction as follows:

$$g' = (u', v') = \frac{g}{\lambda} = (\frac{u}{\lambda}, \frac{v}{\lambda}) \tag{2.3}$$

where λ is the downsampling ratio. But for generating the Gaussian kernel this is often changed to:

$$g'' = (u'', v'') = R(g') = R(\frac{u}{\lambda}, \frac{v}{\lambda}) \tag{2.4}$$

where R is a rounding operation such as floor or ceiling. This rounding operation can result in inaccurate heatmaps and thus encoding errors. An example of an inaccurate location of a keypoint can be seen in Figure 14a, where the blue dot is the correct location and the purple dot is the location after equation 2.4 is applied; the red arrow indicates the translation vector. To solve this problem, the authors propose to abandon the usage of equation 2.4 and just use g' as in equation 2.3 for generating the Gaussian kernel.
They show that using the standard shifting (see equation 2.1), as opposed to no shifting,

can improve the AP performance of the HRNet-W32 model (Sun et al., 2019) by 5.7 percentage points on the COCO validation set, from 61.2 to 66.9. This means that the shifting is useful for human pose estimation. But the decoding method proposed by the authors can increase the AP performance by an additional 1.5 percentage points and thus, is a better decoding method. Their proposed unbiased encoding method is able to add an additional 2.3 percentage points, bringing the AP performance up to 70.7. This distribution-aware coordinate representation of keypoint (DARK) method is able to improve the performance of other models, on the COCO validation set, such as the Hourglass model (Newell et al., 2016) and the SimpleBaseline model (Xiao et al., 2018). When they tested using DARK in combination with the HRNet-W48 model, with an input size of 384×288, on the COCO test-dev set the improvement was smaller but still 0.7 percentage points. The improved performance and the same processing speed make DARK a good addition to any pose estimator.

2.5.2 Unbiased Data Processing

The Unbiased Data Processing (UDP) was proposed by Huang et al. (2020b) and aims to limit the reduced performance due to incorrect encoding and decoding. The authors argue that it is common for top-down human pose estimation models to use biased data processing which leads to reduced performance of these models. The data processing for these top-down methods consists of data transformation, data augmentation and encoding-decoding.

Data augmentation is a common method for increasing the number of images and their diversity as well but to a lesser extent. This method can help improve the performance and the robustness of the model. There are numerous types of data augmentation, but some commonly used ones for human pose estimation are random rotation, flipping, random scaling and half body (used by HRNet)(Wang et al., 2018).

The data transformation refers to the transformation of the keypoint location between different coordinate systems such as network input and output. This transformation occurs when cropping, rotating, resizing or flipping the image. The authors state that most top-down state-of-the-art models (such as CPN (Chen et al., 2018), HRNet (Sun et al., 2019) and Simple Baseline (Xiao et al., 2018)) use 'pixel' to measure the size of images, which can lead to reduced accuracy in measuring distances and converting locations. This is because 'pixel' is a discrete space while positioning requires continuous space and thus, the authors argue, if an image's size is (w_p,h_p) (where $_p$ denotes 'pixel') then its size in continuous space would be $w_p - 1$ by $h_p - 1$. This means that using 'pixel' as measurement can reduce the performance when using a flipping strategy in the inference.

Before training, top-down models rely on the transformation of the input image, as the image is cropped and resized based on bounding boxes of people present in the input

image. The image is resized to ($w_c + 1$,$h_c + 1$)(c for cropped and resized), after which it is again cropped, now by one pixel to get an image of (w_c,h_c). Then during inference, the output heatmap receives padding to go from (w_o,h_o)(o for output) to ($w_o + 1$,$h_o + 1$) and this updated heatmap is then used to find the corresponding location in the bounding box in the input image.

When the image is flipped, which is a common augmentation strategy, a small incorrect offset of $-\frac{s-1}{s}$ can occur, where s (stride factor) = $w_c/w_o = h_c/h_o$. For this reason, HRNet and Simple Baseline shift the result from flipped images by one pixel in the output coordinate system to reduce the prediction error, which reduces it to $|\frac{1}{2s}|$. For this correction, the error present will be smaller than without the shifting if $s > 1.5$, assuming that the output resolution will not be higher than the input image resolution. This means that as long as the input width/height is more than 1.5 times as large as the output width/height, the correction helps reduce the error. Hence, HRNet and Simple Baseline, as well as other methods that encounter this error, will see less impact of this error as the input image resolution is higher.

The authors propose to use 'unit length' to measure images instead and is defined as the distance between two adjacent pixels in a specific space (Huang et al., 2020b). Models would create the output heatmap as follows: $k_o = \frac{k_c}{s}$, with k_o and k_c being the coordinate of a keypoint in the cropped and output images respectively. Instead, these models should use $t = (w_c - 1)/(w_o - 1) = (h_c - 1)/(h_o - 1)$ and then $k_o = \frac{k_c}{t}$.

Encoding and decoding entail the transformation of the heatmap location to the ground-truth keypoint location, e.g, the location of a knee in the input image and the location of that knee in a smaller heatmap. This transformation is necessary because often the outputted heatmap has a lower resolution than the input image but the ground-truth keypoint locations are based on the input image. The advantage of this is that it does not require a predefined heatmap, and so any model, assuming the input layer's resolution is correct, can process the images. But this means that the model is required to transform the data.

This encoding and decoding was first proposed by Tompson et al. (2014) and is widely used by many models. The encoding is the process of transforming the ground-truth keypoint location to the heatmap with a Gaussian distribution centred at a corresponding coordinate, which is used for training the model. Decoding is the reverse, so computing the predicted location in the input image based on a heatmap produced by the model.

The location of the ground-truth keypoint is $k = (m, n)$ in the heatmap. Both Simple Baseline and HRNet first round the coordinates to obtain $k_q = (R(m), R(n))$ where R is the rounding operation. The location of the highest response in the outputted heatmap is $\hat{k}_q = (\hat{m}_q, \hat{n}_q)$. The authors state that the expected error of both $\hat{m}_q$ and $\hat{n}_q$ is $\frac{1}{4}$ 'unit length'. HRNet and Simple Baseline address this by shifting the predicted location by 0.25 pixels in the direction of the second-highest peak. This reduces the expected error to $\frac{1}{8}$ 'unit length' (with the variance going from $\frac{1}{48}$ to $\frac{1}{192} \approx 0.0052$) but is slightly increased when the slight

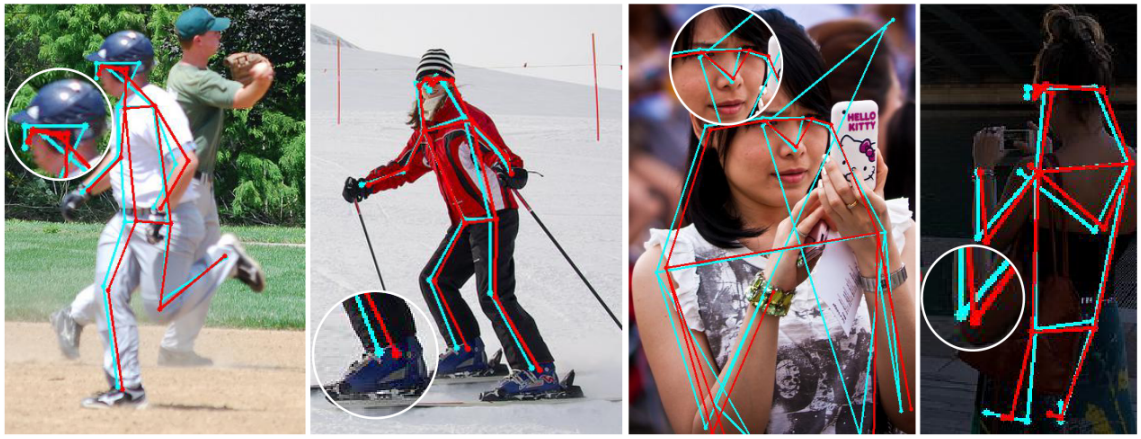


Figure 15. Differences in keypoint locations between UDP (red) and HRNet (cyan) processed at 128×96 (Wang et al., 2022).

error of $|\frac{1}{2s}|$ is taken into account and becomes $\frac{5}{32}$ (the variance becomes $\frac{37}{3072} \approx 0.012$). This shows that the decoding error is more severe than the data transformation error. Therefore, the authors propose a new encoding-decoding method with an expected error value of zero. The ground-truth keypoint $k = (m, n)$ is encoded into a heatmap H as:

$$H(x, y, k) = \begin{cases} 1, & \textbf{if } (x-m)^2 + (y-n)^2 < r \\ 0, & \textbf{otherwise} \end{cases} \tag{2.5}$$

where $H(x, y, k)$ indicates whether a pixel is marked as the keypoint(1) or not(0) for a point (x,y) given the keypoint's location of k and r being the distance threshold (Huang et al., 2020b). After this, a Gaussian kernel is applied to create the Gaussian distribution. As can be seen in Figure 15, just applying UDP can result in slightly different estimates. The images, processed at a low resolution of 128×96, are from the COCO dataset and show that at least in certain cases the UDP model can predict the keypoint locations more accurately (Wang et al., 2022).

Using UDP for the encoding and decoding can improve a model's performance without significantly increasing the number of parameters or the GFLOPS. The performance improvement on the COCO test-dev set for Simple Baseline (Xiao et al., 2018) and HRNet (Sun et al., 2019) can be seen in Table 5. The improvement is more significant for models with a smaller input size and models with a simpler backbone. The number of parameters increase with approximately 200,000 for all models in the Table, which is not a lot for models that have 34 million (Simple Baseline ResNet-50) or 28.5 million (HRNet-W32) parameters to begin with. The increase in GFLOPS due to UDP is approximately 0.1 for all models, which again is not a lot as the simplest versions of Simple Baseline and HRNet have 8.9 and 7.1 GFLOPS without UDP.

	Simple Baseline (2018)	Simple Baseline (2018)	Simple Baseline (2018)	Simple Baseline (2018)	HRNet (2019)	HRNet (2019)	HRNet (2019)	HRNet (2019)
Backbone	ResNet-50	ResNet-50	ResNet-152	ResNet-152	W32	W32	W48	W48
Input size	256×192	384×288	256×192	384×288	256×192	384×288	256×192	384×288
AP	70.2	71.3	71.9	73.8	73.5	74.9	74.3	75.5
AP with UDP	71.7	72.9	72.5	74.7	75.2	76.1	75.7	76.5

Table 5. Performance improvements achieved by using UDP on the COCO test-dev using the Simple Baseline model (Xiao et al., 2018), with the ResNet backbone (He et al., 2016) and the HRNet model (Sun et al., 2019).

UDP++

In 2020, Huang et al. (2020a) developed the UDP++ model based on the Unbiased Data Processing. The UDP++ model uses a version of the HRNet-W48 (Sun et al., 2019) model which they call HRNetW48plus, but I cannot find what the "plus" adds to the standard HRNet-W48 network. The model's input size is 384×288 and it uses the UDP for more accurate data transformation and encoding-decoding. Currently, it is the number one model on the COCO leaderboard under the name of 'XForwardAI' with AP of 80.8 on the COCO test-dev set. There have not been any additions to the leaderboard since UDP++ in august of 2020 (Lin et al., 2022).

UDP Pose Polarised Self Attention

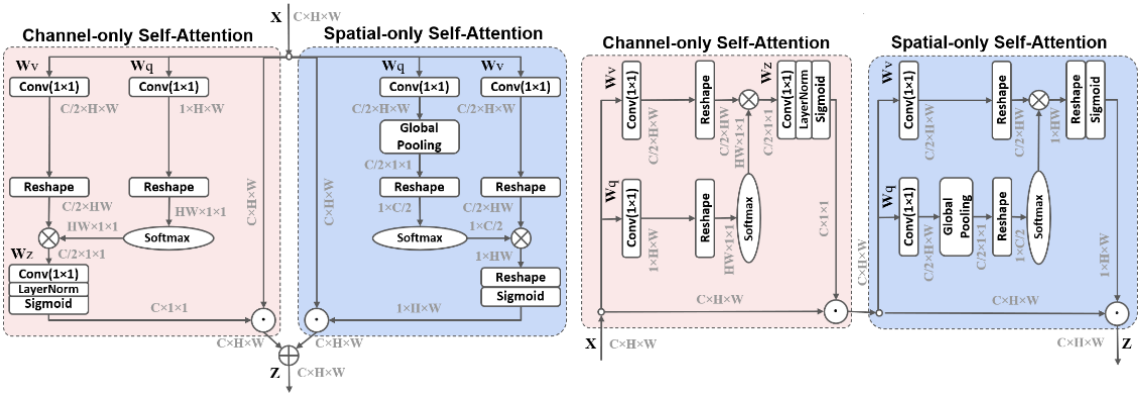


Figure 16. (a) The parallel Polarised Self Attention block (Liu et al., 2021). (b) The sequential Polarised Self Attention block (Liu et al., 2021).

Another model based on UDP is the UDP-Pose-PSA model which was proposed by Liu et al. (2021) in 2021 and adds Polarised Self Attention blocks. Attention mechanisms try to tackle the weakness of standard convolution. With the self-attention mechanism, an attention tensor is computed for each individual input tensor, after which it is used to weigh these input tensors. The idea behind self-attention is to grasp long-range interactions and has proven to be very successful at improving the performance of computer vision models. The authors propose the PSA block to bring this to human pose estimation and other

regression tasks such as instance segmentation. The Polarised is related to the filtering and is borrowed from photography. A self-attention block receives a tensor as its input to adjust the prominence of features, which the authors state as being similar to optical lenses in photography where they filter light. The polarisation in photography only allows light going in one direction to pass.

Their proposed PSA mechanism has two main features: (1) filtering in the form of collapsing the features in a certain direction while maintaining them in the opposite direction and (2) increasing the dynamic range by using Softmax normalisation at the smallest tensor inside the PSA block. They created two slightly different PSA blocks, the parallel one and the sequential one, and they can be seen in Figure 16. The channel-only attention refers to the channels of the input, which are the Red-Green-and-Blue channels in the case of human pose estimation images. The spatial only self-attention refers to the width and height of the image. Normally this type of self-attention would be expensive but they manage to keep the reduced speed within reasonable limits. This PSA block can be used with models a bottleneck or basic residual block, such as ResNet and HRNet which are very common backbones. The authors add it after the first 3×3 convolutional layer in each residual block.

Using PSA can increase the AP performance of some models up to 4 percentage points. When used in combination with the simple baseline (Xiao et al., 2018) and HRNet (Sun et al., 2019), their performance on the COCO val2017 set increases, with an input size of 384x288. For the simple baseline the AP performance increases by 4.3 and 3.7 points for the ResNet-50 (He et al., 2016) and ResNet-152 (He et al., 2016) models respectively. For the HRNet-W32 and HRNet-W48, the increase in performance is 2.9 and 2.6 points respectively. On the COCO test-dev set the performance of the UDP-Pose-ASP achieves the highest score and outperformed all other models with an AP of 79.5. The UDP-Pose model without PSA achieves an AP of 76.2, which indicates that using PSA can improve performance significantly. As for the increased processing time, the GFLOPS for the Simple Baseline (Xiao et al., 2018) variants, ResNet-50 and ResNet-152, increased by 0.9 and 2.2 respectively to 20.9 and 37.5. For the two HRNet variants, HRNet-W32 and HRNet-W48, the increases are 1.1 and 2.3 to 17.1 and 35.2 respectively.

2.5.3 Adaptive heatmap

The articles discussed above regarding UDP and DARK discuss some issues related to the heatmaps produced by top-down methods which are mainly caused by the encoding and decoding which is essential for all top-down methods. Bottom-up methods do not face such issues as they do not crop and resize the image. Instead, they keep the original image or only resize it. Nevertheless, bottom-up methods have their own issue. Luo et al. (2021) look at the heatmap regression for bottom-up methods and the conjoined inequality. In both top-down and bottom-up methods, the ground-truth heatmap is created by putting

a 2D Gaussian kernel on the keypoint for each keypoint. This allows for computing the loss of a model using L2, or the sum of the squared differences and implementing and computing the loss in this manner is effortless.

As top-down methods crop out a person based on its bounding box and then resize this cropped image they make sure that each person has roughly the same scale, but this is not the case for bottom-up methods where no cropping is used. Therefore, as the authors note, this causes some issues as the standard deviation for the Gaussian kernel is the same for each keypoint. When the model has to predict the keypoints for people in an image at different sizes (people stand at different distances from the camera), the standard deviation is still the same which means that having a slight offset in regards to the ground-truth keypoint is penalised harsher for larger/closer people. The authors propose scale-adaptive heatmap regression (SAHR) to adjust the standard deviation for the heatmaps for each keypoint individually to compensate for the different scales. They propose to add a new branch that will predict a scale map (with the same size as heatmaps). This scale map indicates the estimate for the standard deviation for the Gaussian kernel that should be used with the ground-truth heatmaps. If a person appears larger and thus has a larger scale, the corresponding Gaussian kernels should have larger standard deviations, as well as smaller standard deviations for smaller people.

The second issue is the inequality regarding people in the foreground versus in the background. They state that bottom-up models perform better on smaller people, in the background, and due to SAHR this difference increased. Therefore, the authors propose weight-adaptive heatmap regression (WAHR) which ensures that the model focuses more on harder (larger) people in the image by adjusting the training loss weights.

Using the HigherHRNet-W48 (640×640) as their backbone in combination with SWAHR (SAHR and WAHR combined), they achieve an AP score of 70.2 on the COCO test-dev set. This score improves the HigherHRNet-W48 baseline by 1.8 percentage points. For the HigherHRNet-W32 (512×512) the performance is improved from 66.4 to 67.9. When they used multi-scale testing they found that HigherHRNet-W48 went from 70.5 to 72.0. Moreover, the performance comes rather close to that of top-down methods such as HRNet-W48 and Simple Baseline which achieve 75.5 and 73.7 respectively. This is noteworthy as, in general, top-down methods achieve higher scores partially due to the advantage of having single-sized input images based on the bounding boxes of people present in the complete image. They also looked at the performance on the COCO val2017 set, using single-scale testing, to find out the additional performance for SAHR and WAHR individually using the HigherHRNet-W32 (512×512) model. They found that using either improves the baseline with 1.3 AP points for WAHR and 0.7 points for SAHR and a combined improvement of 1.8. Just as with UDP, the increase in parameters, ∼100,000, and the increase in GFLOPS, 0.1 and 0.3 for the W32 and W48 backbones, is negligible.

On the CrowdPose dataset (Li et al., 2019), HigherHRNet-W48 with SWAHR achieves

an AP of 71.6 with single-scale testing and an AP of 73.8 with multi-scale testing. These are improvements of 5.7 and 6.2 percentage points over the standard HigherHRNet-W48 model (see Section 2.4.3).

2.6 Performance metrics

When the models are created they need to be tested and in order to compare them with other models, there needs to be a clearly defined metric that can be used to measure the performance of the models. While with classification tasks it would be easy to determine whether a model's prediction is correct (the model either predicted the right class or not), with human pose estimation this is less straightforward. We could consider a prediction correct only if it has the exact same location as the ground-truth, but this would be too strict. As discussed in Section 2.5, some errors can occur due to inaccurate encoding and decoding and for some images, a model might not be able to predict the exact same location as the ground-truth due to decoding inaccuracy. Moreover, a keypoint estimation that is two pixels removed from the ground-truth location is 'more correct' than if it were twenty pixels removed.

If we want to be able to express the model's performance in terms of accuracy we will need a binary label for whether the prediction is correct. Thus, we cannot simply use the distance to the ground-truth, even if we take the person's scale and size into consideration. Therefore, the metrics most commonly used for determining the correctness of a keypoint estimation use some distance threshold; if the distance between the prediction and ground-truth is smaller than the threshold it is considered correct, else it is incorrect. This threshold is often defined based on the image or person in the image.

2.6.1 Percentage of Correctly estimated body Parts (PCP)

The Percentage of Correctly estimated body Parts (PCP) is used for the FLIC dataset (see 2.7.2) and was used for multiple older human pose estimation datasets (Eichner et al., 2012) (Sapp and Taskar, 2013). The PCP metric was originally used for the Buffy dataset and considers a part correct if its segment endpoints lie within 50% of the ground-truth segment length from their annotated length (Ferrari et al., 2008).

Although the PCP was important for the first human pose estimation models and datasets, it also has some disadvantages. Yang and Ramanan (2013) mention a few problems with the PCP metric in its original form. The first is mainly related to the Buffy dataset, where the metric described in the paper is different from the one used in the toolkit; the PCP in the toolkit uses a relaxed version. Secondly, PCP is very sensitive to foreshortening, a limb appearing shorter due to its angle (towards or point away from the camera), of limbs and thus is not equally strict in all images. Lastly, false positives are not penalised by the Buffy toolkit. A false positive occurs when a model predicts a pose (combination of keypoints)

somewhere when it is not actually there, which means that predicting a large number of poses is advantageous.

For the FLIC dataset, there was a redefinition of the PCP metric, ridding PCP of the ambiguity of the original definition. The definition for PCP for a certain radius r(pixels) and image t is defined by Sapp and Taskar (2013) as:

$$PCP(r,t) = \begin{cases} 1, & \textbf{if } \frac{\theta \cdot ||y_i^{t*} - y_i^t||_2}{||y_{l\_hip}^t - y_{r\_sho}^t||_2} \leq r \\ 0, & \textbf{otherwise} \end{cases} \tag{2.6}$$

where θ is the torso length, y_i^{t*} is the predicted joint location for the i^{th} joint and y_i^t is the ground-truth location. $||\cdot||_2$ is the L2 norm of the vector or in other words, the square root of the sum of the squared vector values, or the Euclidean distance between the two vectors (locations). $||y_{l\_hip}^t - y_{r\_sho}^t||_2$ is thus the Euclidean distance between $y_{l\_hip}^t$ (left hip) and $y_{r\_sho}^t$ (right shoulder). This equation determines whether the predicted joint location is within a Euclidean distance of r pixels, where the torso size is defined as the Euclidean distance between the left hip and the right shoulder. To get the accuracy for a whole set of images, the authors simply use the following equation:

$$Accuracy(r) = \frac{100}{N} \sum_{t=1}^{N} 1(PCP(r,t)) \tag{2.7}$$

where r is the maximum radius in pixels within which an estimation is considered correct and N is the number of images in the set. $PCP(r,t)$ is defined by equation 2.6. In addition to FLIC, the LSP dataset also uses PCP as its metric.

2.6.2 Percentage of Correctly predicted Keypoints (PCK)

As mentioned in Section 2.6.1, Yang and Ramanan (2013) criticised the original PCP metric and they proposed a new metric called Percentage of Correctly predicted Keypoints (PCK). PCK rids itself of the tight annotated bounding box requirement, and not only evaluate a subset of verified bounding boxes found by a detector because this tends to favour rigid poses. The authors note that given a bounding box, the estimator must return the keypoint locations and is defined as:

$$PCK = \begin{cases} 1, & \textbf{if } ||y_i^* - y_i||_2 \leq \alpha \cdot max(h,w) \\ 0, & \textbf{otherwise} \end{cases} \tag{2.8}$$

where h and w are the height and width of the bounding box respectively. The α variable can be adjusted based on the dataset or to find out the performance for different levels of strictness. The fact that the maximum distance between the ground-truth and prediction is based on the bounding box means that foreshortening is no longer a problem and

thus keypoints are penalised more equally. However, PCK does require the ground-truth bounding box in order to be used but it is not always available, especially with novel data.

2.6.3 Percentage of Correctly predicted Keypoints head (PCKh)

When developing the MPII dataset (Andriluka et al., 2014), the authors discuss both the PCP and PCK metrics. They preferred the PCK but also proposed a slight alteration to its definition. They call this new metric PCKh, where the 'h' stands for head, and define it as:

$$PCKh = \begin{cases} 1, & \textbf{if } ||y_i^* - y_i||_2 \leq 0.5 \cdot H \\ 0, & \textbf{otherwise} \end{cases} \tag{2.9}$$

where H is the head length. The authors chose to use the head length to ensure that the metrics are articulation-independent. With PCK the threshold is not entirely pose-invariant. If, e.g., a person is 180cm tall and a picture of him standing up is taken (the body is positioned perpendicular to the camera and his arms are positioned along his body), then the threshold is dependent on his height, in pixels, in the picture. This is because the height in the photo will be larger than the width. If we now take another picture of him but this time he points his arms straight up, then the height is still larger than the width, as his width in the image has not changed. But, in the new picture the bounding box has become larger as the highest point is no longer the top of his head, but the end of his arms. This means that a model has to be less accurate when estimating the location of the joints. By using the head length this problem is solved as the size of someone's head cannot be altered while the bounding box can.

2.6.4 Object Keypoint Similarity (OKS), Average Precision (AP) and Average Recall (AR)

The Object Keypoint Similarity (OKS) metric is used for the MS COCO dataset (see Section 2.7.4) and is defined as:

$$OKS = \frac{\sum_i exp(-d_i^2/2s^2k_i^2)\delta(v_i > 0)}{\sum_i \delta(v_i > 0)} \tag{2.10}$$

where d_i is the Euclidean distance between the ground-truth and predicted location for a keypoint i, s is the scale of the person, k_i is a per-keypoint constant that controls falloff and v_i is the visibility flag (Lin et al., 2014) (Cheng et al., 2020) (Lin et al., 2022). δ is defined as min(v_i,2) (Golda, 2019). The scale, s, is defined as the square root of the object segment area in pixels (Ruggero Ronchi and Perona, 2017). The per-keypoint constants (see Table 6), k_i, were calculated by looking at the standard deviation of humans' (redundant) annotations on 5000 images so that the OKS is perceptually meaningful and easy to interpret (Golda, 2019) (Ruggero Ronchi and Perona, 2017). These 5000 redundantly annotated images are from the validation set and the k_i is set to $2\sigma_i$, where σ_i

is the standard deviation of the annotations for keypoint i in the 5000 images (Lin et al., 2022). The v_i can either be 0, 1 or 2, where 0 means that the keypoint is not labelled

Joint	Eyes	Nose	Ears	Wrists	Elbows	Shoulders	Knees	Ankles	Hips
k_i	0.025	0.026	0.035	0.062	0.072	0.079	0.087	0.089	0.107

Table 6. The k_i values used for the OKS metric (Lin et al., 2022).

(no ground-truth annotation), 1 means that it is labelled but not visible and 2 means that it is labelled and visible. The $v_i > 0$ part of equation 2.10 ensures that keypoints that are not visible and not labelled in an image do not affect the score of a model (Lin et al., 2022). As the OKS returns a value between 0 and 1, it is not immediately clear where the cutoff would be, what value is considered correct and what value is not. That is where the threshold comes in, when the OKS is larger than the threshold it is considered a correct estimate. The threshold can be any value between 0 and 1 as well, but usually, it is between 0.5 and 0.95. The Euclidean distance between the ground-truth and the predicted location for a keypoint can be defined as:

$$d_i = ||y_i^* - y_i||_2 \tag{2.11}$$

where y_i^* and y_i are the estimate and ground-truth location for keypont i respectively. This definition of OKS can be seen as IoU(intersection over union) for keypoints. Some interesting and important notes on the OKS are: (1) at an OKS of 0.5, humans perform nearly perfect (95%), (2) the median OKS of human annotators is ~ 0.91 and (3) human annotator performance drops rapidly after an OKS of 0.95.
In order to compare models, the authors use the Average Precision(AP, sometimes mAP) and Average Recall(AR) which are based on the OKS. Unless specified otherwise, AP and AR refer to the mean of AP and AR scores at OKS = {0.50, 0.55, ..., 0.90, 0.95} thresholds, AP^x refers to the AP with an OKS threshold of x (idem for AR^x) (Cheng et al., 2020) (Lin et al., 2022). Precision refers to how accurate the estimator's predictions are and how close they are to the ground-truth position. Recall refers to how many of the ground-truth annotated keypoints it managed to detect and predict. So a pose estimator that only predicts keypoints that it is confident about and that are easy, thus the prediction being precise, will have a relatively high AP but low AR. A pose estimator that can detect keypoints well, or just makes a guess when it has very little confidence, will have a high AR and low AP. The AP^{50} and AP^{75} are alternative names for $AP^{0.5}$ and $AP^{0.75}$. AP^M and AP^L refer to the persons with a scale, s, that is medium or large respectively. A person's scale is considered medium if $32 < s < 96$ and large if $s > 96$ while persons with a scale smaller than 32 are not annotated (Lin et al., 2022).

2.7 Datasets

Many different datasets can be used for training and testing a human pose estimation model. The earlier models had fewer images and contained, overall, images that are easier for models to predict the keypoint locations on. As models were able to achieve higher performances on the early dataset, the demand for more challenging ones became more pronounced. These more challenging datasets also tend to have more diversity in their images (not only movie frames or sports images) as well as more images overall. One of the most recent datasets, CrowdPose (see Section 2.7.5), was constructed with 'crowdedness' and occlusion in mind.

2.7.1 LSP

The Leeds Sports Pose (LSP) dataset was created by Johnson and Everingham (2010) in 2010 and then updated a year later (2011). The authors wanted to have a larger and more realistically sized dataset for human pose estimation. The dataset consists of 2,000 images of full-body poses. The authors searched on Flickr for images using the following queries: athletics, badminton, baseball, gymnastics, parkour, soccer, tennis and volleyball. The images were scaled, for testing, such that the labelled person would take up approximately 150 pixels in length. The dataset was split up into a training set and a test set, both containing 1,000 images.

In the second article (Johnson and Everingham, 2011) they extended the dataset as they noticed that there was a bias for upright poses in the original LSP dataset. The authors decided to use Amazon Mechanical Turk, which is an online platform where you can create tasks for other people to complete in exchange for a small monetary reward, as collecting and labelling images can be quite laborious. They queried 10,800 images with the tags "gymnastics", "parkour" and "athletics" as they found that these have the most challenging poses in the original LSP dataset. They manually selected the images, ensuring a wide range of challenging poses. The so-called Turkers had to label 14 joints per image in combination with an indication of the visibility of each joint. After rejecting some of the annotations and images, they had 10,000 labelled images left and improved upon the annotations using an annotation refinement model. The new images, sometimes seen as a distinct dataset called LSP extended, were added to the training set which means that the LSP training set now has 11,000 images and the test set 1,000.

2.7.2 Frames Labelled In Cinema (FLIC)

The Frames Labelled In Cinema (FLIC) dataset was created by Sapp and Taskar (2013) in 2013. The dataset was created because existing datasets such as Buffy and Pascal Stickmen were quite small, only containing hundreds of images, and the authors wanted a larger dataset for training their model. The FLIC dataset contains 5003 images from popular Hollywood movies and was created by running a state-of-the-art person detector

on movies. They used 30 movies and every tenth frame of each movie. Approximately 20,000 candidate images were extracted based on the person detector's confidence. These images were then labelled using Amazon Mechanical Turk ($0.01 per image in this case). The so-called Turkers labelled ten upper-body joints per image, with five Turkers labelling each image. The authors used the median of the results per image to be robust to outlier labels. Then some images were rejected by the authors if the person was occluded or severely non-frontal. Of the 5003 images, 1016 or about 20% were set aside to be part of the test set.

2.7.3 Max Planck Institut Informatik's dataset (MMPII)

Andriluka et al. (2014) created the MPII dataset in 2014 and aimed to include a wide variety of poses. For collecting the images they used 491[3] activities in 21 different categories (e.g., activity "rock climbing" in category "sports"). The images were collected from YouTube using queries that are based on the activity. They collected 3,913 videos spread across 491 activities of which they manually selected frames while making sure there were people present in the frame. They also made sure to not use frames with very low quality, frames where the person appears very small or frames where only a small part of the person is visible. Additionally, they tried to select frames that had different people present or the same person but in a substantially different pose. This resulted in 24,920 frames where the people present in the frames were annotated by the authors and Turkers, but they did ignore the people in large crowds. The joints and joints' visibility were annotated with specific person-centric left/right annotations and a total of 16 joints per person. They also selected the Turkers carefully to ensure annotation quality based on a qualification task. In total there are 40,522 people for whom the keypoints are labelled. They divided the images such that $\sim$%75 (28,821 people) belong to the training set and the rest ($\sim$%25 of images or 11,701 people) to the test set while making sure that images from one video were grouped and placed in either the test set or the training set.

2.7.4 Microsoft Common Objects in Context (COCO)

The Microsoft Common Objects in Context (MS COCO or just COCO) dataset is one of the most important datasets for human pose estimation at the moment. Nearly all modern human pose estimation models are tested on this dataset due to its large sizes and the yearly challenges from 2016 to 2020. The dataset was originally created in 2014 by Lin et al. (2014) and was meant for object detection and object segmentation. In total, there are 200,000 images with 250,000 person instances where each person is labelled with 17 keypoints. The dataset has changed over the years, e.g., in the articles by Sun et al. (2019), Xiao et al. (2018), Huang et al. (2020b) (HRNet, Simple Baseline and UDP), they mention that they used the train2017 dataset for training which consists of 57,000

[3]The article (Andriluka et al., 2014) mentions 491 activities while the website mentions 410, http://human-pose.mpi-inf.mpg.de/

images and 150,000 person instances while He et al. (2017) trained on the trainval35k set. The train2017 is the most commonly used training set and there is not an updated training set on the official website [4] (Lin et al., 2022). The 2017 validation set has 5,000 images and the 2017 test-dev set has 20,000 images (Sun et al., 2019). The test-dev set is not available for download as the challenge related to it is not finished yet. The performance of some models' best-performing variants can be found in Table 7. Here, BU means bottom-up and TD means top-down.

	Stacked Hourglass (2016)	Mask R-CNN (2017)	Open-Pose (2019)	G-RMI (2017)	Ass. Embed.* (2017)	Person-lab* (2018)	Higher HRNet-W48* (2020)	CPN+ (2018)	Simple baseline (2018)	HRNet-W48 (2019)	UDP++ (2020a)
Approach	-	TD	BU	BU	BU	BU	BU	TD	TD	TD	TD
AP	0.46	63.1	64.2	64.9	65.5	68.7	70.5	73.0	73.7	75.5	80.8

Table 7. Highest achieved performance of the models discussed before on the COCO test-dev set using the AP (see Section 2.6.4). TD means top-down and BU means bottom-up. * indicates using multi-scale. + indicates ensembled models.

2.7.5 CrowdPose

The CrowdPose dataset was proposed by Li et al. (2019) in 2019 and consists of images of 'crowds'. The authors coined the crowd index which looks at the number of keypoints belonging to person $\neg a$ in the bounding box of person a. They looked at the MSCOCO, MPII and AI challenger (not discussed) datasets and found that in these datasets, most images have a crowd index that is very small. The authors wanted to create a dataset with this crowd index in mind and strove for equal distribution. They divided the images of these three datasets into groups based on their crowd index ranging from 0.0 to 1.0 with group sizes of 0.05. They uniformly sampled 30,000 images from these 20 groups and used these for their new dataset. First, they had to standardise the annotations as MSCOCO has 17 keypoints, MPII has 16 and AI challenger has 14. They chose to use 14 keypoints per person with full-body bounding boxes, analysed the images and chose 20,000 images based on their quality. Each image was annotated by two persons and they used the average location of the two annotations for each keypoint. If the difference between the two annotations of a keypoint were too different, the image was re-annotated. Of the 20,000 images with 80,000 person instances, 10,000 are part of the training set, 2,000 are part of the validation set and 8,000 are part of the test set. The performance of some models' best performing variant can be found in Table 8. The best performing model, I^2R-Net ((Ding et al., 2022)) has not been discussed.

[4] https://cocodataset.org/#download

	Mask R-CNN (2018)	Simple Baseline (2018)	SPPE (2019)	Higher-HRNet-W48 (2020)	HRNet-W48 (2019)	SWAHR Higher-HRNet-W48 (2021)	HRNet-W48+ (2019)	I^2R-Net (2022)
Approach TD	TD	TD	BU	TD	BU	TD	C	
AP	57.2	60.8	66.0	67.6	71.3	71.6	72.8	77.4

Table 8. Highest achieved performance of the models discussed before, except I^2R-Net, on the crowdpose test set using the AP (see Section 2.6.4). HrNet-W48+ used extra training data. TD means top-down, BU means bottom-up and C means combination of top-down and bottom-up.

2.8 Data augmentation and general model improvements

The creation of a successful human pose estimation model mainly depends on the architecture and the data used for training the model. Abundant high-quality data is paramount for training a pose estimation model which can generalise what it has learned and use it on unseen data. Being able to generalise well is one of the most difficult challenges for computer vision models. When this abundant high-quality data is absent, the generalisation performance will be sub-optimal and this can result in the model overfitting (Shorten and Khoshgoftaar, 2019). It is possible to prevent (most of) the overfitting by looking at the training and validation accuracy or loss. If the validation accuracy decreases or the validation loss increases while the training accuracy goes up or the training loss goes down, then this indicates that the model has become worse at predicting for the validation set. So, when this occurs, the model might be overfitting.

Any computer vision model has to learn to recognise something (e.g. a human pose) regardless of factors such as illumination, viewpoint, occlusion, background and scale (Shorten and Khoshgoftaar, 2019). For human pose estimation, the model also has to deal with high diversity in the appearance of people and possible poses. Although there are quite some human pose datasets available (see Section 2.7), these datasets all come with their own restrictions. Some datasets only have a limited number of keypoints, some only contain images of very specific events or they mostly contain images where (almost) all keypoints are visible and thus do not represent occlusion well. It would be possible to create a new dataset with more balanced data, but obtaining new high-quality pose data requires laborious annotations as all the keypoints have to be annotated separately and precisely. Nevertheless, it is possible to artificially increase the number of datapoints in a dataset by using data augmentation.

Data augmentation comprises a number of techniques that artificially enhance the size of the dataset in order to improve the model's performance as well as make the model more robust (Shorten and Khoshgoftaar, 2019). Data augmentation can improve the model's performance without any adjustments to its architecture and allows the model to generalise better due to the extra modified data (Park et al., 2020) (Perez and Wang, 2017). This is especially helpful for tasks where there is very little data, which is the case with, e.g.,

medical data due to concerns over the patients' privacy (Shorten and Khoshgoftaar, 2019). Although there are also other methods for preventing overfitting and increasing a model's performance and robustness, data augmentation tackles the root of the problem, which is the lack of enough data (Shorten and Khoshgoftaar, 2019). Tackling the root of the problem, combined with the ease of this technique, makes for an excellent method for enhancing human pose estimation models.

The data augmentation methods commonly used in human pose estimation can be divided into two groups, geometric augmentation and occlusion augmentation. Geometric augmentation comprises methods where only the spatial location of the content of the image is changed (Shorten and Khoshgoftaar, 2019). Among geometric augmentation are techniques such as flipping, cropping/scaling, rotating and translating, which can prevent spatial biases such as a person's head always being in the top centre of the image (see Figure 17. With flipping the image is flipped around one of its axes, for human pose

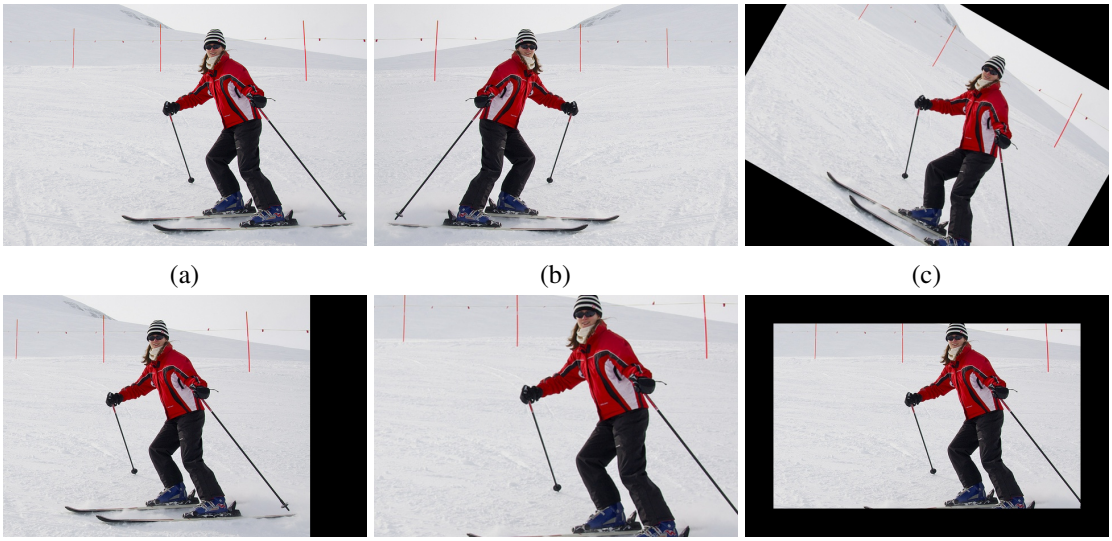


Figure 17. Image 785 from the COCO validation set: (a) Normal. (b) Flipped horizontally. (c) Rotated -30 degrees. (d) Translated -100 pixels. (e) Scaled with a factor of 1.3. (f) Cropped.

estimation, this refers to flipping the horizontal axis. Flipping the vertical axis would result in the image being upside-down, which is not how we tend to look at images and so it is not very useful.

Cropping cuts off parts of one or more sides of the image followed by enlargement but is only useful for top-down methods if the augmentation crops out a part of the bounding box of a person, for example cropping out the legs of the person. This is because everything outside of each person's bounding box is discarded by top-down models anyway.

Rotating is self-explanatory and with translation, the image's spatial dimensions are kept the same (unlike cropping) while the image loses pixels on one side while gaining padding on the opposite side.

The second group of commonly used augmentations, occlusion augmentation, con-

sists of methods for artificially occluding a part of the image and was inspired by the dropout (Shorten and Khoshgoftaar, 2019). For human pose estimation, this usually means occluding one or more keypoints and is often done using a geometric shape, such as a rectangle or circle, which is then placed on top of the image. Random erase and cutout are two examples of occlusion augmentation but essentially achieve the same result, a patch of the original image is coloured with a single colour to occlude what was originally there (Huang et al., 2020a). Sometimes it might even be advantageous to combine multiple augmentation methods and apply them to the images in order to make them more distinct from the original ones (Shorten and Khoshgoftaar, 2019).

In spite of the fact that data augmentation can increase a dataset's size and in turn improve the performance and robustness, one ought to be careful with it so as to not overuse it. At best, applying more data augmentation can result in diminishing returns as no truly new data is added and at worst it results in overfitting on the data. Overfitting due to data augmentation is especially likely for a dataset with a very limited number of datapoints (Shorten and Khoshgoftaar, 2019).

Moreover, applying a lot of augmentations can create some new problems in terms of the size of the dataset. Applying only one augmentation to all the images already doubles the size and all these images have to be processed as well. Data augmentation can strain the available memory if the augmentations are done offline (before the training) or increase the required training time if the augmentations are done online (during the training) (Shorten and Khoshgoftaar, 2019).

Using too many different augmentations methods, with or without combining multiple, inflates the number of images significantly and can be especially problematic by creating biases such as spatial biases (Shorten and Khoshgoftaar, 2019). Geometric augmentation can prevent a spatial bias in bottom-up methods, as they process a whole image at once, but for top-down methods, that only process a cropped version of the original image based on the bounding box, it does not work so well. Additionally, there are also some biases specific to human pose estimation that should be avoided.

As humans can take on unaccountably many different poses, the dataset should preferably show this diversity. However, geometric augmentation will not result in different poses (if a man in the original image is touching his head, he will still be doing that in the augmented images). Occlusion augmentation cannot effectively tackle this problem either as it can only occlude parts of poses, not create new ones. In general, each dataset will always have its own biases and no augmentation method can correct a dataset with a severe diversity deficiency (Shorten and Khoshgoftaar, 2019).

2.8.1 Overview of augmentation methods used

Most of the datasets for human pose estimation are relatively small, especially before COCO (see Section 2.7.4), thus data augmentation is commonly used for training the

	Flipping	Rotating	Scaling	Translating	Cropping	Occlusion augmentation
DeepPose (Toshev and Szegedy, 2014)	✓					
Tompson et al. (Tompson et al., 2014)	✓		✓			
Tompson et al. (Tompson et al., 2015)	✓	✓[±20°]	✓[0.50, 1.50]			
CPM (Wei et al., 2016)	✓	✓[±40°]	✓[0.70, 1.30]			
Stacked Hourglass (Newell et al., 2016)	✓	✓[±30°]	✓[0.75, 1.25]			
Mask R-CNN (He et al., 2017)			✓[0.80, 1.00]			
CPN (Chen et al., 2018)	✓	✓[±45°]	✓[0.70, 1.35]			
Simple Baseline (Xiao et al., 2018)	✓	✓[±40°]	✓[0.70, 1.30]			
HRNet (Sun et al., 2019)	✓	✓[±45°]	✓[0.65, 1.35]		✓	
UDP++ (Huang et al., 2020a)†						✓
OpenPose (Cao et al., 2019)						
PersonLab (Papandreou et al., 2018)	✓		✓[0.50, 1.50]	✓		
HigherHRNet (Cheng et al., 2020)	✓	✓[±30°]	✓[0.75, 1.5]	✓[±40]		
SWAHR (Luo et al., 2021)	✓	✓[±30°]	✓[0.75, 1.25]	✓[±40]		

Table 9. Data augmentation methods used by the models discussed, with the four bottom models being the bottom-up models and the others being top-down models. † The authors did mention that they used other augmentation methods, but did not specify which.

models described before. Table 9 shows the augmentation methods used for a number of models, while for the models or model extensions that are absent in the Table[5] there was no mentioning of data augmentation in the corresponding papers. Additionally, in case there is only a ✓and not a range specified, then the authors did not mention the range in their paper.

From Table 9 it follows that flipping is the most common data augmentation technique and it is also the most straightforward of all as there are no parameters to control. Augmentation methods such as rotating, scaling and translating all have a parameter (degrees to rotate, percentage to scale and number of pixels to translate) that can be controlled and the selected ranges can have an effect on the usefulness of the augmentation. Rotating an image only by a few degrees does not have a huge effect, so the range should be big enough. The same applies to scaling and translating, as scaling or translating the image by a very small margin does not take advantage of the augmentation's potential.

One important observation is that the translating augmentation is only used by bottom-up models because top-down methods use a tightly fitting bounding box around a person and crop the image to this bounding box. The method would only have an effect if it removed pixels from inside a person's bounding box, but then this effect would be the same as cropping as top-down methods are mostly translation-invariant. Bottom-up models, however, are not translation-invariant as they always use the whole image.

[5]The models by Yang and Ramanan (2011, 2013), IEF (Carreira et al., 2016), UDP (Huang et al., 2020b), UDP-Pose (Liu et al., 2021)

Meanwhile, cropping augmentation only has an effect on top-down models if it crops out part of the bounding box. Sun et al. (2019) use half-body augmentation, which is a cropping augmentation proposed by Wang et al. (2018), and either crops the image such that only the upper joints (wrists, elbows, shoulders and head) are visible or only the bottom joints (ankles, knees and hips).
Another cropping augmentation, body cropping augmentation, was proposed by Park et al. (2020) and attempts to find a number of crops that crop out some of the keypoints present in the image. They select five different crops where each has four keypoints remaining visible and make sure that not the same keypoints are hidden in the five crops. When the authors tested it on the COCO 2017 test set, they found a performance improvement for CPN (Chen et al., 2018), Simple Baseline (Xiao et al., 2018) and HRNet (Sun et al., 2019). The performance improvement was more clearly visible when a smaller input of 256×192 was used (+1.4, +0.9 and +0.8 AP respectively) than for the larger 384×288 (+0.9, +0.8 and +0.7 AP respectively). Moreover, they found that body cropping augmentation clearly outperforms half-body augmentation for the three models mentioned (0.4, 0.7 and 0.1 respectively).
Lastly, occlusion augmentation is only mentioned by Huang et al. (2020a) (of all the models in the Table) and seems to be a relatively new augmentation technique in the field of human pose estimation.

2.8.2 Occlusion augmentation

In the last few years, there has been more focus on occlusions in human pose estimation and this is demonstrated by the arrival of the CrowdPose dataset (Li et al., 2019)(see Section 2.7.5) and the usage of occlusion augmentation. In order to train a model to achieve high scores on the CrodwPose dataset, it might be useful to try to increase the model's robustness and use occlusion augmentation to increase its performance on crowded images. Occlusion augmentation cannot only help with increasing a model's performance on crowded images but is also a good method for preventing overfitting (Huang et al., 2020c).
Huang et al. (2020c) propose Augmentation by Information Dropping (AID) as an occlusion augmentation to force models to rely less on visual cues and more on learning constraint cues. The constraint cues regard information about a certain keypoint that can be extracted from an image without the keypoint being visible, but by using the location of other keypoints and the distances between them instead. The goal of information dropping is to better teach the model the important discriminative features in the image which enhances the model's robustness. AID ensures that the model relies more on constraint cues because it blocks the visual cues, which is how the robustness is enhanced. The authors mention multiple different occlusion augmentations such as random erase, cutout, hide-and-seek (HaS) and Grid mask (examples can be found in Figure 18).

(a) (b) (c)

Figure 18. Occlusion augmentation methods on image 785 from the COCO validation set. (a) Cutout/random erase. (b) Hide and seek (Singh et al., 2018). (c) Grid mask (Chen et al., 2020).

Random erasing (2020) is defined by Zhong et al. as an augmentation method that randomly selects a rectangle region in the image and then erases the pixels and replaces them with arbitrary values. Zhong et al. also mention using the mean pixel value of the ImageNet (Krizhevsky et al., 2012) dataset, which turns out to be a light grey colour. Huang et al. (2020c) use cutout from the article by DeVries and Taylor (2017) where it is defined as removing a contiguous section of the input image. The method was inspired by dropout and applies a zero-mask to an arbitrary location, which is portrayed as a grey/black[6] patch in an example image. DeVries and Taylor mention that the size of the cutout is more important to control than the shape is and thus use a rectangle as their basic shape.

With both random erase and cutout a section of the image is occluded by a patch to hide the visual information underneath. The methods are very similar in many regards - e.g., both see the size as a hyper-parameter - but they also differ in a few aspects. Random erase is always a rectangle while cutout can be any shape and random erase's shape has to be completely inside the image boundaries, while for cutout only the centre has to be inside. The colour is different but is not of great importance as any colour hides the visual information underneath. Because they are so similar, Huang et al. (2020c) do not research the difference in effects that they have on a model's performance, instead, they only test the effect of cutout.

The data augmentation method called Hide and Seek is proposed by Singh et al. (2018) with the idea being to hide visual information by randomly putting patches on the image (see Figure 18b). The image is first divided up into a grid of squares where each patch inside the grid has a probability of 0.5 to be hidden. It is important to make sure that the patches that are hidden in one image are different from the hidden patches in the next one as well as different from the hidden patches of the same image the next time (epoch) that the model sees the image. The authors mention that the colour distribution of the images in the training set, after augmentations, should be roughly the same as the colour distribution of the test set. That is why the patches are hidden by setting the pixels inside the square to the mean RGB vector of all the images in the dataset.

[6] In the article by DeVries and Taylor (2017) they show a grey cutout while Huang et al. (2020c) show a black cutout

Grid mask is the most recent of the four occlusion augmentation methods and was proposed by Chen et al. (2020). Grid mask creates a grid and hides some patches in the grid in a consistent manner (see Figure 18c). Because it is based on a grid, the squares that are hidden always have a consistent distance r between them and so no two hidden squares touch. The length of the square's sides, d, does not have to be equal to r, unlike with hide and seek. The authors do not mention how the pixels are hidden; whether they take on an average colour or just grey/black.

As none of the four occlusion augmentation methods was created specifically for human pose estimation there is a small disadvantage to them. Especially for random erase and cutout, the section of the image that is hidden might not contain a ground-truth keypoint (joint). When this is the case the augmentation might be less useful as the model might not have relied on the hidden section for estimating the location even if it were not hidden. Both HaS and grid mask have a higher probability of hiding a keypoint and thus might be able to improve the model's performance and robustness better (Chen et al., 2020). Therefore, it is important to look at the random erase and cutout size to increase the probability of hiding a keypoint.

Huang et al. (2020c) found that for the bottom-up method HigherHRNet (Cheng et al., 2020)(baseline 67.8 AP), HaS is the best augmentation (69.0) followed by grid mask (68.2) and cutout (68.1) on the COCO val set. For top-down method HRNet (Sun et al., 2019)(baseline 77.2 AP), cutout (77.8) slightly outperforms HaS (77.7) and grid mask(77.6). Chen et al. (2020) found in all their studies that grid mask outperforms cutout/random erase and HaS on tasks such as classification and object detection. Huang et al. (2020c) mention they only used a limited hyper-parameter search for the augmentation methods, therefore it is possible that this explains grid mask performing worse than cutout and HaS.

One important aspect of occlusion augmentation that should be taken into consideration is the training schedule of the model. Huang et al. (2020c) state that when occlusion augmentation is applied from the start of the training process, it can even negatively impact the model's performance. The inclusion of AID at the start of the training can hinder early learning and postpone training-loss saturation.

They mention two possible solutions, the first being to double the number of training epochs. The second is to split training into two equal-duration stages where occlusion augmentation is only included in the second stage. They found that both solutions have a similar effect and outperform simply doubling the number of epochs without including AID.

Pytel et al. (2021) conclude that targeted occlusion augmentation methods such as cutout, blurring, multi-keypoint cutout and partmix do not improve a model's performance significantly. A targeted occlusion augmentation is one where the occlusion is not at a random location but on a keypoint. For testing, they used the HRNet (Sun et al., 2019) model and use the same training strategy as Sun et al. (2019) with 210 epochs in total. Flipping,

rotating($\pm 45°$) and scaling[0.65, 1.35] helped improve performance from 65.3 to 73.9 AP on the COCO val set and half-body augmentation improved it further to 74.3. However, occlusion augmentation can, at most, improve the AP by 0.2 (from 74.3 to 74.5 AP) and at worst it can decrease performance by 0.9, depending on the occlusion augmentation method.

Pytel et al. (2021) sketch a bleak image of occlusion augmentation while Huang et al. (2020c) show the opposite. The reason for the disappointing results achieved by Pytel et al. (2021) is likely the training schedule. They adopted the one by Sun et al. (2019) which was, in all likeliness, a good schedule for HRNet without the extra occlusion augmentation. But as Huang et al. (2020c) mention, the occlusion augmentation should not be included from the start and the number of epochs should be increased. Neither of those conditions is met by Pytel et al. (2021) and this will limit the usefulness of occlusion augmentation, as Huang et al. (2020c) mentioned.

Lastly, Pytel et al. (2021) mention that occlusion augmentation does not improve a model's performance on occluded keypoints. However, Huang et al. (2020c) show that occlusion augmentation manages to improve a model's performance on both visible and occluded keypoints in the COCO val set, with the performance improvement on occluded keypoints being larger on average.

Keypoint-centric occlusion augmentation

The occlusion augmentation methods discussed above do help with improving a model's performance on occluded keypoints, but it might be possible to come up with an occlusion augmentation method that is more effective and efficient. One goal of occlusion augmentation is to make models more robust and better at predicting the location of occluded keypoints. However, it is more difficult to predict the location of an occluded keypoint based on visual cues alone. Humans do not solely rely on visual cues when they have to locate an occluded keypoint either (Huang et al., 2020a). Therefore, an occlusion augmentation that focuses on a model learning constraint cues would be useful but also requires more attention than random occlusions (Huang et al., 2020a).

With this goal in mind, Huang et al. (2020a) propose keypoint-aware occlusion augmentation. For this form of occlusion augmentation, an arbitrary keypoint k has to be selected from the set of annotated keypoints in the image. Next, an $r \in [0.1 * w, 0.2 * w]$, where w is the width of the network input (cropped image), and a vector δ, for which the range or value is not defined, are selected. The system creates a circle with a radius of r at the location of k with an offset of δ. This offset is there to ensure that the model does not use the circle to infer that the occluded keypoint is at the centre of the circle, which would be the case if the offset was absent.

During training, they used a 0.5 probability to use k, with δ, as the centre of the circle and a 0.5 probability to use a random location. Similar to AID (Huang et al., 2020c), they

first trained for 200 epochs without the occlusion augmentation and then 200 epochs with the augmentation method to fine-tune the model. This resulted in a 0.6 AP improvement on the COCO test-dev set for both the HRNet-W32 and HRNet-W48 backbones. The improvement was the largest for their HRNet-W48plus backbone with a 0.7 percentage point improvement.

2.8.3 Other performance improving methods

Besides data augmentation, there are some other methods for improving a model's performance without changing the architecture of the model itself. One effective method is to use a pose refinement network such as PoseRefiner (Fieraru et al., 2018) or PoseFix (Moon et al., 2019). The models take the original image and an image with the predicted pose as input, which they adjust and refine to make the predicted pose more accurate (Fieraru et al., 2018)(Moon et al., 2019). The predicted pose that is inputted comes from any bottom-up or top-down pose estimation model (Moon et al., 2019).

PoseFix outperforms PoseRefiner and is able to improve the performance of some pose estimation models by a few percentage points (Moon et al., 2019). The performance improvement is greater for bottom-up methods (between +5.0 and +7.3 AP on COCO test-dev) than for top-down methods (between +2.5 and +5.7 AP)(Moon et al., 2019). One advantage is the simplicity of this approach, it can just be added after any pose estimation model. But because it is a whole new network, it also slows down the processing. In general, a pose refinement network can improve a model's performance more than many single data augmentation methods can, but data augmentation methods are easier to implement and only have an impact on training time, not on the speed of estimating the poses.

One performance improvement method that is commonly used (see Table 10) by state-of-the-art pose estimation models is flipping at test time. When the model has to predict the locations of the keypoints, the image is processed in its original and horizontally flipped form. The heatmaps for those two versions get averaged to give the final predictions and can result in a slight performance improvement (1% in the case of the Stacked Hourglass model) (Newell et al., 2016) (Chen et al., 2018).

Another method called Online Hard Keypoint Mining (OHKM) is less commonly used (see Table 10) but can improve performance (+0.8 AP on the COCO val set for CPN (Chen et al., 2018)). As Chen et al. (2018) describe it, there is an additional loss where only the top 8 keypoint losses are punished to learn what the hard keypoints are and improve the performance on those keypoints. Online Hard Example Mining (OHEM), originally proposed by Shrivastava et al. (2016), is similar but looks at difficult images instead of keypoints (Chen et al., 2018)(He et al., 2017). The idea is that there are mostly easy images and few difficult ones in a dataset and so an algorithm tries to find the difficult ones and train more on these in order to boost the performance (Shrivastava et al., 2016).

Dropout is a regularisation method to prevent overfitting and increase robustness by ran-

	Flipping at test	Dropout	OHEM/OHKM
DeepPose (Toshev and Szegedy, 2014)		✓(0.6, FC)	
Tompson et al. (Tompson et al., 2014)			
Tompson et al. (Tompson et al., 2015)		✓	
CPM (Wei et al., 2016)			
Stacked Hourglass (Newell et al., 2016)	✓		
Mask R-CNN (He et al., 2017)	✓		✓OHEM
CPN (Chen et al., 2018)	✓		✓OHKM
Simple Baseline (Xiao et al., 2018)	✓		
HRNet (Sun et al., 2019)	✓		
UDP (Huang et al., 2020b)	✓		
UDP++ (Huang et al., 2020a)	✓		
OpenPose (Cao et al., 2019)			
PersonLab (Papandreou et al., 2018)			
HigherHRNet (Cheng et al., 2020)	✓		
SWAHR (Luo et al., 2021)	✓		

Table 10. Performance improving methods outside of data augmentation.

domly 'dropping' nodes and is applied to the DeepPose (Toshev and Szegedy, 2014) model's fully connected layers (0.6 probability, see Table 10). In the case of fully connected layers, each node is dropped with a certain probability during training, which means that the connections to the nodes are temporarily removed (Srivastava et al., 2014). This means that the model cannot rely on those nodes anymore and thus has to learn to perform the same task without those nodes. By dropping out different nodes each time an example is passed and trained on, the model builds in a certain level of redundancy and thus becomes more robust.

More recent models do not have any fully connected layers anymore, because, unlike DeepPose (Toshev and Szegedy, 2014), they do not predict keypoints directly, but via heatmaps. This means that the standard dropout for fully connected layers cannot be used. Although it is possible to use a slightly altered version for convolutional layers, the performance improvement and other advantages are less clearly visible from the results, if there are any at all (Toshev and Szegedy, 2014)(Garbin et al., 2020)(Gal and Ghahramani, 2015). Srivastava et al. (2014) show that dropout can help when applied to the first layers, but thanks to batch normalisation it became less important (Ioffe and Szegedy, 2015).

Batch normalisation is a different regularisation method that can reduce the required training time where models with dropout usually increase required training time (Srivastava et al., 2014)(Ioffe and Szegedy, 2015). However, Tompson et al. (2015) introduce the spatial dropout, which does manage to improve the model's performance, especially when the training set is small, such as Flic (see Section 2.7.2). Instead of nodes getting 'dropped', with spatial dropout a whole feature map gets dropped, which is based on the colour channels (RGB) in case of human pose estimation (Tompson et al., 2015). Just as with regular dropout, batch normalisation is a good alternative to prevent overfitting and increase robustness.

One commonly used method for improving the performance of bottom-up methods is to use multi-scale inference or multi-scale test. The idea is to process the input image at different scales and then aggregate the resulting heatmaps to create the final prediction (Luo et al., 2021). When Luo et al. tested the HigherHRNet model with help from SWAHR (see Section 2.5.3) they used scales of 0.5, 1.0 and 1.5 and aggregated the resulting heatmaps. Combining the predictions at different scales at test time can improve performance with a by percentage points, 1.8 in the case of the HigherHRNet model in the article by Luo et al. (2021).

Somewhat similar to the multi-scale testing approach is the Post-data augmentation approach by Toyoda et al. (2019) where they use multi-rotation testing to improve the performance on difficult and extreme poses, e.g., a handstand. This approach does not require any extra training but seems to only be advantageous for rare poses.

2.9 Synthetic data

There are many images that can be used for training and testing human pose estimation models and data augmentation allows us to increase the number in case the available training data is not sufficient. However, when looking at pose estimation with rare poses or of children there tends to be less data available (Ludl et al., 2018). Synthesising data can also be useful when the application requires very specific data to be trained on (Covre et al., 2019). Moreover, synthesising data allows for the collection of diverse data and allows for more control on secondary factors such as people's appearances and environmental factors (such as illumination) (Wang et al., 2019).

2.9.1 Synthetic head pose

Wang et al. (2019) propose a coarse-to-fine model to estimate head poses. The authors mention two reasons for choosing to synthesise head pose data. The first reason is that synthesising the data allows for the creation of a diverse dataset. Using their proposed pipeline they can achieve diversity in poses as well as people's appearances as they can control the race, gender and age of the synthetic heads. The second reason is to improve

the performance of the model on more difficult cases such as occlusion and heterogeneous illumination of the faces.
The available datasets were rather limited, especially in terms of more difficult poses. They created a dataset containing 310,000 head poses which were generated from 300 3D head models. The images of one head in the dataset can differ in the pitch, yaw and roll of the head, as well as the lighting of the face and blurring.
A difference between the head pose dataset and real heads can be found in the distribution of degrees of rotation. Where datasets with real heads tend to have a distribution of degrees of rotation that more or less resemble a normal distribution (with no rotation as the mean), in the synthetic dataset the probabilities for degrees of rotation are more equal.
They used a variety of illuminations, which includes coloured illumination as well, in order to resemble real data more closely. A Gaussian blur was added to some of the heads to better resemble images in datasets of real people, as these images can be slightly blurred due to either the camera or the head moving. In order to achieve more diversity in their synthesised dataset, the heads differ in terms of skin tone as well as facial features. Unlike real datasets, the synthetic heads lack hair and different facial expressions.

2.9.2 Synthetic data for corner cases

The article by Ludl et al. (2018) discusses their research into using synthetic data to improve the performance of human pose estimation models on difficult corner cases. They created two datasets, one with synthetic data that acts as a training set, called SIM, and one with images of real humans that act as the test set, called RARESIM. Both datasets are based on corner cases, specifically, people performing handstands, flips (somersaults) or kicks. Normally these difficult poses would occur (mostly) inside a gymnastics hall, but the authors wanted to generate data where these poses occur outside. Their synthetic dataset, SIM, contains 36,225 images and the test set, RARESIM, contains 44 images. The images in the RARESIM dataset were collected and annotated by hand. The SIM dataset is based on 35 different camera positions (in total, 1,035 images per camera) and each image contains one 3D model. There is also an extended version of SIM called BIG-SIM, which has the same poses but uses 48 camera positions. The 6880 images per camera result in 453,998 images in total.
The authors used two different 3D models that appear in the different scenes. In order to create the scenes, they imported a 3D model into Unity as well as a background image. After which they use motion capture data of humans performing the actions to create the poses of the 3D model. For the background they collected images from the internet as well as images from the COCO dataset that do not contain people, to prevent the 3D model from being in front of a human in the synthetic image. For the SIM data, the background remained the same for a whole motion-capture action while for the BIG-SIM the background changed each frame.

In order to test how the synthetic data affects the performance of a human pose estimation model, they chose to use their own implementation of OpenPose (Cao et al., 2019) using pre-trained weights from the original OpenPose model. They combined the synthetic data with images from the COCO (Lin et al., 2014) training set for training the model. The model trained on the SIM dataset has a synthetic to real image ratio of approximately 0.29 to 1. The model trained on the BIG-SIM dataset used many more synthetic images and thus resulting in a ratio of approximately 3.6 to 1.
Their baseline achieved a PCK score of 42.0 on the RARESIM test set, while the model trained on SIM achieved a score of 55.6. This is a significant improvement over the baseline and shows how effective synthetic data can be. Their model trained on the BIG-SIM training set improved the performance slightly to 55.7. When they tested the trained models on the COCO validation set they found that the baseline had the highest performance with an AP of 56.8, while the models trained on SIM and BIG-SIM achieved an AP of 56.6 and 55.3 respectively.
The performance improvement obtained by Ludl et al. their model is significant and noteworthy, but the context should be taken into account. They trained for a few very specific and rare poses (handstands, flips and kicks) and the baseline performance, 42.0, has a lot of room for improvement. The difficulty in these poses for the estimators is just that, the poses. Meanwhile, with adult-child interaction, there are also some rare poses but it requires the model to improve on more than just the poses. The model needs to become better at predicting occluded keypoints and handling impartially visible people due to keypoints being outside the image bounds. Training a model to be better at occluded keypoints is fairly difficult and keypoints being outside the image bounds do not help either.

2.9.3 Synthesising using 3D poses

In addition to synthesising pose in the form of 2D RGB data for 2D human pose estimation, synthesising 2D images has also been used for 3D human pose estimation. While annotating pose data for 2D pose estimation is rather laborious, for 3D pose estimation, it is also significantly more complicated (Chen et al., 2016). While datasets such as COCO (Lin et al., 2014) and MPII (Andriluka et al., 2014) could be annotated via crowdsourcing, e.g., on Amazon mechanical Turk, this is not viable for 3D pose estimation. As the name suggests, the annotations need to be 3D locations of joints which is not as straightforward as annotating 2D joint locations as there one can just specify a location by clicking on the pixel where the joint is located. With 3D annotations, the annotator has to go from an image (two dimensions) to a three-dimensional representation. To achieve this the annotator needs to be able to accurately predict depth based on a single 2D image and this is inherently difficult for humans (Chen et al., 2016). This is also the reason why there is a lack of (suitable) data and thus we have to look elsewhere.

There are Motion Capture (MoCap) datasets that contain a large set of 3D joint locations and these could be used for pose estimation if it were not for their massive lack of diversity. Datasets such as Human3.6M (Ionescu et al., 2013)(Catalin Ionescu, 2011) or CMU MoCap[7], but the RGB images in these datasets were not meant to be used for pose estimation as they lack aesthetic diversity. In Human3.6M there are eleven actors, six males and five females, and only five of them appear in the training data (Ionescu et al., 2013)(Catalin Ionescu, 2011). The actors wear the same clothes in all their videos and additionally, they have markers on their clothing which would generally not be the case in real-world data.

Nevertheless, the Human3.6M and CMU MoCap data can be quite useful for synthesising 3D human pose estimation data by using the pose data to position 3D models with more diversity. Sarafianos et al. (2016) used eight 3D human models, four women and four men, to perform one of three actions. They made use of MakeHuman to generate the models and then created videos of the models in Blender. They wanted to focus on four key aspects, gender and body shape, the viewpoint's position and rotation, the actions performed and clothing. They used three camera rotations and two distances as well as two clothing options (Sarafianos et al., 2016). Although this approach increases the aesthetic diversity, it is still not close enough to real-world data.

Synthesising data for 3D human pose estimation or action recognition has some difficulties which are not always addressed. This can result in less realistic data and potentially reduce the usefulness of the synthesised data. One such difficulty is the environment in which the synthetic human model is placed. In some cases, the background is ignored and the synthesised images do not have a real background, just a solid colour (Sarafianos et al., 2016) or the background does not fit the pose which can result in the model appearing to float in the air (Matthews et al., 2020)(Varol et al., 2017). Hassan et al. (2019) note that because the pose is influenced by the world, for example, the height and shape of a chair, the environment should be considered more when synthesising data and performing 3D pose estimation. When the environment is disregarded issues with the background can occur while the environment can actually be used to improve synthetic data and in turn pose estimators.

Another difficulty is related to the low diversity in poses and actions. As mentioned above, the Human3.6M dataset has 17 actions (Ionescu et al., 2013)(Catalin Ionescu, 2011), Sarafianos et al. (2016) used only three actions and others face a similar shortage (Hassan et al., 2019). Chen et al. (2016) see the low diversity in poses as one of Human3.6M's major flaws and use the CMU MoCap[8] dataset instead. As a solution to the low diversity, they suggest a solution would be to add small variations to the pose data from a MoCap dataset such as CMU MoCap or Human3.6M to still keep the resulting poses realistic

[7] http://mocap.cs.cmu.edu/

[8] http://mocap.cs.cmu.edu/

while also increasing the variety (Chen et al., 2016).
One last challenge is related to the aesthetics of the models used to synthesise data. It is often difficult to generate enough variation in body size, clothing and materials. Matthews et al. (2020) opted to not apply any colour or material to their models and Sarafianos et al. (2016) used the same clothing materials for all their models while the clothing is limited as well. Chen et al. (2016) have more diversity in the materials they used, 1,000 images for the upper body clothing and 1,000 for the lower body clothing. These images that were found online were then transferred onto the models. The downside of their approach is that there is no actual clothing on the models, the clothing materials are just pasted on a 'naked' human model. It is not clear whether this impacts the performance of a pose estimator trained on their data, but it does show the difficulty of creating enough variation. Varol et al. (2017) used a similar approach to rendering the materials onto the models and use 938 different textures. They mention that their human model bodies are varied and realistic by combining the CAESAR dataset with the SMPL (Skinned Multi-Person Linear) body model (Loper et al., 2015). Moreover, just as Chen et al. (2016), they used the CMU MoCap dataset for the poses which allows them to generate a very large dataset.

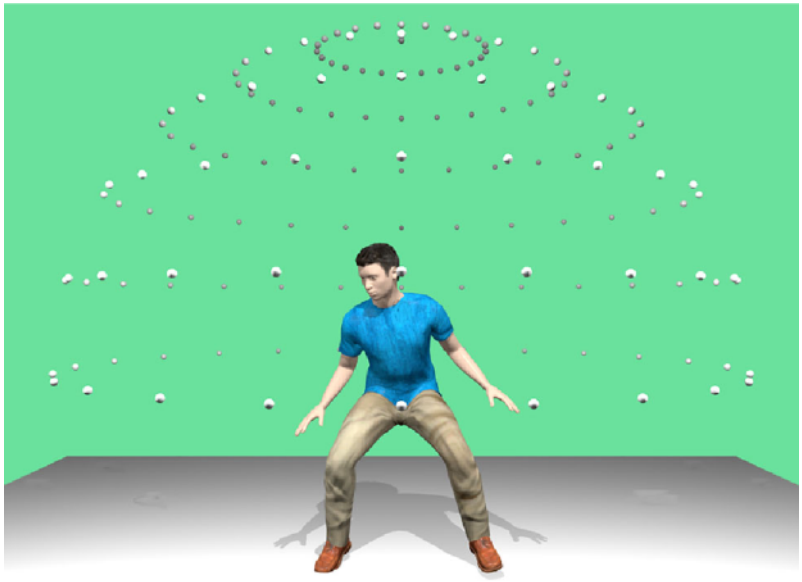


Figure 19. The 180 different camera viewpoints used by Liu et al. (2019).

Liu et al. (Liu et al., 2019) use the CMU MoCap data too but have a different approach to increase the material diversity as well as add variation to the poses of the MoCap data. From the CMU MoCap dataset, they randomly selected 50,000 frames as pose candidates. The candidates were clustered into 339 clusters and one pose from each cluster was selected. They used these 339 poses to position and rotate a 3D human model. The models they used were created in MakeHuman as it allows you to equip a skeleton to the model which makes transferring the pose easier. Additionally, MakeHuman allows you to vary the body shape and proportions and choose the clothing option. The authors created

four models and used Blender to adjust their poses. In Blender, they also randomised the clothing materials (262 upper body and 183 lower body clothing materials downloaded from Google). Moreover, they randomised the lighting, camera position and rotation, background image and model shape. 180 camera viewpoints were used with the viewpoints spread evenly on half of a sphere around and above the model. The cameras were 12 degrees apart in the horizontal and vertical direction, as can be seen in Figure 19.

2.10 Human child pose estimation

Datasets used for training human pose estimation models mostly contain images of adults and there are no publicly available datasets specifically for child pose estimation (Sciortino et al., 2017). However, accurately predicting the poses of children will be necessary when models are used for activity recognition, human-computer interaction and other applications (Sciortino et al., 2017).

There are also some general healthcare-related applications that could benefit from human pose estimators such as quantifying the progression of multiple sclerosis or assessing Alzheimer's (Hesse et al., 2018a). More specifically, there are healthcare applications for children too, one such application is related to communication. While human communication is done verbally as well as using non-verbal communication, such as body-language, babies do not rely on verbal communication as it is not an option. Instead, babies have their own communication methods. It is important for the babies' well-being that they can communicate their needs to their parents or caretakers. Babies use body-language to achieve this and this type of communication can be understood by a human pose estimator (Yurtsever and Eken, 2022). This means that the utilisation of pose estimators could improve the babies' well-being as well as reduce the workload on the caretakers.

There are certain applications where the pose estimation of children is even more useful. An example of such an application is the detection of certain medical conditions in an early stage (Sciortino et al., 2017)(Groos et al., 2021). Prechtl (1990) found that the quality of spontaneous movements can be a good marker for diagnosing developmental impairments. The discovery by Prechtl (1990) led to General Movement Assessment (GMA), which allows us to detect of neurodevelopmental disorders at a very young age (Hesse et al., 2018a)(Groos et al., 2021). With assessment of spontaneous movements and other markers, observers can identify infants with a high risk of disorders such as cerebral palsy, autism spectrum disorder and the Rett syndrome (Groos et al., 2021)(Hesse et al., 2018a). Currently the risk assessment is performed by highly trained specialists, if it is even performed at all (Groos et al., 2021)(Hashemi et al., 2012). Due to the demand for high-skilled observers, it cannot be implemented at large scale (Groos et al., 2021). GMA has some more disadvantages, it is still very time-consuming and the outcome is based

on a subjective evaluation instead of an objective one (Hesse et al., 2017). As the early detection and intervention can significantly improve the child's outcome, making this more accessible is a worthwhile goal (Hashemi et al., 2012). An automated system could allow widespread assessment, early interventions (Hesse et al., 2018a) and reduce the cost of this GMA as well, while the reduced cost would also make it accessible to more people (Groos et al., 2021).

It is already technically possible to use a sensor-based approach to collect motion capture data of an infant which can be processed by a computer model. But this approach has some major disadvantages, and might not work in practice. Generally speaking, babies do not like to have the markers attached to them and can easily start crying. In one experiment, the motion capture recordings had to be stopped due to two-thirds of the babies starting to cry as well as some technical difficulties. Moreover, using this type of approach is rather laborious and it may even affect the movements of the baby (Hesse et al., 2018a).

Using a video-based approach could resolve this and even allow parents to collect data on their infant at home (Groos et al., 2021). An AI model could then process the footage and use human pose estimation to make an assessment of the child's risk of developmental disorders. This would also reduce the cost of the assessment and make it quicker to perform. The approach of Groos et al. (2021) only requires a video of an infant where the camera is positioned above the infant that is lying down. This allows parents to film their infant at home instead of requiring the infant to be observed in a hospital. When they tested their trained models on a test of 4024 images, their best model achieved a PCKh@0.1 of 81.11. The performance of the model comes very close to that of human annotators.

As mentioned before, autism spectrum disorder can be detected in an early stage by assessing a baby's movement. Babies with cerebral palsy and a high risk for developing autism have a higher probability than others to have delays in motor skills and presenting atypical movements (Zhou et al., 2021)(Huang et al., 2021). Hashemi et al. (2012) note that in several studies it has been shown that behaviours indicating autism spectrum disorder were present in home videos of children who were later diagnosed with this disorder.

There are several behavioural indicators exhibited by children with autism, mostly in activities involving visual attention. The authors mention four types of behaviour from the Autism Observation Scale of Infants (AOSI). The first is the shared interest in an event or object, where it is expected that the infant will look at the other person with whom the infant is sharing the interest. Limited looking at the other person is an early risk sign of autism spectrum disorder. The second type is visual tracking, where the infant is expected to track an interesting object as it is moved horizontally across the infant's field of view. Infants with autism spectrum disorder tend to show delayed or discontinuous tracking. The third indicator is related to disengagement of attention, where an infant is expected to be able to move its attention from one visual stimulus to another. The last indicator is regarding atypical motor behaviour. Here there is not one specific motion for assessing

this, instead, the observer has to evaluate the motor behaviour based on a longer period. Especially this fourth indicator could be taken over by an AI to reduce the cost of detecting autism spectrum disorder in an early stage as assessing this indicator is the most tedious.
Another healthcare-related application of human pose estimation is analysing the behaviour of children who possibly have some emotion regulation problems, e.g., anxiety, aggression and social problems (Salah, 2021). AI models could assist experts by estimating the pose or analysing non-verbal communication such as body-language. This data can then be used by the experts to assess a child similar to how pose information can be used by GMA experts (Salah, 2021)(Hesse et al., 2018a).
Due to the lack of pose data of infants and children, there have been some synthetic datasets created, mostly based on babies. A common approach for this is using the SMIL model (Skinned Multi-Infant Linear model) (Hesse et al., 2018b), which is based on the SMPL model (Loper et al., 2015), to create models of babies which can then be adjusted to generate synthetic data (Zhou et al., 2021)(Hesse et al., 2017). SyRIP (Huang et al., 2021) is a hybrid dataset containing real images found online and a larger part of synthetic data generated using the SMIL model. Models designed to predict the presence of cerebral palsy or autism can benefit from these synthetic datasets, especially because there is so little real data available (Hesse et al., 2017)(Huang et al., 2021).

3. Methodology

In this section, I discuss how I synthesised the data as well as the variables that I used, e.g., how many images to synthesise or what camera angles to use. First I discuss the creation of the human models, what challenges I faced and how I tried to solve those. One such challenge was the creation of new clothing meshes. In general, 3D objects consist of a set of vertices which are combined to create faces. The combination of these vertices and faces make up a mesh, which is how the human models and their clothing are created. In the next sections, a vertex refers to a point in 3D space. Three or four of these vertices combined make up one face, if it has three vertices then it is called a triangle and with four it is called a quad. Most of the meshes that I used are made up of triangles, and some are made up of quads, but there are no meshes made up of triangles and quads.
After discussing the models, I explain how the scenes are created. A scene is a static setting with two models, one child and one adult, and in this scene, the models interact thanks to their poses. One example of a scene would be the adult model holding the child model up in the air. Because the scenes are static, all the synthesised images of one scene show the same models, albeit with different clothes, in the exact same pose.
In the third part of this section, I discuss how the synthesising takes place and what choices I made in regards to randomisation among others. I explained how the annotations are generated as well as the challenges accompanying the methods used.
The fourth section discusses annotating the test set and how I computed the area and the bounding box. Lastly, in the final section, I go into some details about training and testing the pose estimator models with the synthetic dataset and discuss some of the ablation experiments that I performed.

3.1 Human model creation

The first step for synthesising human pose estimation data is to create the human models that will be present in the synthetic data. For this process, I chose to use MakeHuman, which is a free to use and open source program which is able to create 3D human models of both adults and children. Especially the ability to create both adult and child models is paramount for synthesising adult-child interactions. While I have looked into alternative software such as MetaHuman creator (Unreal Engine, 2021), which is able to create the most realistic human models of all the software that I looked at, these alternatives have some major disadvantages.
Most alternatives, such as MetaHuman creator, are not open-source, which makes mass-producing models harder, and most importantly, none of the alternatives allows you

to create models of young children. According to the documentation of MetaHuman creator (Unreal Engine, 2021), there are three different height options, with "short" resulting in a height of 149 and 165 centimetres for the female and male bodies respectively. Simply scaling the models is not an option either, as human adults have different body proportions than children (Burdi et al., 1969). Additionally, MetaHuman creator is limited to early access and only allows for using the model in Unreal engine at the moment of writing.

(a) (b)

Figure 20. (a) Taro, a model from MetaHuman creator. (b) a model from MakeHuman.

As 3D models of young children are of such importance, MakeHuman was the best option for creating the human models and MakeHuman comes with some advantages. One such advantage is the possibility to change each individual material, the clothing and even the skin colour, of the models separately in third-party programs such as Unity. This makes randomising the clothing during the synthesising easier as this means that there is no need for a unique model for each possible combination of the clothing options and skin colour. Moreover, it is possible to create new clothing meshes in Blender which can then be equipped to human models in MakeHuman and the possibilities are thus not limited to the few options provided by MakeHuman.

MakeHuman uses a number of continuous variables to create a model with the most important variables being gender, age, muscle, weight, height, proportions and race. The range of all these values is 0 to 1 with the default value being 0.5. For most variables, it is the case that the aspect that they describe is also affected by other variables. For example, increasing the weight variable makes the model look heavier, but so does decreasing the muscle variable. Both the proportions and gender variables affect the shoulder-hip proportions and age affects all aspects.

The race variable is split up into three sub-variables named African, Asian and Caucasian, where the sum of these three sub-variables always adds up to 1. Changing these race sub-variables has an effect on the height and smaller effects on body parts but the change is most noticeable in the face, with the change in the shape of the nose and eyes being the most visible.

Additionally, there are many more variables that allow you to adjust minor details such as the location of the eyes, ears and nose on the face, the length and shape of arms and legs, as well as the scale and shape of the torso, hips and stomach. These variables are grouped together based on the area that they affect, such as the left/right eye, the nose, the hips etc. It is neither feasible nor useful to discuss all of the variables that can be adjusted, but the majority of these variables affect (a part of) the face. There are eighteen groups with face-specific variables and some of these groups, such as the eyes, have many individual variables within them. Considering this, MakeHuman does allow for an extensive amount of detailing of the human models.

Besides the body shape, there are a few other aspects of a human model that can be changed in MakeHuman. The program has a number of options for eye colour, eyebrows, eyelashes and hair. For these aspects, it is also possible to download additional options inside the community tab. I opted to download a few extra haircuts to increase the variety while trying to maintain realism. A lot of the options available for downloading are either small variations of other haircuts or they tend to look unrealistic, e.g., the haircut has a solid colour, low poly count and little depth.

MakeHuman offers different skins which can be divided into three groups that correspond to the race sub-variables, African, Asian and Caucasian. For each of these races, there are at least three options per gender, one for young people, one for middle-aged adults and one for the elderly.

Besides the model changes, MakeHuman also offers a few skeleton rigs which are necessary in order to adjust the poses in software such as Unity. They are not visible but they do influence the degrees of freedom that the model has and how the "flesh" changes with different poses. These skeleton rigs try to mimic the human skeleton while also reducing the number of bones to make it more manageable.

Some skeleton rigs focus more on mimicking the human bones while others focus more on reducing the number of bones. I decided to use the "Game engine" skeleton rig as this one had a relatively low number of bones (53) while still having some detail such as individual finger bones. Although MakeHuman also offers a few poses and expressions, I found that they both caused issues with the mass production such as either crashing MakeHuman or the model being severely deformed.

3.1.1 Clothing

One of the most important advantages of MakeHuman is that it allows us to equip the models with all kinds of clothes which can help diversify the aesthetics of the human models. The clothes are just meshes rendered on top of the models to resemble real clothes as much as possible. The size and shape of these clothes are not independent, instead, they are based on the model that is equipped with the clothing. This means that the clothing shrinks and grows as the model changes shape via the weight or height and thus ensures that the clothing fits the model well. This is an important feature because without this it would not be viable to arbitrarily equip models with clothing in the mass production phase. MakeHuman is able to achieve this 'universal' fit by basing the location of the clothing's vertices (the vertices that make up the mesh) on the corresponding vertices in the model's mesh.

MakeHuman comes with a few clothing options but these are not ideal for the adult-child interaction scenes as most options consist of formal clothing. However, this is not a problem as it is possible to download more clothing options inside the program or create new clothes via Blender. When selecting clothes there are a number of requirements that should be met in order for me to use them.

The first requirement is that the clothes should be fitting for the adult-child interaction setting, this means that the standard tuxedos, gala dresses and other formal clothing are not an option and neither is the themed clothing options available for download. The reason for this is that these synthetic scenes should generalise well to real adult-child interactions where formal clothing is not often worn.

Secondly, the clothing must fit a model well as a bad fit can result in the skin of the model protruding from the clothing or the clothing being so large that it results in a loss of realism.

Thirdly, the clothes must adjust well with a change in age or race in order to reduce the number of unique clothes and simplify the mass production of the models. As many aspects of the body are influenced by both age and race, a certain shirt or trousers can look fine at first but then look unrealistic when the age or race variables are changed. When the changes are small this is not a problem, but I did discover that larger changes can distort the clothing. For example, changing the age variable from adult to infant or vice versa will usually result in unrealistic clothing with some protrusions in the clothing mesh (see Figure 21).

Lastly, the clothing mesh should have enough detail, where detail here refers to the number of vertices in the mesh. The reason for this requirement is to ensure that the model mesh does not have multiple orders of magnitude more vertices.

I first turned to the clothing available for downloading and tested it against the four requirements mentioned above. Almost all clothing options available fulfil the last requirements with many also meeting the first requirement as there are enough realistic and casual

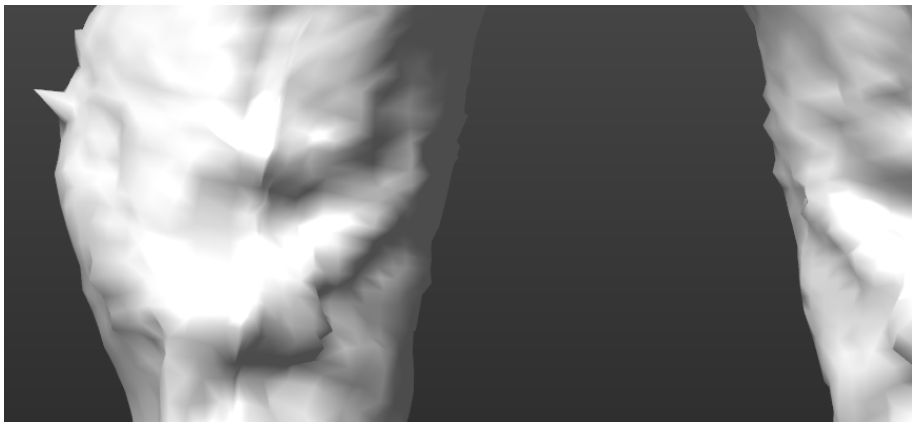


Figure 21. Protrusions when a child model is equipped with trousers for an adult male.

clothes available. I then proceeded to try a number of the remaining options to find out if they meet the second and third requirements as well. Most of the remaining options satisfied the second requirement as they tended to have a good fit for some of the models. Although many clothes did have some skin protruding the clothes, this turned out to be less of a problem as there is an option to remove the faces of the human model underneath the clothing, e.g., the faces of the model's shoulders are removed and thus they cannot protrude the clothing anymore. This does, however, require the clothing to specify a list of vertices that ought to be removed. If the clothing does not specify this list then MakeHuman does not remove any faces below the clothing and thus the protrusions persist.

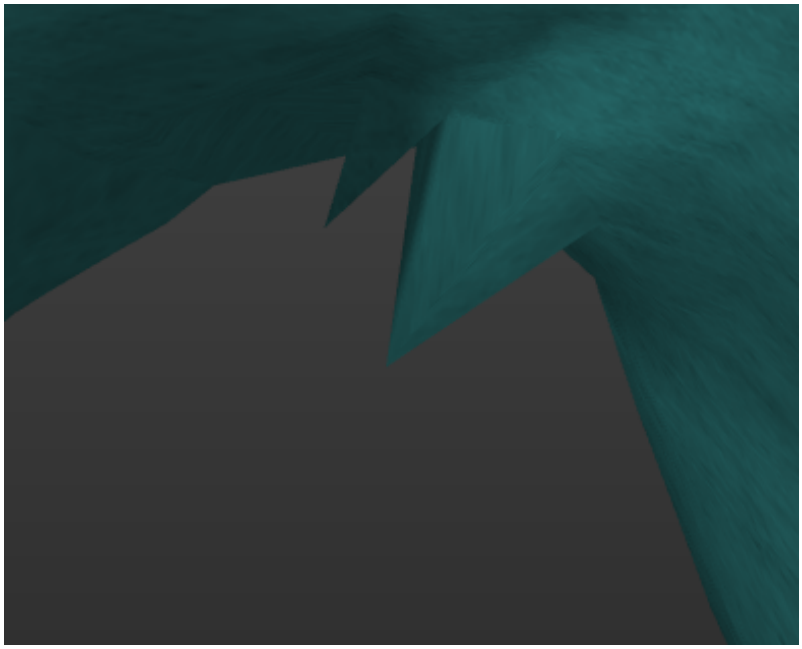


Figure 22. Protrusions in a mesh when a female is equipped with a sweater for males.

Most of these clothing options are either meant for male or female models as the change in the model's body when the gender variable changes are clearly noticeable. This can lead

to problems around the chest, armpits and neck as some protrusions in the clothing appear (see Figure 22). The third requirement, being able to adjust well to a change in age or race, turned out to be difficult to satisfy as often the topologies (the spread of the vertices in the model's mesh) differ between models of different ages and races. Tops, trousers and shoes were not noticeably impacted by a change in race, while a change in age did result in distortion for most clothing options. Meanwhile, some accessories such as glasses were severely distorted by a change in race.

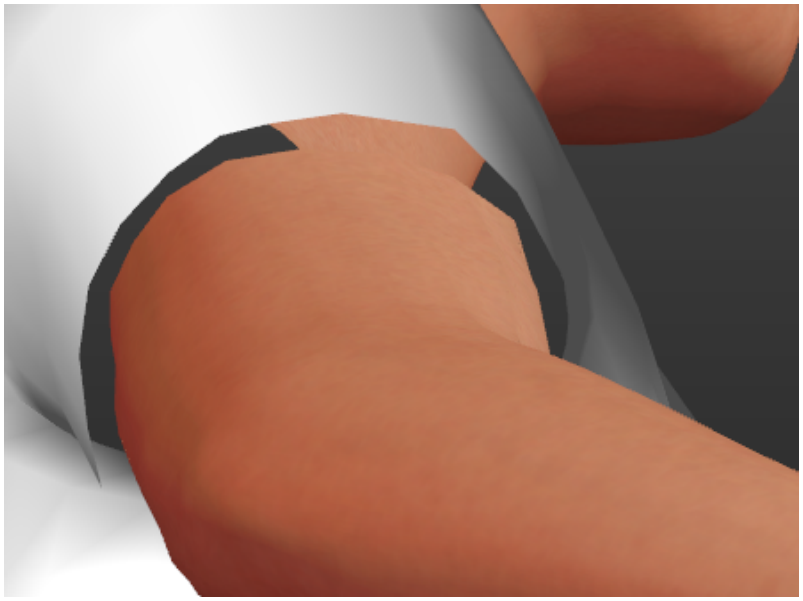


Figure 23. The absence of faces of the skin underneath the clothing. You can see the neck of the model through the sleeve.

Once I had found some clothes that satisfied the four requirements, I exported some models to Unity in order to test the rotation of the limbs. During this process, I stumbled upon another possible issue with the clothing. The clothing that met the four requirements tended to have relatively large holes for the arms, legs and head. Because of these larger holes, there were no protrusions of the skin underneath and thus they met the requirements. Deleting the face right near these holes is uncommon as then, with the right camera angle, it is easy to see that all the skin underneath the clothing has been removed as if there is no real body, just a head with some limbs and clothing (see Figure 23). The problem with these larger holes is that it allows to see more skin underneath the clothing but as the skin farther away is removed, it again looks as if there is no real body.
As it is possible to create and adjust clothes in Blender I tried to shrink these holes but this either resulted in more protrusions or the hole still being large enough to see the lack of skin underneath the clothes. This problem was present with all the clothes that I tried and therefore I opted to resolve this in a different manner. I added a cone shape pointing into the mesh such that it would intersect with the skin in combination with shrinking the hole.

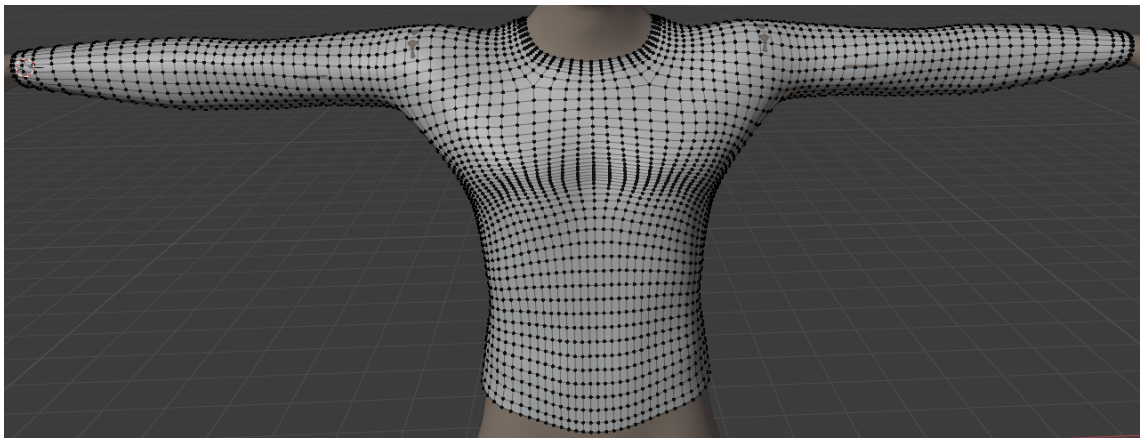


Figure 24. The mesh of a sweater (male) in Blender.

By doing this it is no longer possible to see the lack of skin underneath the clothing mesh. These cones were added to holes at the end of the sleeves, the hole for the neck and the holes at the end of the legs of shorts.
Most of the clothing still had some minor issues which only became clear when the models wearing the clothing took on different non-standard poses, for example holding their arms up in the air. Instead of trying to patch all the less frequent issues that I encountered, I decided to create new clothing for the ones where I had encountered up to that point. The only clothing meshes that I did not create in the end are the turtle neck (adults-only) and the loose trousers, but I did adjust the original clothing meshes to better fit the target group's models and to remove certain protrusions. The reason that the turtle neck is adults only is that a lot of aesthetic issues occur when it is equipped on a child model but nearly none when equipped on a male or female adult model. I chose not to create my own turtle neck for child models as it takes quite some time to create and test a new clothing mesh and I already had some clothing variety for the child models.
In addition to the main clothing, a model can also be equipped with shoes and accessories such as glasses or facial hair. Neither shoes nor accessories are afflicted by the same issues as full, upper and lower body clothing except for one pair of shoes that I tested, called "tennisshoes", which seems to crumble up a bit at the toes when a child model is equipped with them. Fortunately, MakeHuman comes with a few good shoes which do not have the same issue. As for the accessories, these all need to be downloaded via the community tab in MakeHuman and look as intended when equipped on any model that I have tested.

3.1.2 Mass production plugin

Creating a number of scenes requires multiple human models and a mass production method can help with this while maintaining variety in the models. MakeHuman has the ability to randomly create a multitude of models via the "mass produce" tab which comes with a number of settings. This is a good starting point and I have made a modified version

with a few additional options to have more control over the mass-produced models.
The first edit that I made was in regards to the sliders to specify the macro settings such as age, weight and height. These sliders specify the minimum and maximum values for the corresponding variables (see Section 3.1) but these sliders do not show what value the variable takes. I added a numerical value which indicates the value that the corresponding variables can use when the mass production starts.
Next, I added a setting to select mandatory clothing for the models where each clothing option can be made mandatory for either or both genders. I use this to ensure that each model has the correct accessories as well as all the required meshes for localising the keypoints (see Section 3.3.3) and occlusion detection (see Section 3.3.4).

Figure 25. Tennishoes equipped on a child model where the toes look shriveled.

One of the larger changes that I made was to redo the exporting such that the save location can be specified in MakeHuman, the models can get exported to two different file formats and the name of the model is based on the values of the randomised variables. Specifying the save directory inside MakeHuman allows for more flexibility and makes them easier to find. The exporting to two different file formats is necessary because I use the Collada file (".dae") to import the models into Unity but these files cannot be reopened in MakeHuman. Meanwhile, the MakeHumanModel files (".mhm") do allow MakeHuman to load the model and make changes but I was not able to import these files into Unity. I added this exporting option in case I might encounter some issues with the models and I would have to load the models to fix them. I could of course mass produce completely new models but this process takes quite some time. In the end, I used this feature to reload the child models and change their shoes as it turned out that the mesh of the shoes had some issues (see Figure 25).
For both the two file formats, the name is the same, except for the extension, and is based on the model itself. The filename consists of six parts separated by an underscore with the parts describing the gender, age, height in centimetres, race variable, clothing combination and a unique identification number.

Furthermore, I also fixed a small problem present in the original exporting when exporting to the Collada file format. Originally the models are all saved in one folder where selecting the Collada file format results in a Collada file accompanied by a textures folder. This folder is called the same for each mass-produced model, meaning that all the textures for all the models will be placed in this one textures folder. If all the texture components have unique names this would not really cause any issues, but I have encountered one case where two different components had the same name. This meant that the texture of an earlier mass-produced model was overwritten by that of a later one. I solved this by first creating a new folder for each mass-produced model and then this new folder will be the export location, thus ensuring that each textures folder only contains the textures of one model and so no texture is overwritten. Lastly, I added a feature that MakeHuman creates a text file with the values of all the variables used to create the model as well as a list of all the clothing meshes equipped.

The last change that I made is the addition of distribution settings. The original mass production creates a number of models and randomly selects the values for the major variables such as gender and race. I wanted to have more control over this so I added options to have an equal distribution for the gender and race variables, meaning that mass-producing six models results in three males and three females and one of each race. The race can also be pre-determined which ensures that all mass-produced models have a certain race. The last distribution setting is the option to create a model for each clothing combination consisting of one full body clothing or a combination of one upper and one lower body clothes.

3.1.3 Mass-producing models

Once the mass production plugin was finished I was able to mass produce randomised models and test these in Unity. During this testing phase I came across some aesthetic issues with the clothing and shoes, described in Section 3.1.1, as well as discovering the need for invisible clothing meshes which serve the purpose of creating annotations when synthesising data (see Section 3.1.4). During this testing process, I created a new set of human models multiple times and I changed some of the randomisation settings to improve the quality of the results.

The first issue I encountered regards the randomisation of the variables that are part of the torso and hip groups. In these groups, there are some variables that affect the location of the spine and shoulders relative to the hips. For these and the other variables causing issues, the change is most severe with infants. It seems as if the effect of changing these variables is not scaled with the smaller overall body size of infant models. When these torso and hip variables are randomised there tend to be a number of models with irregularities which look like scoliosis or other spinal diseases (see Figure 26). Other exemplar issues are severe widening/narrowing of the chest, unnaturally large dorsi muscles, unnaturally large

Figure 26. Child model with a spine irregularity.

chest muscles, severe widening/narrowing of the hips and unnaturally large/small hips.
The variables in the stomach group, specifically the pregnancy and muscular tone, increase the stomach to an unnatural extent for infant models. Seeing a few "pregnant" infants did occur, although I am not sure whether this was caused by the pregnancy shape or the muscular tone variable.
The last group causing issues, with the pelvis variables, resulted in saggy buttocks or a bump or hole in the genital area. As most variables in these groups could cause some issues and I did not want to spend hours on tuning them to have aesthetically better results, I decided to exclude these groups from randomisation for all models. This means that the values for all these variables are 0.5 for all the models, but it does not mean that all the models have the same stomach/hip/chest sizes. There is also a group of variables called "measure" of which the variables' values are randomised. These variables control things such as the circumference of certain body parts such as the hips as well as relative distances between parts of the torso.
Moreover, the size of the hips, torso, stomach and buttocks is also dependent on the gender and weight variables. One might assume that the clothing meshes hide most of the problems like clothing in the real world can hide certain body irregularities. However, the locations of vertices of the clothing meshes are all dependent on the location of the vertices of the body, which means that the clothing fits the body rather well. The advantage of this is that a clothing mesh can look fitting on a wider range of bodies. The corresponding disadvantage of this is that body irregularities are still visible even if the model is equipped with clothing meshes.

All the other variables that are being randomised, thus excluding the hips, torso, stomach and buttocks groups, have a max deviation of 0.15 for the children and 0.30 for the adults. The reason for this difference in max deviation is to prevent unrealistic models as a deviation has a significantly more noticeable effect on child models. The value ranges are treated as Gaussian distributions with a standard deviation of 0.4 times the max deviation, ergo 0.06 and 0.12 for the child and adult models respectively.
The ranges of the main variables such as age, weight and height can also be specified and have the most impact on the final models. The age variable was most important to control as I needed to create infants and thus set the range of the age variable to [0.0, 0.03] which corresponds to an age range of approximately 1 to 2.3 years. Meanwhile, the range for the adults was [0.5, 0.66] which corresponds to an age range of 25.4 to 46.3 years old. The weight and height ranges were both set to [0.15, 0.85] and the muscle range was left unchanged at [0.3, 0.8]. The values for these variables are uniformly selected from the variable-specific ranges. The sex variable's value was selected either from the range of [0.75, 1.00], which indicates that it is a male, or [0.0, 0.25], which indicates that it is a female. The reason for excluding the (0.25, 0.75) range is to ensure a realistic amount of sexual dimorphism is present in the set of models.

3.1.4 Invisible clothing meshes

Besides the visible clothing meshes with which the models are equipped, there are also some mandatory invisible clothing meshes that help with determining the location of keypoints and detecting occlusions during the synthesising process. These meshes do not remove any faces underneath and are also turned invisible in Unity as there is no need for them to be visible.
The invisible clothing meshes serve either as facial keypoints or as collider meshes for detecting occlusions. The meshes serving as facial keypoints consist of some of the vertices that make up the facial parts that correspond to a keypoint (ears, eyes and nose). By keeping the location of the mesh's vertices the same as the corresponding vertices that are part of the model, the new mesh will always perfectly fit any model. This means that, e.g., a nose mesh is always a perfect copy of the nose of the model that is equipped with this mesh. Therefore, by determining the position of this mesh in the world, we can determine the position of the model's nose. This is of course necessary for creating the ground-truth annotations.
The meshes serving as collider meshes correspond to a singular body part of a model and can help with determining occlusions and self-occlusions. Similar to the meshes serving as facial keypoints, these collider meshes have vertices of which the location corresponds perfectly to their corresponding vertices on the model. I go into further detail about these meshes in Section 3.3.4.

3.2 Creating the interaction scenes

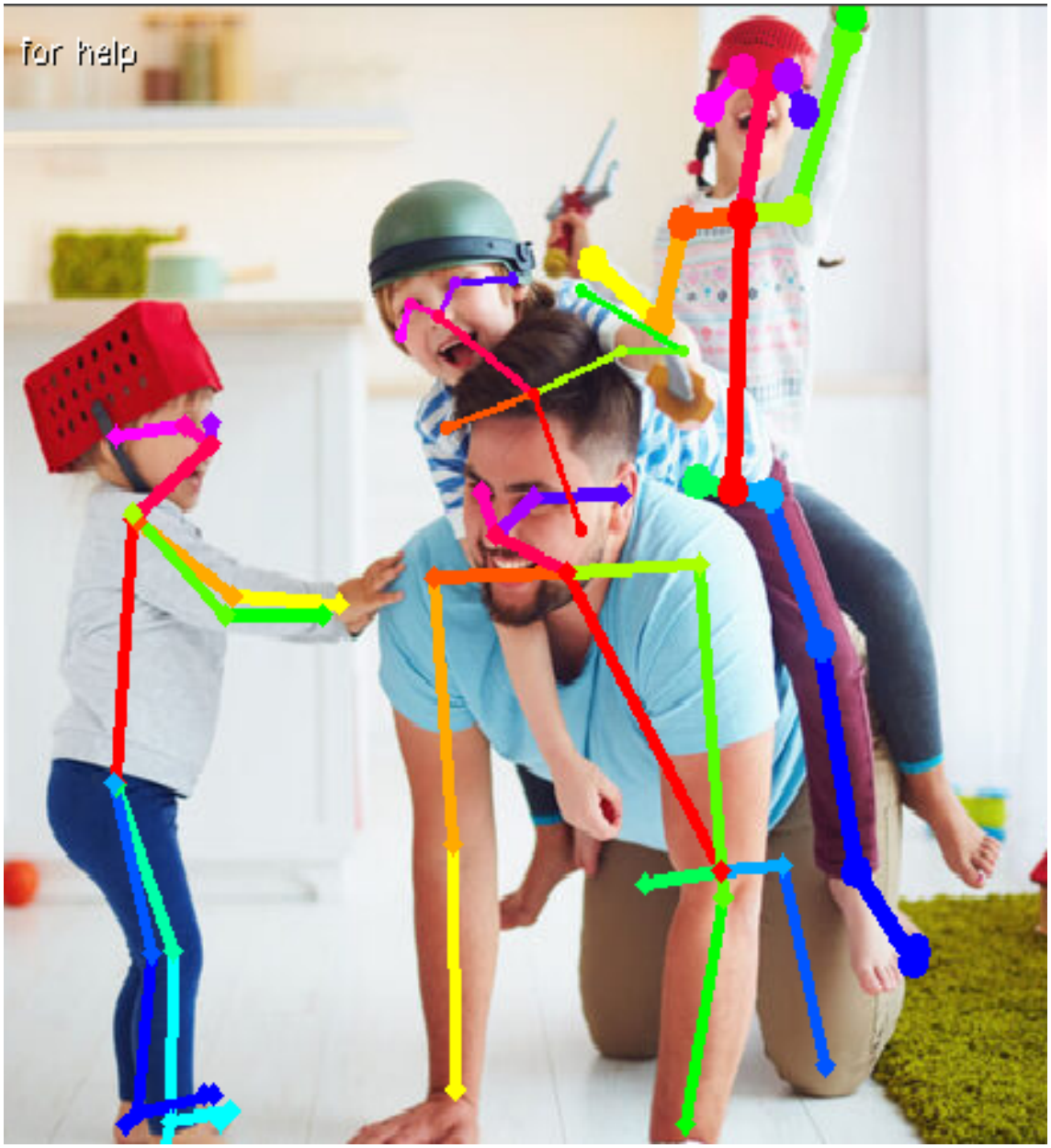


Figure 27. OpenPose prediction example.

Once the mass production of models was finished and all models were imported into Unity I could start creating the interaction scenes. At that time I knew little of what the test data would look like, therefore I first tested a human pose estimator on some images of adult-child interaction that I found online. One example of this can be seen in Figure 27 While I was inspecting the predicted poses I concluded that the pose estimator would benefit most from data where the adult and child are close together with some inter-person occlusion. Therefore, I decided to create scenes where the adult and child are relatively close together and as a result of this there will be inter-person occlusion when the camera

is rotated around the people.
The first step to creating these interaction scenes was finding some images of interaction that can serve as inspiration. I could try to create scenes without looking up real interaction but this would have an increased risk of creating unrealistic scenes. A scene can be seen as a setting with an adult and a child interacting inside a room. The scenes are (mostly) unique in the combination of adult and child models as well as the combination of the models' poses. In Unity, a scene object can be seen as a whole new world or a new level in a video game. An empty scene in Unity, so without the human models, consists of a floor, a camera that can render the scene and a lighting source. The floor is there to position the human models on and to prevent the models from floating in the synthesised images.
The first step to creating a new scene is to copy this standard scene and place two human models, one adult and one child, inside of the scene object. These models were selected semi-randomly, I programmed a method that would return a model from a set of available models. Originally all models were in the set and after picking one model it would be removed from the set and when the set was empty I added all models again. I did this to ensure that each model is present in at least two scenes. In total, I created 40 scenes so most models are present in two scenes and eight models are present in three.
It would also be possible to use just one scene object in Unity and then create all the different scenes in one scene object. The advantage of this is that there is no need to switch between different scene objects when synthesising data. This approach, however, also comes with a disadvantage for creating the interaction scenes. In order to create a scene, the models have to be positioned and their bodyparts have to be positioned and rotated to create the scene. Doing this required a lot of adjustments and scrolling through the scene object's hierarchy of gameobjects (models, floor, camera etc.). If all the scenes are present in one scene object there would be a lot of gameobjects in the hierarchy, even if the models are invisible to the camera, and this makes finding the right one more difficult.
After I had created the first 30 scenes I saw some footage of the Youth dataset. As this dataset could serve as a good test set due to the setting and the interaction between adult and child, I decided to create the last scenes with poses more like the example video.

3.3 Synthesising data

In order to use the scenes efficiently there need to be many images of one scene, but all these images need to be different enough from one another as well. This can be done by rotating and moving the camera around the people in the scene to take a screenshot of the scene from different angles and thus have different images. Increasing the aesthetic diversity of the images can be done by randomising certain aspects of the scenes such as lighting, materials, models and the background. I discuss the randomisation of lighting, clothing and their materials in Section 3.3.2. Randomising the background is quite easy in

Unity but since I was not certain what the test data would look like, I chose to use a bright green background (like a green screen) which could be replaced after the synthesising. In the end, I got permission to use the Youth data as a test set, so the green backgrounds were replaced by images of the empty room used to create the Youth data. As for the floor, I used a blue-grey carpet texture instead of randomising it.
Randomising the models themselves is currently not possible as the close interaction and skin contact should stay the same and this cannot be guaranteed if a model in the scene is replaced by another. At the very least this requires the other model to change its pose to adapt to the body shape and size of the new model. If, e.g., an adult and a child are sitting on the floor next to each other and the adult's hand is on the child's shoulder, changing either the adult or child model could end the skin contact. The child's shoulder might be higher or lower due to the child model being taller or shorter and so the adult's hand would either be floating above the shoulder or would intersect with the child model. On the other hand, the adult might be taller or shorter as well which could give the same problem. Hence, changing models requires the program to know exactly where the body parts should be to keep the skin contact present.
One could argue that the hovering hand would not be an issue, but the hand intersecting with the shoulder, as if there were a hole in the child's body would certainly be problematic and unrealistic. Therefore, I decided not to pursue this possibility any further, although I know that it is possible to achieve this. Inverse kinematics would allow the model's hand to be placed on the shoulder, but to implement this successfully for all the scenes I would have to take the rotation and location of all joints into consideration. Moreover, it would require the program to know when the models make skin contact, which is currently not the case.

3.3.1 Camera randomisation

The most straightforward method of increasing aesthetic diversity in the synthesised data is randomising the location and rotation of the camera. To aid with the randomisation of the camera's location and rotation I implemented a number of camera variables which can be divided into three groups.
The first group consists of the horizontal rotation and the vertical rotation. The horizontal rotation determines the angle of the camera on the y-axis as well as a general location. With each increase in the horizontal rotation, the camera rotates a certain amount of degrees on its y-axis and changes the horizontal position. The position is computed such that the distance between the camera and the scene centre remains the same and the angle between the current position, centre and start position is equal to the Euler angle of the camera's y-axis. The equations to compute the position of the camera are defined in equations 3.1,

3.2 and 3.3, which indicate the x, y and z coordinates for the camera.

$$P_{cam_x} = (D_{base} * \cos(\delta_V)) * \sin(\delta_H) \tag{3.1}$$

$$P_{cam_y} = \max H_{min}, C_y + (D_{base} * \sin(\delta_V)) \tag{3.2}$$

$$P_{cam_z} = (D_{base} * \cos(\delta_V)) * \cos(\delta_H) \tag{3.3}$$

In these equations, the P_{cam} is the position of the camera with the x, y and z subscripts indicating the axis of the position. The δ_V and δ_H are the vertical and horizontal degrees respectively. H_{min} is the minimum height and the D_{base} is the base distance to the camera. These equations ensure that the camera rotates along a circle where the centre of the circle corresponds to the centre of the scene. The vertical rotation works in a similar way as it affects the angle of the camera on the x-axis as well as the position.
The base distance to the camera is based on the scene centre, which is defined as:

$$C = \frac{\sum_{m \in M} \sum_{k \in K_m} P_k}{\sum_{m \in M} \sum_{k \in K_m} 1} \tag{3.4}$$

where C is the scene centre, M is the set of models in the scene and K_m is the set of keypoint positions for model m. Then the base distance can be computed using the following equation:

$$D_{base} = \max_{m \in M} \max_{k \in K_m} d_{k,C} + (abs(k_y - C_y) / \tan(FOV_{vertical})) \tag{3.5}$$

where $d_{k,C}$ is the distance between keypoint k as defined in Equation 3.6 and the scene centre C. The $FOV_{vertical}$ is the vertical field of view and is constant during the synthesising. The value is approximately 28.8 for the images that I synthesised as it is based on the resolution of the camera view in Unity. The $d_{k,C}$ is defined as:

$$d_{k,C} = \sqrt{(k_x - C_x)^2 + (k_z - C_z)^2} \tag{3.6}$$

where $d_{k,C}$ is the Euclidean distance between the keypoint, k, and the scene centre, C, while ignoring the y axis (the height). To put the definition of the base distance in words, it computes the minimal distance between the camera and the scene centre such that all the keypoints are inside the camera frustum, regardless of the horizontal degrees. It is also important to mention that it does not take the other variables into account. This minimum distance is the maximum distance required based on the Euclidean (2D, ignoring y-axis) distance between a keypoint and the camera in combination with the distance required between the camera and said keypoint, if the keypoint and scene centre are aligned (centre is perfectly behind keypoint), such that the keypoints are inside the camera frustum.
An example of this alignment can be seen in Figure 28 where the dot in the nearest

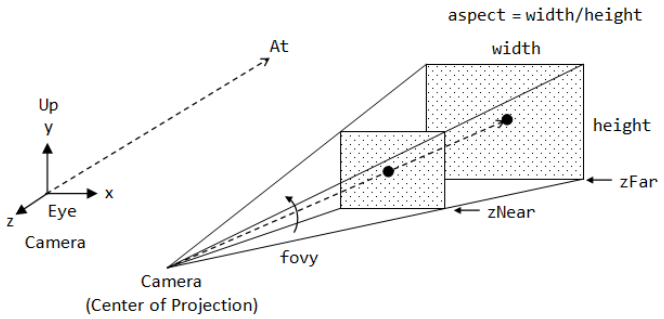


Figure 28. An illustration of a camera's field of view (Hussain, 2018).

rectangle is the keypoint and the dot in the farthest rectangle is the scene centre. Here the y-coordinate of the keypoint is equal to the scene centre's y coordinate, but this is often not the case. The vertical field of view, indicated by the 'fovy' in Figure 28 is the angle between the centre in the (near) rectangle, in this case, the keypoint/black dot, the camera itself and a point on the (near) rectangle at the top.

The second group of camera variables are the distance and height variables. These two variables are multiplied by the height or distance to the camera. Here the height only changes the y value of the camera's position while the distance variable changes the distance between the camera and the scene centre and can affect the x, y and z values of the camera's position. This is because of the vertical degrees and the horizontal degrees. The idea behind these two variables is to add variation to the camera's position while not affecting the camera's rotation. To compute the camera position with the random height and distance multipliers, the D_{base} in equations 3.1, 3.2 and 3.3 is multiplied by the distance multiplier. For the height variable, the C_y variable in Equation 3.2 is multiplied by the height multiplier.

The third and last group of camera variables do the opposite of the second group, they affect the camera's rotation but not the position and consist of the roll, tilt and yaw variables. The roll variable is similar to rotating augmentation as discussed in Section 2.8, where the camera is no longer aligned with the horizon. The tilt variable affects the angle of the camera on the x-axis as if the camera is tilted down to the ground or up into the air. The yaw variable rotates the camera around the y-axis as if the camera is turned towards the right or left.

3.3.2 Randomisation

One possible issue that could arise as a result of the relatively high number of images per scene is a pose estimation model overfitting to the human models' appearances. If this overfitting were to occur due to a lack of diversity in the synthesised images, this would make them far less useful and thus this ought to be avoided as much as possible. Although the set of humans has the same amount or more diversity than other synthetic datasets such as SIM by Ludl et al. (2018), as well as the actors in the Human3.6M dataset (Ionescu et al.,

2013)(Catalin Ionescu, 2011), the diversity in appearances is still far below the diversity of real humans which are present in (most) non-synthetics pose estimation datasets.
The most straightforward manner in which the diversity can be improved is to just increase the number of human models. This is made possible by the mass production plugin described in Section 3.1.2 and has the potential to increase the diversity of body shapes in particular. However, this would likely not solve the potential of overfitting entirely. This is because MakeHuman, and thus the mass production plugin, does allow for a lot of variation in body shapes, but this can sometimes lead to unrealistic or extremely rare models as described in Section 3.1.3.
Moreover, other aspects that can help increase diversity are not so easily changed in MakeHuman, e.g. clothing, hair and materials/textures. This means that increasing diversity via MakeHuman would not be enough to prevent overfitting as much as possible. There are, however, other issues with using many different human models which are created using MakeHuman. The largest issue is that, although the body shapes change, the differences between models are not as clearly visible in the 2D images. Some of the changes would be hidden by clothing or hair, while others are just only noticeable when zoomed in, for example, minor changes to the facial structures. Having many different models would also not necessarily mean that they can all be used, as there are 40 scenes and thus at maximum 80 models could be used.
One simple manner of increasing the diversity of the models' appearances is to randomise things such as materials and clothes, instead of randomising the models themselves. The primary aspect that gets randomised is the clothing of the models. Because MakeHuman removes faces underneath the clothing to prevent artefacts such as skin poking through a shirt, it is not possible to equip one human model with, e.g., shorts and trousers at the same time and then just switch between them in unity. Instead, there are multiple variations of one model, one variation for each clothing combination. Unity is able to switch between these different variations at run-time, and because the bodies of the variations are exactly the same, the pose of one variation can be copied to another variation without any problem. The only minor issue that could arise is an intersection of a looser type of clothing with the human model in certain poses. But, the intersection is small and thus often it looks like the body part just indents the clothing.
The materials of all the clothes are randomised with the upper body clothes having the most options. The upper body clothes have a probability of 50% to use a material based on images from the dataset by Medeiros et al. (2017). I used 1006 images from their dataset as materials with most of the images being bright and colourful. Meanwhile, the other half of the time the material is based on one of five white textures with a colour filter on top where the RGB values are ≤ 191 or 0.75 and randomly generated. I used these same five white textures for the materials of the lower body clothes, 100% of the time, but then the colour filter limits the RGB values to ≤ 127 or 0.5. I chose to use this because trousers are

often not that bright, usually a dark colour such as black.
The material of the shoes is not changed itself, but instead, a colour filter is placed on top of it, which can make it appear as if the material has changed. There is a 12% probability that a random colour is chosen such that the RGB values are all ≤ 204 or 0.8, this value was chosen to prevent the shoes from being too bright. With a probability of 48% the shoe gets any monochrome colour and the remaining 40% of the time the original material is used.
The accessories, eyes and lighting is randomised as well as this can be done easily and might help prevent overfitting. For the accessories it is just a matter of making them visible or invisible, the material for the accessories is not changed. The material for the eyes is selected from one of the options presented by MakeHuman while the light is rotated randomly to change the shadows.
It would also be possible to randomise the skin and hair colours but I decided to forego this as it would require a lot more manual labour to still keep the realism. In theory, you could just have a few different hair colours and assign one to a model, but then Unity would have to check first whether that specific colour is realistic in combination with the model's race. As for the skin colour, the facial features would stay the same and so changing the skin colour at random can make the models look less realistic. As I already had quite some variation, randomising the skin colour is probably not necessary.

3.3.3 Keypoint position

The position of people's keypoints is the most important part of the image annotation, thus Unity should be able to determine the positions accurately and efficiently during the synthesising. Fortunately, this is fairly simple for the keypoints corresponding to joints in the body. Unity can get the global 3D position of the joints by using the bones in the model's skeleton and then use this global position to compute the 2D position in the screenshot image using the Camera.WorldToScreenPoint method. This 2D position is used as the ground-truth position of the keypoint in the annotation. Meanwhile, the global 3D position can be used for 3D pose estimation when combined with the global 3D position of the camera.
Computing the position of the facial keypoints (eyes, ears and nose) is more complicated than for the body keypoints. This is because there are no bones or other parts in the human model that can be used to get the global position. My first approach was to try to get the global position of the model's vertices and faces that correspond to these keypoints and then use the barycentre of these to compute the centre of the eyes, ears and nose. However, Unity is unable to get the global position of these vertices during run-time and even if it did, it would be difficult to find which vertices are part of the nose for example. These vertices are not labelled, but instead, the mesh is stored as an array of vertices and an array of faces. Unity does allow you to get the positions of the vertices of a prefab, which in

the case of the human models is a model with its original unchanged pose. But without knowing which vertices belong to the eyes, ears and nose, this is not useful.
I resolved this problem using invisible clothing meshes, as discussed in Section 3.1.4, which serve as the facial keypoints. Because these meshes are positioned right on the eyes, ears and nose, and only contain the right vertices, these can be used to compute the position of the facial keypoints. Right before the synthesising starts, Unity creates a new gameobject and uses the offset of the invisible facial keypoint as its position. This offset is relative to the global position of the model as a whole and is defined as the barycentre of the mesh. The new gameobject's position is equal to the centre of the facial keypoint mesh and thus it serves well as a keypoint. The 3D and 2D positions of these new gameobjects can be retrieved in the same manner as for the body keypoints.

3.3.4 Occlusion detection

The second most important part of the annotation is the visibility flag which indicates whether a keypoint is visible or not. This visibility flag indicates how difficult it is to correctly locate the keypoint and also impacts the average precision as it is used in the OKS metric (see Equation 2.10 in Section 2.6.4). While the COCO dataset uses three options (not annotated, annotated but not visible and annotated while visible), I decided to use six options with different definitions. The definitions of the six options can be found in Table 11. First of all, unlike manually annotated images, all the keypoints are annotated as the synthesising method can compute the position of keypoint even if they are outside of the image bounds.

ID	Definition
0	The keypoint is outside the camera frustum (not visible in the image)
1	The keypoint is completely occluded by another person
2	The keypoint is completely self occluded
3	The keypoint is completely occluded, partially by itself and partially by another person
4	The keypoint is partially visible
5	The keypoint is visible

Table 11. The six occlusion options with their definition.

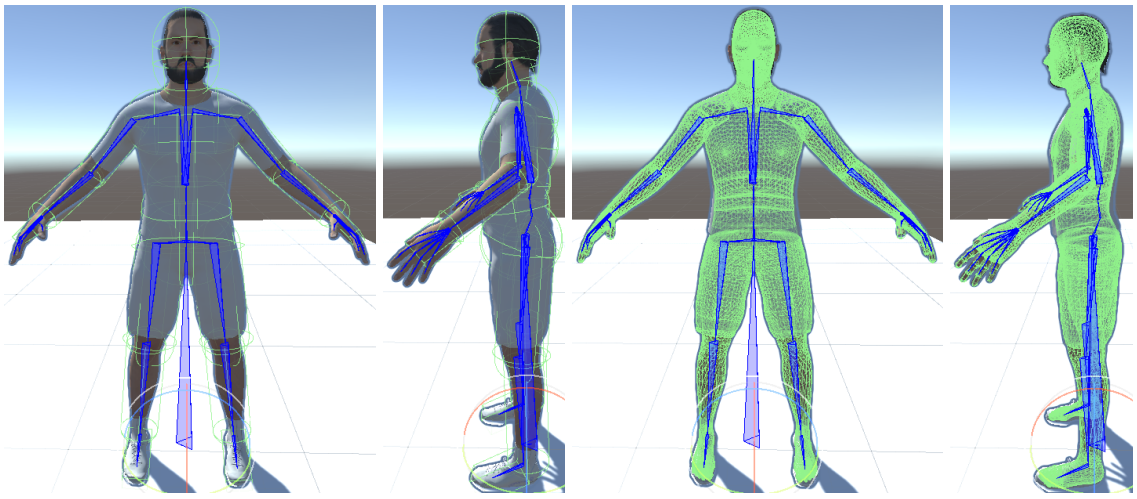
Keypoints that are outside the camera frustum, indicated by a 0, can be found by inspecting the position on the 2D image. Unity will in fact return three coordinates where the z indicates if it is in front of the camera or behind. When a keypoint is behind the camera then the keypoint is clearly not visible. If the x or y coordinates fall outside of the acceptable range of the screen (negative value or larger than the width/height respectively), the keypoint is not visible either.

Determining whether the keypoint is occluded, self-occluded or partially occluded can be challenging. Using only the keypoint does not allow us to determine whether it is partially occluded or completely occluded. To achieve this, I do not look at the keypoint alone, but I also use four helping points. These points are located such that a triangle consisting of this point, the keypoint and the camera has a 90 degrees angle. Two of these points are positioned to the right and left sides of the keypoint and the other two are above and below the keypoint with all the four helping points being located on a sphere where the keypoint is the centre and the radius is keypoint-specific constant. Keypoints corresponding to larger joints have a sphere with a larger radius, just like how the OKS uses a keypoint-specific constant, k_i.

First, the materials of the models in the scene are all changed to an unlit single colour material, with a unique colour for each model. Unlit means that it does not receive any shadows, hence the object is a smooth colour no matter the lighting and appears to have no depth. Ergo, all the pixels on the screen displaying the object have the exact same RGB values, except for some pixels at the border of the object. These will have a mixture of the object's colour and another object or background colour due to anti-aliasing. Secondly, the facial keypoints receive their own unique unlit colour. Then for each point, keypoint or helping point, the colour value of the pixel displaying the point is compared to the colour that it should have if it were not occluded. If the pixel has this colour then it is not occluded but it might be self-occluded. In either case, the method uses raycasts to try to determine whether it is occluded or self-occluded.

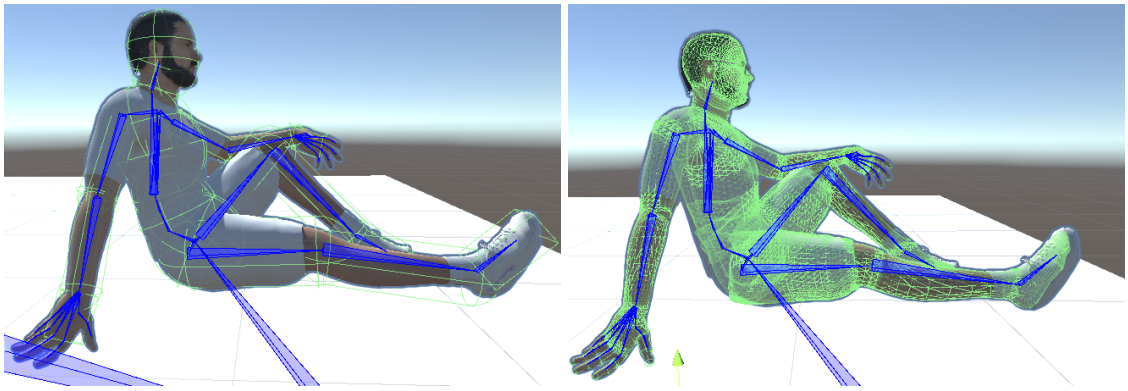
Raycasts are rays (thin straight lines) that travel from a starting point in a certain direction for a certain distance. A raycast works like using a laser pointer to point at an object, where the laser pointer is the start position (the camera is the starting point in the case of a raycast). If the object you point at is occluded by a different object (excluding all translucent and transparent objects), then the laser pointer's light will not be visible on the object you point at. The direction of the raycast is based on the position of the point relative to the camera while the distance is equal to the distance between the camera and the point. If the distance were smaller it would stop prematurely and perhaps miss the occluding object and if it were larger then it might detect something behind the point. Regardless of whether the keypoint is occluded, the raycast hits an object with a collider of the model. Unlike a linecast, which travels from one predetermined point to another, the raycast does not stop at the first hit but returns all hits. This is necessary when the colliders are not perfectly fitting the model.

The usage of raycasts relies on the presence of colliders, as those are the only objects that it can detect. At first I tried using a combination of capsule and box colliders to cover the most important parts of the body. I tried fitting these capsules and boxes to the model based on the length of body parts, but the result was not perfect (see Figure 29 and 30). Either the colliders were larger than the body parts they were attached to or parts of the



(a) (b) (c) (d)

Figure 29. (a) Front view of fitted capsule colliders (green). (b) Side view of fitted capsule colliders (green). (c) Front view of mesh colliders (green). (d) Side view of mesh colliders.



(a) (b)

Figure 30. (a) View of model with capsule colliders sitting. (b) View of model with mesh colliders sitting.

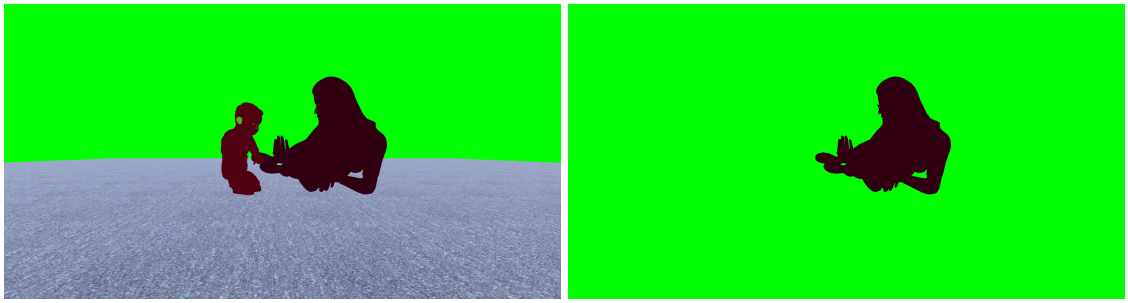
body would fall outside of the colliders. The first case is a problem because this can give false positives, as a raycast would detect the collider of a limb while that limb does not actually occlude the keypoint that the raycast travels to. The second case would result in some false negatives, a raycast would fail to detect the occlusion of the keypoint because part of the body covering the keypoint falls outside of the collider. An additional issue was that the colliders had to be sex-specific due to slightly different positions of the bones inside the model.

My solution to this issue was to use invisible clothing meshes that act as colliders. The idea is that, in Blender, meshes are created based on the movable body parts. So, e.g., there would be a mesh based on the upper arm that would consist of vertices at the exact same place as the upper arm of any model. Because the vertices of clothing meshes are based on the corresponding vertices of the model, this mesh will fit any model perfectly. Creating these clothing meshes is also not too difficult, it takes around 5 minutes per mesh. I created

meshes for the balls of the feet, feet, calves, thighs, hips, multiple for the torso, shoulders, upperarms, lowerarms, hands, neck, head without nose and the nose. With these clothing meshes as colliders, the accuracy of raycasts goes up as there are far fewer false positives and false negatives. However, this problem is not completely solved as these colliders are static meshes, while the body is morphed due to the rotations of bones.

3.3.5 Area and bounding box computation

The area and the bounding box are not essential for human pose estimation annotations, but they are helpful nonetheless and also make the OKS metric more accurate. The area is used by the OKS metric, see Equation 2.10, and when absent, the bounding box can be used to estimate the area. However, this will never be as accurate as humans do not appear as rectangles on images, instead, they can take on complicated shapes, which means that the area is never equal to the bounding box. When the bounding box is not present, an estimation can be computed by finding the tightest fitting box encompassing all the keypoints, possibly with some margin. This is not as accurate as the bounding box of annotations because parts of the body often extend further than the keypoints but this extension is not necessarily distributed equally. While the bounding box is not used by the OKS metric, it is still useful as top-down pose estimation models require a bounding box to train or test, which they use to crop the image to then predict the keypoint locations.



(a) (b)

Figure 31. Screenshots of (a) the scene during occlusion detection and (b) the scene during the area computation.

Computing the area and the bounding box can be done without any difficulty by changing the aesthetics of the scene and then loading the screen to a texture object. First, one person in the scene is selected to compute the area and bounding box, and then the model's materials are all changed to an unlit single-colour material. Next, the other person is turned invisible and the floor's material is changed to an unlit material with the same green colour as the background. Then the texture object consists of an image with a green background and one person in a single colour on top of it. Then to compute, it counts the number of pixels that do not have the same colour as the background and saves the extreme values which are defined as the most left, right, top and bottom positions.
It would be possible to count all the pixels where the colour values are the same as the

person, which would give a lower area and possibly a smaller bounding box. At first glance, it might seem as if these two methods do the exact same as the background is one solid colour and the person is another solid colour, but due to anti-aliasing, there are some pixels for which the colour values are a mixture of those two solid colours. The 2D screen cannot convey the same amount of details that the 3D world possesses and thus some details are lost. This combined with the anti-aliasing causes some pixels to have a mixture of the colours as the human model is visible in part of the pixel but not in the whole pixel.
I decided to include these pixels because there can be quite many of such pixels and they do show parts of the model. By including them the bounding box might be slightly larger than it should have been and the area is an overestimation. Excluding them would cause the bounding box to exclude small parts of the model and the area would be an underestimation. Overall, I suspect the impact is minimal as the bounding box is only used by the top-down models and while the area is used by the OKS metric, its effect is small as the metric uses the square root of the area. A larger area means that a pixel can be slightly farther away from the ground-truth location while still being considered a correct prediction.

3.3.6 Annotation process

Synthesising data is a repetition of important steps, such as computing annotations, where many steps rely on the previous step to work correctly and efficiently. Synthesising 43,200 images takes quite some time and thus it is important that not too much time is wasted. In general, what takes the most time but is not always necessary are the computations required for a new frame. Therefore, minimising the number of frames required can make synthesising more efficient.
Using the tasks mentioned in Table 12, the number of frames required to synthesise one image is four, one for taking the screenshot, one for detecting occlusions and two for computing the area and the bounding box as there are two people in each scene. The global initialisation is only performed once at the start of the synthesising process. The camera counter object is used to keep track of the camera's position and rotation as well as compute the next position and rotation. Once the camera has visited all the required positions and rotations, the camera counter is reset and a new scene is loaded.
The scene initialisation is done after a scene is loaded and takes up four frames because certain tasks can only be done after others have been finished. Randomising the models' materials can only be done after the model variations have been added, which relies on the facial keypoints being updated (as they are then copied to the model variations). The updating of the facial keypoint in turn relies on the creation of the models' corresponding objects. Randomising the materials and lighting is done as the last task during initialisation to ensure that the models have acceptable materials, as they start off with completely white clothes. The actual change will only be visible in the next frame and thus it cannot be done during the same step as the picture is taken.

Step	Tasks
Global initialisation	Fill the clothing material list used for randomising Initialise the camera counter object
Scene initialisation	Create lights Changing of floor material Creating list of models and corresponding objects Update facial keypoints Add model variations to the scene Set the start position of the camera Randomise models' materials and lighting
Image creation	Generate image file name Take screenshot Update models' materials to unlit
Occlusion detection	Determine occlusions Update floor material to unlit Hide all but one model
Computing area and bounding box	Determine bounding box for visible model Compute model's area
If computing for last person	Write the annotations to a file Randomise material and lighting Revert floor material change Show both models Update camera counter object Update camera position and rotation

Table 12. The steps taken during the annotating process.

Once a scene has been initialised, the last four steps are repeated till the camera counter indicates that a new scene needs to be loaded. First, the file name for the image (and annotation file) is generated based on the camera counter object. As an example, 'H10_V0_D0_He1_R0_T0_Y1.png' can be read as horizontal rotation step 10, vertical rotation step 0, distance step 0, height step 1, roll step 0, tilt step 0 and yaw step 1. This indicates that the horizontal rotation is in its tenth step, or rotated 120 degrees from the start, some extra random height has been added and some random yaw as well. The other variables, with a 0 behind the letter, are in their zero step and have not changed. The reason for using such a file name pattern is that it makes creating subsets of the synthetic data really easy. If, for example, I want to exclude all the images with a camera roll (roll rotation) then I can just ignore the files where 'R0' is absent. Figure 42 shows an example annotation file.

As discussed in Section 3.3.4, the models' materials have to be changed to an unlit material so this is done after taking the screenshot. Then in the occlusion detection step, the occlusions are detected as discussed in Section 3.3.4. In this step, the floor's material

is changed to an unlit material with the same colour as the background to help with computing the area and bounding box, which is done as mentioned in Section 3.3.5. Then when the area and bounding are being computed for the last person, an extra step takes place. Here all the annotations are written to a text file, materials and the models' visibility are changed for the next screenshot and the camera is made ready for the next screenshot. Then, the image creation, occlusion detection and computing area and bounding box are repeated.
The annotation file consists of the multipliers for the distance and height as well as the added roll, tilt and yaw rotation angles. The camera's global position and rotation are also specified, the same is true for the light's rotation. Then, the keypoint names, positions (2D and 3D), and their occlusion identified are specified. Lastly, the file specifies for each model: the name, the bounding box, visible accessories, area and the names of the clothing materials.
Because training a pose estimation model requires a single annotation file, I wrote a simple script that reads all the annotation files and then creates a single annotation file with a COCO format. The reason for not putting it in a COCO-like annotation file from the start is because of all the extra information in the annotation files, such as the 3D position and material names. This would clutter the COCO-like annotation file.

3.4 Annotating test set

The goal of synthesising pose data is to improve a pose estimator model's performance by training it on the synthesised data. As the scenes for the synthetic data consist of (indoor) adult-child interactions to improve the performance of pose estimators on adult-child interactions, the test set should ideally contain similar images. As mentioned before, there is currently not a good test set available for adult-child interaction, hence, there is a need to annotate the image to create a test set. For this purpose, I used Coco Annotator (Brooks, 2019) which simplifies the annotation process to a certain extent.
The first step is to create a new dataset in Coco Annotator and add some images. You can then create the keypoint labels and start annotating the images. During annotation, I placed the keypoints on the joints if I can say with enough certainty where the joint is located in the image. With visible keypoints this is straightforward but when a keypoint is occluded this can become rather difficult. Lastly, if a keypoint was outside of the image bounds I did not annotate it as it is not visible. Coco Annotator then set the visibility flag of these keypoints to '0', to indicate that they are not annotated and not visible.
Besides the location of the keypoints and their visibility flags, the OKS metric also requires the area of each annotated person (see Section 2.6.4 for OKS definition). Normally the area is computed based on the segmentation of the person, but as I did not intend to annotate this I had to compute it differently. I decided to use the area of the tightest fitting box around

the keypoints as the area for the annotated person. This is not a perfect solution as for some annotated people the area is an underestimation (when few keypoints are annotated) while for others it is an overestimation (when a lot of background is present inside this box).
Testing top-down models on the test set requires the bounding box for each person as those models crop the image to each person before predicting the keypoint locations. Here I could use the tightest fitting box around the keypoints again, but this would be especially problematic when few keypoints are annotated. The bounding box would only contain a small part of a person while more is visible in the pre-cropped image. Instead, I tried to find a good margin to add to each side of this tightest fitting box which then included parts of the person that would otherwise be cropped out (e.g. top of the head, feet and hands).

3.5 Training and testing pose estimators

The purpose of synthesising the adult-child interaction data is to improve the performance of current state-of-the-art pose estimators on (real) adult-child interactions. The models can be finetuned by training on the synthetic data, which should improve their performance. To test whether the training on synthetic data helps, the performances on a test set, before and after training, can be compared. The first step is to test the state of the models on the test set. Then those models can be finetuned by training on a combination of synthetic data and Coco training data, after which they can be tested on the same test set. The synthetic data is combined with the Coco training data to prevent overfitting on the synthetic data.
One might assume it is unfair to compare the finetuned with the non-finetuned versions as the finetuned versions have had more epochs to train. It would be possible to train those non-finetuned models on Coco training data alone (so no synthetic data) to compare the impact. Although this might work and result in a performance improvement, it is not advised. The reason for this is that those non-finetuned models are state-of-the-art pose estimator models which were trained on the same Coco training data. These models have been trained to increase performance as much as possible while preventing overfitting. This means that it is reasonable to assume that training the models on more epochs on the same Coco training data results in overfitting on the data. Therefore, I compare the models as they were trained by their creators with said models finetuned on a combination of synthetic and Coco training data.
Besides comparing the effect that finetuning has on a model's performance on the test set, I also researched what are good values for some of the training variables. These ablation experiments can give insight into what variables are most important for training a pose estimator on the synthetic dataset. The first experiment regards the standard data augmentation strategy and what effect changing it has. If some data augmentation method impedes the learning of the model I should be able to see this in the results with worse performance on the test set. Additionally, I look into what layers of the model should be

frozen such that it does not lose what those layers have learned.
Inspired by the article of Wang et al. (2019) about synthetic head poses, I would like to research the effect of adding Gaussian blur to the images on the performance. The synthetic images likely have less blur as the 3D human models are static, while images often have some motion-blur due to the people in the image moving. Therefore, the trained model might benefit from the blurred synthetic images as they are more similar to real images.
As for the dataset, I experimented with excluding some of the scenes as well as images with a specific variable value, such as the vertical degrees. It might be that certain images are redundant and that the models can be trained without those images while still obtaining similar scores.

4. Results

In this section, I discuss all the results of my research. I start by discussing the results of creating the 3D human models and which choices I made regarding their clothing as well as how many models I ended up generating. Next, I explain how synthesising a dataset went and the decisions I have made regarding variables related to the camera parameters. Certain images are less useful to use than others, so I discuss which images are included in the final synthetic dataset. As described in Table 11, I have made use of six visibility flags and I can use these to get an idea of what percentage of keypoints is occluded. Additionally, I discuss how the images are pre-processed before being used to add a fitting background image.
After discussing the synthesised dataset, I continue with discussing the Youth test set for which I hand-annotated several images from the Youth images dataset. I discuss some details of the images, and the two variants of the test set and I analysed the percentage of visible keypoints in this test set and compare it to that of the COCO validation set.
Lastly, the performance of some state-of-the-art models, finetuned on the synthetic data, on the Youth test dataset is discussed. These are arguably the most important results as they can show how effective the synthetic data is for finetuning a pose estimator to improve its performance on adult-child interaction data. I dove deeper into how and with what data the pose estimators ought to be finetuned to obtain better performance. This was done using several ablation studies looking into training parameters and the effect they have on a model's performance.

4.1 The mass produced models

Synthesising the data required me to mass produce the 3D human models, where the clothing also enabled more aesthetic diversity than any other aspect. The clothing for the models consisted of a romper (with long or short sleeves/legs) for the children only and a combination of a top and trousers for both the children and adults. The top is either a (short-sleeved) t-shirt, a (long-sleeved) sweater or a (long-sleeved) turtle neck, with the turtle neck only worn by adult models. The trousers have four options, loose trousers, tight trousers, loose shorts and tight shorts.
The clothing also had a few gender and age-group specific variants, which is to prevent protrusions and create a more realistic fit. The tight trousers and shorts are the only exception as they do not have any aesthetic issues due to the vertices of these meshes being so close to the corresponding vertices on the human model. The loose trousers have four variants (one for each male/female and child/adult combination) while the shirt,

sweater and shorts have three, one for the children and two gender-specific ones for the adult models. The child-only short-sleeved and long-sleeved rompers have gender specific versions as well while the turtle neck has a single non-gender-specific version.
In total there are ten clothing combinations for the child models, two rompers and eight t-shirt/sweater and trouser combinations, and twelve combinations for the adult models, t-shirt/sweater/turtle neck and loose or tight trousers/shorts. The reason that the models are not just equipped with all the clothing options at the same time is that each clothing mesh removes the faces of the model's body underneath the clothing. Therefore, although it is possible to turn a clothing mesh invisible in Unity, it is not possible to bring back deleted faces of the model's body. This means that if a model was equipped with trousers and shorts, making the trousers invisible means that the lower legs are invisible as well as the corresponding faces were deleted by MakeHuman.
As described at the end of Section 3.1.1, the tennisshoes caused some issues with the child models, therefore the options for the child model are shoes 5 and 6, which are provided by MakeHuman itself, unlike the tennisshoes which needed to be downloaded via the community tab. The adult models are all equipped with the tennisshoes as there were no issues for the adult models and because the shoes are white, the colour can easily be changed in Unity. As for the accessories, these are considered mandatory clothing as each model is equipped with accessories which are suiting for the age range. This means that all the models are equipped with glasses, the adult females with earrings and a ring and the adult males with a ring, a beard and a moustache. As none of the accessories removes faces underneath their meshes they can easily be shown and hidden inside Unity. This means that they are good candidates for extra randomisation while synthesising images.
The final batch of mass-produced models consists of 36 unique humans, of which 18 are adults and 18 are children. The distribution of race and sex is equal in both age groups and thus there are three unique models for each combination of adult/child, male/female and African/Asian/Caucasian. The child models have ten different clothing versions, two with a long or short romper and eight with the upper and lower body combinations discussed in Section 3.1.3. The adult models have twelve different clothing versions, which brings the total number of unique model variants to 396 with a total size of 9.2GB. Mass production of these 396 variants took about two hours.

4.2 Synthesised dataset

Synthesising a large number of images with enough variation required me to decide on the values for the camera variables, as these variables determine how many images are synthesised and how aesthetically similar they can be. The variables for which I had to determine the values are discussed in Section 3.3.1.
Of these variables, the horizontal and vertical rotation are most important as they deliver the

most aesthetic variation. I decided to use a step size of 12 degrees for both the horizontal and vertical rotation, just like Liu et al. (2019), with a restricted range for the vertical rotation of [0, 24] (Figure 19 gives an idea of how the camera moved around the people). The reason for this small range is that using values higher would place the camera in angles (bird's eye view) that are not present in the test set and are also not that common in the real dataset. The values for these two variables are always deterministic to ensure the presence of enough diversity. The distance and height variables have a standard value of 1.0 but they are also randomised for an additional round.

As for the third group of variables (tilt, roll and yaw), I chose not to use an additional round to randomise the tilt value as an acceptable range is already small to begin with. A relatively small tilt of around 30 degrees could already result in the models being outside of the image bounds and tilts of only a few degrees would not change the perspective that much. Hence cropping such that the top or bottom of the image is removed could be done after synthesising and would have almost the exact same result. The roll and yaw were used during synthesising with both of them having a single randomised round. For the roll, the ranges are [-30, -10] and [10, 30] and for the yaw, the range is recomputed every time, such that no (additional) keypoint fall outside of the camera frustum.

These variables with their defined values result in 30 different values for the horizontal rotation, 3 for the vertical, 2 for the distance, 2 for the height and 3 for the roll and yaw. As the roll and yaw are not combined, I synthesised 1080 unique images ($30 \cdot 3 \cdot 2 \cdot 2 \cdot 3 = 1080$) for each scene, or 43,200 images in total. Some example images of the scenes can be found in Figure 32.

Creating the scenes that are used for synthesising took me approximately 16 hours in total, which is 24 minutes per scene. Certain scenes, with skin contact, for example, took longer while scenes, where the models do not touch each other, took less time. To make the rotating and positioning of models in the scene somewhat quicker, I rotated and positioned a few models that could serve as pose examples. If I wanted to update a model's pose such that it would be sitting I would look at these pose example models and copy their poses. Because each model has a different-sized body part, copying a pose is not enough to create a good scene. After the pose was copied I still needed to adjust the limbs to make the pose fit well with the environment. Without this, the foot of a model could be inside the floor instead of on top of it or it could have the model floating in the air. Moreover, I would often adjust the pose to create a new and unique pose, the pose example served as a starting point and meant that I could focus more on finer rotations.

In addition to the 40 scenes used for this dataset, I also created 8 scenes which were not used for synthesising. These scenes were created before I knew what the test set would look like and they contained models of older children replacing the older child model with an infant was not as easy. Therefore, I opted to just create new scenes. The scenes that were excluded are scenes 10, 13, 14, 17, 19, 21, 24 and 29.

The images have a resolution of 1200×634 and have an average size of 820,005 bytes

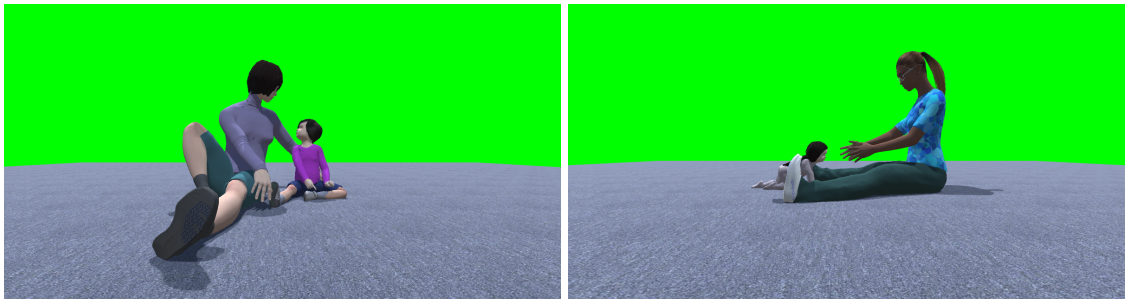


Figure 32. Images of scenes 6 and 35 respectively.

(∼32.99 GB in total). These images also contain some images where none of the keypoints of the child is visible due to the adult occluding the model. No interaction is visible in these images and thus I decided to exclude these from the final synthetic dataset's annotation file. There are 2629 images where all the child's keypoints are occluded and thus the dataset contains 40,571 images.

I also created a subset where all the images have a non-zero roll value (indicated by 'R0' in the image file name) with 27,042 images. The reason for creating this R0 subset is because the pose estimation models that I tested all use rotation augmentation and this augmentation gives the same result as the camera roll in Unity. If a rotation augmentation is performed on an 'R1' image this can result in a doubly rotated image and could potentially harm the performance of a pose estimator. For the experiments described in Section 4.4 I have used this R0 subset unless specified otherwise.

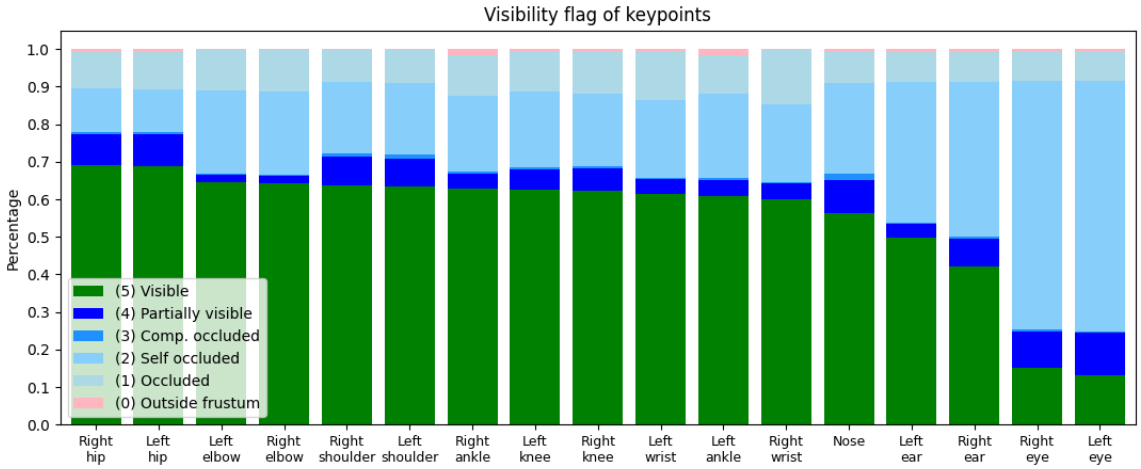


Figure 33. The keypoints of the synthetic data set, R0, with 54,084 people visible.

Training on the synthetic data combined with COCO training data requires the annotations to be converted into the Coco format. All the keypoints with an ID of '0' (outside camera frustum) kept their visibility flag, the keypoints with an ID of '5' (Visible) were changed to a '2' (Annotated and visible) and the remaining keypoints got a '1' as their visibility flag. The ground-truth keypoint locations in this Coco annotation format are rounded to

integers as this is also the case in the Coco training annotations. The area and bounding box are copied without any adjustments.

Figure 33 shows the distribution of the six visibility flags defined in Table 11 per keypoint. Most keypoints are visible in a majority of the images with the eyes being the clear exceptions as they are only visible in less than 20% of the images. Nevertheless, this exception is not that surprising considering that the eyes, unlike ears and the nose, are inside eye sockets in the skull. Additionally, there are some differences between the left and right sides of certain keypoints with significant differences such as the ears. This difference is likely due to the poses used in the scenes as there are only 80 people/poses and so there is not as much pose variation as in a non-synthetic dataset. Fortunately, this imbalance does not cause any issues as the pose estimators can use flipping augmentation and flipping at test time.

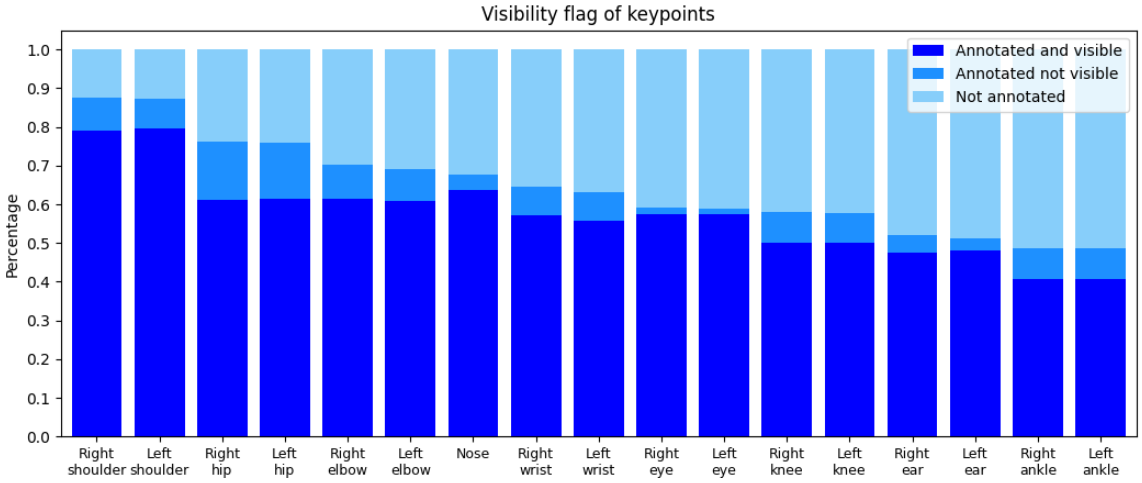


Figure 34. The keypoints of the Coco training set, with 149,813 people visible.

When we compare Figure 33 with Figure 34 we can see that the keypoints in the Coco training set have a more similar distribution of visibility flags. Far fewer keypoints in the synthetic data are not annotated (outside the camera frustum) thanks to the automatic annotations and the camera randomisation variables. There also appears to be less occlusion in the Coco training set as the shoulders are annotated and visible in more than 75% of the images while the most visible keypoints in the synthetic dataset, the hips, are visible in less than 70% of the images. Nonetheless, the Coco training dataset has a significantly higher percentage of visible eyes. A possible explanation for this is that people are more likely to take pictures of others when the people in the image are roughly facing toward the camera. For the synthetic data, this assumption does not hold due to the static interval of the horizontal rotation of 12 degrees.

The last step before using the images to train a pose estimator is to replace the green background with a background-filling image. As I test the fine-tuned pose estimators on images from the Youth images dataset, I used images from the empty (CPI) room where all the videos were captured. This means that the background of the synthetic images and

Figure 35. Images from the empty CPI room.

the test images are similar. Two example images of the CPI room can be seen in Figure 35. Results of replacing the green background can be found in Figure 36.

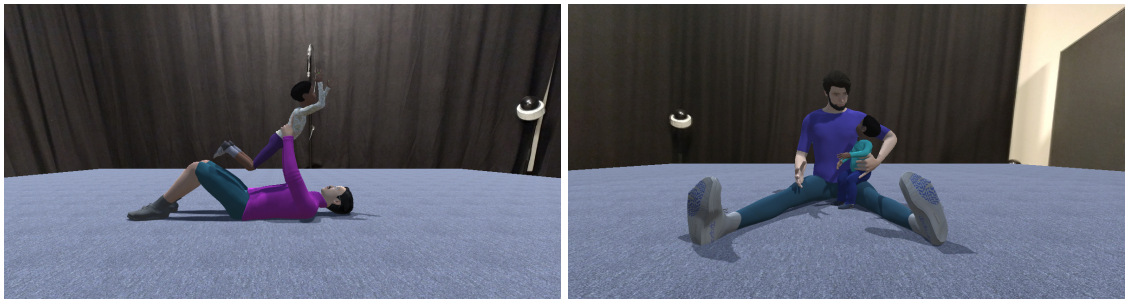


Figure 36. Images of scenes 11 and 33 where the background has been replaced.

4.3 Youth images test set

The Youth images test set consists of 520 images which are combined into 130 image files, four images per file. The images usually consist of one adult and one child but there are some exceptions, e.g., one image contains an additional adult in the background sitting on a chair. In many images, the child (and adult) are playing with some toys while they are sitting or lying on the floor. The images are based on four different camera viewpoints, all contributing to 130 images.

The images are not all of the same quality, there seem to be a few monochrome images and some images appear to be darker than others. As for the resolutions of the images, there are 148 images with a resolution of 704×576, 180 with a resolution of 960×540 and the remaining 192 images have a resolution of 640×400 (the files have resolutions of 1408×1152, 1920×1080 or 1280×800).

In these images, there are 1002 people annotated with a small majority of the annotated people being adults because in some images the child is completely occluded or outside of the camera frustum. These 1002 people have a total of 11,361 annotated keypoints or 11.34 keypoints per person on average. Annotating one image took, on average, 1.5 minutes and annotating all the 520 images took me approximately 13 hours.

As can be seen in Figure 37, certain keypoints are more likely to be occluded or not annotated. The keypoints are sorted from annotated in most images on the left side to

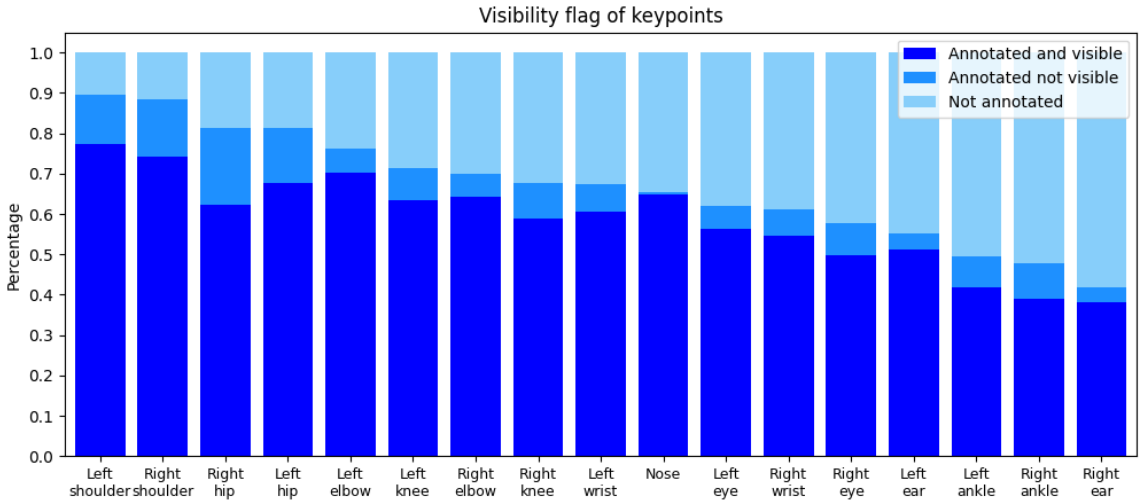


Figure 37. Visibility flags for the keypoints of the test set.

annotated in about half of the images on the right side of the Figure. The shoulders are most often visible in approximately 75% of the images and annotated in approximately 89% of the images. On the other side are the ankles, which are relatively often outside of the camera frustum and thus cannot be annotated properly. The ankles are annotated in ∼49% of the images while the right ear was only annotated in ∼42% of the images. All the facial keypoints have a lower chance of being visible as they can easily point away from the camera. While an eye can be annotated if it is not directly visible based on the location of the other eye and nose, this does not apply to the nose itself. I have only annotated it in three images where it was not visible because it is difficult to annotate the nose when neither the nose nor the eyes are visible. I could make a rough estimate but I chose to forego this as it would lower the quality of the annotations.

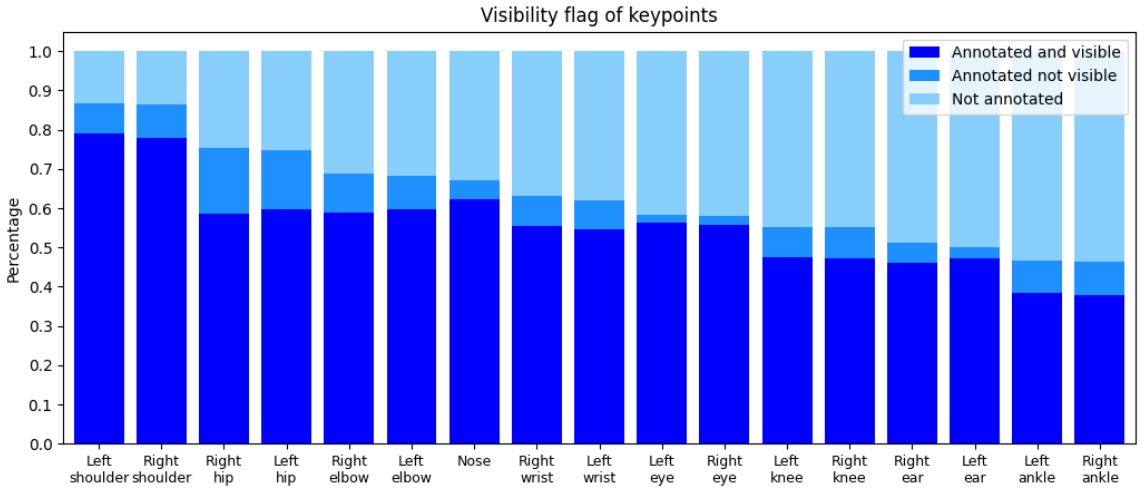


Figure 38. Visibility flags for the keypoints of the Coco validation set.

This distribution of visibility also affects the pose estimation results that I discuss in the

next section. Because there are twice as many left shoulders visible compared to right ears, this means that a model struggling with right ears has an advantage over a model struggling with left shoulders. However, such an unequal distribution of visible keypoints is not too uncommon. When we compare the distribution of visible keypoints of the Youth image test set (Figure 37) with the Coco validation set (Figure 38) we can see that the two distributions are actually fairly similar. The COCO validation set has 6352 people annotated spread over 5000 images versus the 1002 people in 520 images. The order of the four most commonly annotated keypoints is the same and the four least commonly annotated are the same as well, albeit in a slightly different order. One noticeable difference is that the knees are more often annotated in the Youth test set than in the Coco validation set.

As I did not annotate a bounding box for the people in the image I used the tightest fitting box around the keypoints in combination with a certain margin on each side as described in Section 3.4. Using this margin ensures that most of the person is inside this box and gives top-down models a fair chance. There are a few images with very few annotated keypoints and thus, without this margin, the bounding box would not allow the top-down models to predict the keypoints accurately. As an example, there is one image where only the infant's left ankle and left knee could be annotated. These two keypoints are positioned almost perfectly horizontally next to each other, which means that the tightest fitting box around those keypoints would be 83×4. Such a small bounding box would allow neither pose estimators nor humans to determine what part of a body is visible.

To mitigate these issues I tested a few margin options and eventually settled on using 4% of the image width on the left and right sides and 4% of the height on the top and bottom sides. For some annotated people this box is larger than necessary but I suspect this is not much of an issue as the top-down models do not predict multiple poses inside this box. The larger bounding box does mean that there is a higher probability of the other person being (partially) present inside the bounding box but this will always occur when there is inter-person occlusion.

Lastly, I have created and used two variants of the Youth images test set. The first variant consists of 130 image files with four images per file. The second variant, which I will refer to as the split variant, was created by splitting the files into four separate files such that each of the 520 images has its own file. The reason for this second variant is that bottom-up models do not crop the image before processing but scale it such that the image retains its original aspect ratio and is not larger than the model's input size. This means that an image file from the first variant with a resolution of 1920×1080 gets scaled down to 512×288 for the HigherHRNet, in other words, roughly every group of 4 by 4 pixels gets combined into one new pixel. This means a lot of information is unnecessarily lost and so the second variant ensures less information is lost.

4.4 Pose estimation results

This section discusses the performance of several pose estimators on the Youth test set. First I discuss the general training procedure and what variables are considered 'standard'. Next, I discuss the performance of four pose estimators (HigherHRNet-W32 and W48, HRNet-W48 and the Stacked Hourglass) trained via the described procedure in Section 4.4.1 on the Youth test set and compare their performance with the pre-finetuned performance of those models.

Lastly, I discuss a number of ablation experiments regarding the variables used for training and testing. Alternative data augmentation policies were tested and discussed, as well as differing the number of frozen stages and the synthetic-COCO training data distribution. Then I discuss reducing the number of scenes, the number of camera viewpoints via the vertical degrees variable and the amount of detail using Gaussian Blur. Lastly, I also look at the difference in the performance of the bottom-up models on the split and non-split variants of the Youth test set. For all these ablation experiments I used HigherHRNet-W32 for determining the baseline performance as well as the performance after the ablation.

4.4.1 Training policy

The training is done using two bottom-up models, HigherHRNet-W32 and HigherHRNet-W48, and two top-down models, HRNet-W48 and Stacked Hourglass, which are all implemented by MMpose (MMPoseContributors, 2020).

The training data consists of a combination of synthetic data and COCO training data (see Section 2.7.4). These two datasets, however, do not have the same size as the synthetic data only have 40,571 images, and the R0 subset 27,042, while the COCO training set has 64,115 (annotated) images. As for the synthetic data, I used the R0 subset (indicated by 'R0' in the image file name) as the pose estimator models to apply rotation augmentation during training.

To facilitate combining these two datasets with a certain distribution I updated the Pytorch dataloader used by MMpose to use the WeightedRandomSampler. This sampler allows you to set a certain probability for each datapoint in your dataset. The sampler will then randomly select datapoints from the dataset to fill up a batch where datapoints can be selected multiple times in one epoch. The standard approach that I used, which I refer to as 0.3, has a roughly equal probability of selecting a synthetic image as a COCO training image for each batch. This, combined with the larger size of the COCO training set results in a distribution of roughly 30%($\sim$29.7% to be more precise, (27,042/(27,042+64115)$\approx$0.297)) synthetic images in case of the R0 subset, hence I refer to it as 0.3. The other distribution that I used, 0.5 or 50-50, has roughly 50% of the images in each epoch being synthetic. This also means that it uses fewer images per epoch.

I used the data augmentation strategy provided by MMpose which is similar to the strategy

used by their respective authors (see Table 9 for a complete overview). The full overview of the augmentation methods used can be found in Table 13 with their respective probability (p-value) and their range specified. For the HigherHRNet models, the data augmentation is the same as the authors used (Cheng et al., 2020), which uses flipping, rotating and translating. The Stacked Hourglass and HRNet-W48 use the same data augmentation methods with different ranges and probabilities as well as the half-body augmentation (Wang et al., 2018). Additionally, the translating augmentation used by Stacked Hourglass and HRnet-W48 is not the standard translating, instead, they use a random shift of the bounding box centre, which translates the bounding box, and thus the input image, by at most 0.16 times the width and height respectively.

	Flipping	Rotating	Scaling	Translating	Half-body augmentation
HigherHRNet-W32	✓(p=0.5)	✓[±30]	✓[0.75, 1.5]	✓[±40]	
HigherHRnet-W48	✓(p=0.5)	✓[±30]	✓[0.75, 1.5]	✓[±40]	
HRnet-W48	✓(p=0.5)	✓[±40](p=0.6)	✓[0.5, 1.5]	✓[±0.16](p=0.3)†	✓(p=0.3)
Stacked Hourglass	✓(p=0.5)	✓[±40](p=0.6)	✓[0.5, 1.5]	✓[±0.16](p=0.3)†	✓(p=0.3)

Table 13. Data augmentation methods used by the models trained on the synthetic-Coco data. The p value indicates the probability of the augmentation being applied and when absent, the probability is 1. † random shift of the bounding box centre.

Both the HigherHRNet-W32 and HigherHRNet-W48 models use UDP (see Section 2.5.2) and the HRnet-W48 uses DARK (see Section 2.5.1) to combat a bias due to inaccurate encoding and decoding to and from the input image to the heatmap. The input size of the HigherHRNet-W32 and W48 is 512×512 while the HRnet-W48 uses an input size of 384×384 and Stacked Hourglass uses 384×384. The input of the Stacked Hourglass used here has a higher resolution than in the original article by Newell et al. (2016). The two HigherHRNet models use an output resolution of 128×256, HRNet uses 72×96 and Stacked Hourglass uses 96×96.

All four models use a constant learning rate of 0.000015 in combination with the Adam optimiser (Kingma and Ba, 2014). The HigherHRNet-W32, HigherHRNet-W48 and HRNet-W48 had their first three stages frozen during training while the Stacked Hourglass did not have any layers frozen. As the HigherHRNet models have a similar backbone as HRNet, with four stages in total, I just froze the same number of stages for those three models. The Stacked Hourglass model does not have any layers frozen. They were all trained using the step policy with a linear warmup of 500 iterations as this is the standard for models that are part of MMpose. Each model had a batchsize of 16 during training for 20 epochs in total.

Training for 20 epochs took around 18 hours for HigherHRNet-W32, 31 hours and 15 minutes for HigherHRNet-W48, 14 hours and 45 minutes for HRNet-W48 and 21 hours

and 50 minutes for the Stacked Hourglass. The testing used flipping at test time and single-scale testing. The top-down models processed the test images in roughly two minutes while the bottom-up models took three minutes.

Hardware specifications

All the training and testing were done on a server provided to me by the university. The server has two NVIDIA GeForce RTX 2080TIs with 12GB of VRAM. The CPU is an AMD Ryzen Threadripper 2950X with 16 cores and the server is equipped with 64GB of RAM. The code is run on a docker image with Ubuntu 18.04.01. Cuda 10.1, CuDNN 7.6.3, Python 3.7.7 and Pytorch 1.6.0. Initially, I tried training the models using both GPUs but this always resulted in crashes after a few epochs. When I tried training the same models on only one GPU this problem did not occur. I did occasionally run into an 'out of memory' error when the GPU ran out of VRAM but reducing the batch size resolved this issue most of the time.

4.4.2 Youth images test set

The four models (HigherHRNet-W32, HigherHRNet-W48, HRNet-W48 and Stacked Hourglass) are tested on the Youth image set (see Section 4.3) with the bottom-up models (HigherHRNet-W32 and HigherHRNet-W48) tested on the split variant where the 130 image files, all containing four images, are split into 520 image files without changing anything else. As for the testing, I used single-scale testing and flipping at test time. Both the HRNet and the Stacked Hourglass model use the ground-truth bounding boxes as described in Section 4.3.

Model	#Params	AP	AP^{50}	AP^{75}	AP^{M}	AP^{L}	AR
HigherHRNet-W32	28.6M	62.91	87.47	70.82	36.93	68.55	69.12
Finetuned HigherHRNet-W32	28.6M	64.72	87.74	73.54	39.61	70.31	71.64
HigherHRNet-W48	63.8M	64.92	87.77	73.33	40.40	70.40	70.82
Finetuned HigherHRNet-W48	63.8M	65.64	86.90	74.24	39.88	71.77	72.99
HRNet-W48	63.6M	78.03	93.69	86.21	55.59	82.60	80.65
Finetuned HRNet-W48	63.6M	79.17	93.90	86.00	59.57	83.88	82.00
Stacked Hourglass	9.4M	73.12	91.54	81.01	51.41	77.82	76.87
Finetuned Stacked Hourglass	9.4M	74.67	92.68	81.93	53.67	79.16	78.12

Table 14. Test results of the models on the Youth image test set. For the HigherHRNet models the split variant was used. All of the implementations of the models are from MMpose (MMPoseContributors, 2020).

Table 14 shows the performance of the four models before and after fine-tuning. For the

HigherHRNet-W32 the improvement is the largest with a 1.81 percentage point difference. The HigherHRNet-W48 model improved the least with only a 0.72 percentage point difference. The top-down models, HRNet-W48 and Stacked Hourglass, manage to improve their AP by 1.14 and 1.55 percentage points respectively. The AR of the bottom-up models improved by 2.52 and 2.17 percentage points for HigherHRNet-W32 and W48 respectively. For the top-down models, this improvement is smaller with 1.35 and 1.25 for HRNet-W48 and Stacked Hourglass respectively.
One noteworthy observation is the low AP^M scores for all the models, but especially for the bottom-up models. As described in Section 2.6.4, the AP^M refers to annotations of people with a scale of 32 or larger and smaller than 96 or an area of 1024 pixels or more and smaller than 9216. Lastly, it is worth pointing out that the HigherHRNet-W48 had a higher AP^M before finetuning.

4.4.3 Ablation experiments

Conducting the ablation experiments is done using the HigherHRNet-W32 model only as training all four models would take too much computing time. For these experiments, only a few aspects are changed and the exact change is discussed at the start to indicate what the experiment is meant to research. All the other aspects are the same as described in Section 4.4.2. As for the testing, the split variant of the Youth images test set is used and the model uses single-scale testing as well as flipping at test.

Different data augmentation policy

The first ablation experiment that I performed is related to the data augmentation methods used by HigherHRNet, specifically the rotation and scaling augmentation methods. In Section 4.2 I mentioned the exclusion of synthetic images with a non-zero roll augmentation (indicated by 'R1' in the image file name) because all four models already apply rotation augmentation during training. For this experiment, I used the whole synthetic dataset (40,571 images) to see if rotation augmentation combined with the 'R1' images hampers the training process. As for the scaling augmentation, the training images are scaled to fit inside a rectangle of 512×512 before they are processed by HigherHRNet-W32 (regardless of data augmentation). This means that the synthesised images (1200×634) are scaled down to 512×271 and the empty rows are filled. Scaling the image with a value < 1.0 increases the number of rows that are filled by the actual image and effectively partially crops out the sides. So I tested using a scaling range that has a lower minimum and lower maximum scaling multiplier.
Lastly, all the variants discussed here were trained for at least 10 with a batch size of 16 and a distribution of synthetic-COCO of approximately 50-50. The model had the first three stages frozen. After only 10 epochs it can already become clear if a certain model is preferable over the others and so more epochs are not always necessary.
The first variant of the HigherHRNet-W32 model was trained with all the standard data

augmentation methods as provided by MMpose. As can be seen in Table 13, it uses flipping, rotating ($\pm$30), scaling ([0.75, 1.5]) and translating ($\pm$40). The second variant used the same data augmentations but with 0 degrees rotation and the third variant used the standard rotating but an updated scaling range of [0.4, 1.1].

Variant	Epochs	AP	AP^{50}	AP^{75}	AP^{M}	AP^{L}	AR
Standard data augmentation	10	63.65	87.42	70.82	38.14	69.51	71.19
No rotation	10	63.44	87.39	71.11	37.72	69.24	70.98
Updated scaling	10	63.40	88.32	70.90	38.41	68.97	71.14
Baseline*	10	64.47	87.57	73.59	38.92	70.14	71.45
Baseline*	20	64.72	87.74	73.54	39.61	70.31	71.64
Pre-finetuning	0	62.91	87.47	70.82	36.93	68.55	69.12

Table 15. Test results of the models on the Youth image test set with 50-50 data distribution and all images (including R1) with different data augmentation methods. * used the R0 subset with 0.3 data distribution.

The baseline in Table 15 used the exact same data augmentation schedule as described in Table 13 and used the R0 subset instead of the whole synthetic dataset as well as a different data distribution. Table 15 shows that the data augmentation policy does not impact the performance as much. The three variants show some improvement over the pre-finetuning model.

Freezing stages

The HigherHRNet models consist of four stages which make up the backbone (HRNet) and have a head after these stages. As these stages and the head have already been trained for more than 200 epochs on the COCO training set and manage to perform relatively well on test sets, it might be worthwhile to freeze the parameters of certain layers to ensure that it retains their ability to perform well on all kinds of data. Not freezing any layers may cause the training to do more harm than good. Freezing layers can also decrease the likeliness of overfitting as part of the parameters cannot be trained and hence cannot overfit to the new training data.

I experimented with freezing a number of stages and trained the variants as described in Section 4.4.1 except for the number of epochs, they were reduced to 10 instead of 20. The first variant has only the first two stages frozen and a batch size of 8 instead of 16. I tried using 16 but this caused an 'out of memory' error as the variant required more VRAM than was available. By reducing the batch size to 8 this problem was solved. The second (standard) variant has the first three stages frozen with a batch size of 16, as described in Section 4.4.1, and also serves as the baseline. The third variant has all the four stages of the HRNet backbone frozen and so only the parameters of the head can be adjusted during

training. I used a batch size of 32 for this variant as it required less VRAM. Only freezing the first or no stages caused the same 'out of memory' error, even with a smaller batch size of 8 and so I was unable to try those options.

Variant	AP	AP^{50}	AP^{75}	AP^{M}	AP^{L}	AR
Frozen 2 stages	63.88	87.40	71.91	37.99	69.78	71.02
Frozen 3 stages	63.65	87.42	70.83	38.14	69.51	71.19
Frozen 4 stages	62.66	87.24	70.27	36.21	68.47	69.14
Pre-finetuning	62.91	87.47	70.82	36.93	68.55	69.12

Table 16. Test results of the HigherHRNet-W32 variants on the Youth image test set with a different number of frozen stages.

Table 16 shows that freezing fewer stages results in higher performance as the variant with 4 frozen stages is outperformed by the variant with 3 frozen stages is in turn outperformed by the variant with 2 frozen stages. The variant with 4 frozen stages has a lower AP than the pre-finetuning model by 0.25 percentage points. Meanwhile, the difference between the variants with 3 and 2 frozen stages is only 0.23 percentage points.

Synthetic to COCO training images ratio

In the previous two ablation experiments, I used a roughly equal distribution of synthetic versus COCO training data. I will call this a 0.5 distribution because the sampler selects a synthetic image with a probability of 0.5 and a COCO training image with a 0.5 probability as well. This means that, in the case of the R0 subset, each individual synthetic image has a probability of 0.5/27,042 to be selected and each COCO training image has a probability of 0.5/64115. In Section 4.4.2 I used a different distribution, one where each image roughly has the same probability to be selected and I will call this a 0.3 distribution because the synthetic images make up roughly 30% of the total amount of images (in the case of the R0) dataset that I am using here.

Besides the distribution, every other parameter is as described in Section 4.4.1 and the

Variant	AP	AP^{50}	AP^{75}	AP^{M}	AP^{L}	AR
0.5 distribution	63.75	87.31	72.38	38.85	69.42	71.04
0.3 distribution	64.72	87.74	73.54	39.61	70.31	71.64
0.3 distribution (10)*	64.47	87.57	73.59	38.92	70.14	71.45
Pre-finetuning	62.91	87.47	70.82	36.93	68.55	69.12

Table 17. Test results of the models on the Youth image test set where different data distributions were used, as described at the beginning of Section 4.4.3. * indicates that this variant was only trained for 10 epochs.

number of epochs is equal to 20. The 0.3 distribution is the same as used in Section 4.4.2 and serves as a baseline.

As can be seen in Table 17, the model trained with a 0.3 distribution performs better with a 0.97 percentage point difference in AP. Nevertheless, the model with the 0.5 distribution still learns enough to have a 0.84 higher AP score than the pre-finetuning model.

Less scenes

The R0 subset, as well as the whole synthetic dataset, consists of images of 40 scenes, as described in Section 4.2. However, it might be the case that using the images of fewer scenes results in the model having the same performance as using all 40. Or more generally, perhaps the model can achieve the same result by only using half of all the (new) training data. The reason I choose to reduce the number of scenes and not just the number of images is that creating additional images of a scene does not take much time, but creating additional scenes does require more work by a person and not just a computer. Therefore, creating additional scenes is more costly as the computer also has to create images for these additional scenes.

I created a new annotation file where only the images of half the scenes are present. These scenes were arbitrarily selected by a python script and uses images from scene 4, 7, 8, 9, 12, 15, 16, 18, 20, 22, 27, 31, 32, 33, 34, 35, 41, 43, 45 and 47[1]. Besides the reduced number of images, only the R0 images were used and the two variants in Table 18 as described in Section 4.4.1. For the 20 scenes variant the distribution of synthetic versus COCO training data remained the same, images from both datasets roughly have the same probability to be selected for a training batch.

Variant	AP	AP^{50}	AP^{75}	AP^{M}	AP^{L}	AR
20 scenes	63.85	87.05	71.94	38.12	69.54	70.63
40 scenes	64.72	87.74	73.54	39.61	70.31	71.64
Pre-finetuning	62.91	87.47	70.82	36.93	68.55	69.12

Table 18. Test results of the models on the Youth image test set where different data distributions were used, as described at the beginning of Section 4.4.3.

As Table 18 shows, the model does benefit from using all 40 scenes instead of just 20. Using the first 20 scenes improves the AP by 0.94 percentage points compared to the non-finetuned model, while the additional 20 scenes improve it further with 0.87 percentage points. The AP^{75} is increased more than the less strict AP^{50} which means that the model has improved especially at finer predictions. One noteworthy observation is that the AP^{M} improves a lot, 1.19 percentage points with the first 20 scenes and even 1.49 with an

[1]Scenes 10, 13, 14, 17, 19, 21, 24 and 29 were never added to the dataset.

additional 20 scenes.

Single vertical degrees value

The images in the Youth test set as described in Section 4.3 are taken from four camera points and thus there is not as much diversity in terms of height and the tilt of the camera (the angle between the camera's focal point and the floor). The synthetic images have a vertical degrees variable that determines this tilt and has one standard and one random camera height. This means that there is a bit more variation in terms of camera placement and angle in the synthetic images. Therefore it might be the case that only using images with a specific vertical angle can result in very similar performance and this experiment can show the most important vertical angle. The options used in the synthetic dataset are 0, 12 or 24 degrees, as described in Section 4.2. Of these three I suspect that using the images with 24 degrees ('V24' in the image file name) is the least useful as there the camera looks down onto the people more than with the other two options. This is not so much the case in the Youth images test set as three of the four camera viewpoints have a relatively low vertical angle. Therefore, I only look at using the images with a vertical angle of 0 degrees ('V0' in the image file name) or 12 degrees.
Besides the reduced dataset, the other training parameters are as described in Section 4.4.1 with 20 epochs in total and all the images are from the R0 subset.

Variant	AP	AP^{50}	AP^{75}	AP^{M}	AP^{L}	AR
Vertical degrees 0	64.65	87.42	71.99	39.28	70.19	71.07
Vertical degrees 12	64.61	87.88	71.71	39.04	70.18	71.22
Baseline	64.72	87.74	73.54	39.61	70.31	71.64
Pre-finetuning	62.91	87.47	70.82	36.93	68.55	69.12

Table 19. Test results of the models on the Youth image test set where different vertical angles were used.

Table 19 indicates that even if there is only one value, 0, for the vertical degrees, the model still manages to perform well. The same is true for when the vertical degree is 12, as the AP of 64.61 is close to that of 0 (64.65) and the baseline (64.72). Nevertheless, the baseline variant has a higher AP^{75}, 1.55 percentage points, than the 0 and 12 degrees vertical angle variants.

Gaussian blur

One clear difference you notice when you compare a synthetic image with an image from the Youth test set is the sharpness in the synthetic image. This is partly due to the green background replacement and the anti-aliasing that created a smoother transition between the green background colour and the synthetic model's colour. The images from the Youth test set, on the other hand, have a far lower sharpness and have more blur due to motion.

This motion blur might also be accompanied by lower sharpness due to the quality of the images/cameras. In synthetic images, it is very easy to identify the border of the model, while this usually is not the case in real images. Therefore, adding some blur might make the images look more similar to the images from the Youth test set and potentially helps the pose estimator models to obtain higher performance. Before the training started the training images received some Gaussian blur using openCV's Gaussian blur method with a kernel of 3×3 pixels and the default border type. This kernel size determines how blurred the image becomes and a 3×3 size seemed most reasonable after visually inspecting the operation. Examples of a scene with Gaussian blur can be found in Figure 39.
The training variables are as described in Section 4.4.1 with 20 epochs in total. The non-blurred variant serves as a baseline as this was used for the model in Section 4.4.2.

Variant	AP	AP^{50}	AP^{75}	AP^{M}	AP^{L}	AR
Gaussian blur	64.27	87.05	73.28	39.09	69.97	71.31
No blur	64.72	87.74	73.54	39.61	70.31	71.64
Pre-finetuning	62.91	87.47	70.82	36.93	68.55	69.12

Table 20. Test results of the models on the Youth image test set where the training images received no blur or Gaussian blur (3×3 kernel).

As can be seen in Table 20, adding Gaussian blur to the training images does not improve the model's performance but it does not hurt performance a lot either. The performance of the Gaussian blur variant has still improved by 1.36 percentage points over the pre-finetuning model.

Splitting the Youth test images

This ablation experiment regards the test set instead of the training. As described in Section 4.3 there are two variants of the Youth test, a split variant and a non-split variant. Using either variant, presumably, results in the same performance for the top-down models, HRNet-W48 and Stacked Hourglass, as top-down models crop the image to the bounding box of a person before the image is inputted to the model, where this crop is scaled to fit the input size (384×384 for the Stacked Hourglass model and 384×288 for the HRNet-W48). Splitting the test image files into its four images will not result in a different resolution for each individual image and the ground-truth bounding boxes are not affected either. For the bottom-up models, HigherHRNet-W32 and HigherHRNet-W48, this input process is different.
The image is not cropped to any bounding box, instead, the whole image file with the four images (1408×1152, 1920×1080 or 1280×800) is scaled to fit in the model's input layer with a resolution of 512×512. As described in Section 4.3, the images retain there original aspect ration and so each square of 2.75×2.75, 3.75×3.75 or 2.5×2.5 pixels becomes one

pixel for the non-split variant. This means a lot of detail is lost when the image is inputted into the model. However, if the image file with the four images is first split into four separate image files (as is done with the split variant), the squares become 1.375×1.375, 1.875×1.875 or 1.25×1.25 and thus more details remain in the image. These additional details can be used by the pose estimator to better predict the keypoint location and thus the performance of the bottom-up models should be higher on the split variant than on the non-split variant.
The models used for this experiment are the same as described in the sections above that are part of Section 4.4.

Model	Variant	AP	AP^{50}	AP^{75}	AP^{M}	AP^{L}	AR
HigherHRNet-W32	Non-split	58.60	86.85	66.23	29.28	64.25	65.86
HigherHRNet-W32	Split	62.91	87.47	70.82	36.93	68.55	69.12
HigherHRNet-W48	Non-split	61.16	87.22	69.26	30.71	67.18	68.26
HigherHRNet-W48	Split	64.92	87.77	73.33	40.40	70.40	70.82
Finetuned HigherHRNet-W32	Non-split	0.5959	86.47	68.39	32.75	65.05	67.83
Finetuned HigherHRNet-W32	Split	64.72	87.74	73.54	39.61	70.31	71.64
Finetuned HigherHRNet-W48	Non-split	61.33	87.18	68.53	34.62	66.63	69.47
Finetuned HigherHRNet-W48	Split	65.64	86.90	74.24	39.88	71.77	72.99

Table 21. Test results of the bottom-up models on the Youth test set split and non-split variants.

Table 21 shows that using the split variant of the test set results in better performance for all models. When you look at the differences between the finetuned and non-finetuned, the models tested on the non-split variant you can see that the models also seem to learn more for split images. The largest difference that you can see that is the result of using the other test set variant is in the AP^{M} column, where the score can go up as much as 9.69 percentage points.

4.5 Qualitative results

Here are a few example predictions made by HigherHRNet-W32 (subfigures a, c, e, g, i in Figures 40 & 41) and HRNet-W48 (subfigures b, d, f, h, j in Figures 40 & 41) on images that are not part of the Youth test set but are similar as they are from the same Youth set[2]. I tried to select more difficult images as these are the most interesting to analyse.
Just as in Section 4.4, the HRNet-W48 uses the ground-truth bounding boxes but here I annotated the bounding boxes instead of using the keypoints with a margin (there are no

[2]I was permitted to use screenshots, the ones in Figures 40 & 41, from the public video in my thesis.

keypoint annotations for these images). These ground-truth bounding boxes are also visible in the images as the green boxes. In the images in Figures 40 and 41 the keypoints of a single person are connected with lines. The orange-coloured dots are the facial keypoints (eyes, ears and nose), the green dots are the left-side body keypoints (shoulders, elbows, wrists, hips, knees and ankles) and the blue dots are the right-side body keypoints. Some keypoints can be missing in the images and this can be because the keypoint is not inside the image bounds or because the model was not confident enough in its prediction and hence it did not position a dot in the image.

The first image that was processed by the models can be found in Figures 40a and 40b. Here there is no inter-person occlusion and little self-occlusion, thus this image should be relatively easy to estimate for the pose estimators. HRNet-W48 performs well and even estimates the occluded left eye and left ear of the child. For the adult, only the occluded left ankle is not estimated. HigherHRNet-W32 was not able to find as many keypoints in the image.

The Figures 40e, 40f, 41c and 41d show images with some inter-person occlusion and a little bit of overlap in their bounding boxes. Figure 40f shows that HRNet-W48 has estimated the child's pose really well, even the left knee under the blue toy was found. As for the adult's poses, some keypoints are missing but based on the video I can confirm that it has correctly predicted the occluded left wrist. The lower half of her body is mostly self-occluded and neither model managed to predict keypoints (except for the slightly incorrect prediction of the adult's left hip). The HigherHRNet-W32 model predicted significantly fewer keypoints but the ones it did predict seem to be correctly located.

Figures 41c and 41d show a similarly difficult image where HRNet-W48 predicts most keypoints while HigherHRNet-W32 is less complete. Based on the different views from the video I can conclude that the HRNet-W48's prediction of the adult's occluded left elbow is significantly off. In reality, it was behind the right shoulder in the image. The model does manage to detect that the adult's gaze is directed more towards the ground while the child's gaze is directed towards the camera in the background.

Figures 40c, 40d, 41e and 41f are a bit more difficult as there is more occlusion and more overlap between the bounding boxes. Just as before, HigherHRNet-W32 predicts fewer keypoints but the ones predicted seem to be accurate. HRNet-W48 predicts many more but not all are equally accurate, e.g., in Figure 41f the left knee is predicted in the background, left and up concerning the left elbow and left hip. Both models also failed to accurately predict the adult's right wrist in Figures 40c and 40d, even though it is not occluded and does not blend in with the background either.

In Figures 40i and 40j, the child is severely occluded and only the right limbs are not occluded. HigherHRNet-W32 does not manage to detect the child while HRNet-W48 predict the child's right elbow and wrists correctly and mistakes the right leg for the left leg. Based on the video, I can conclude that the adult's left knee is also incorrect, in reality,

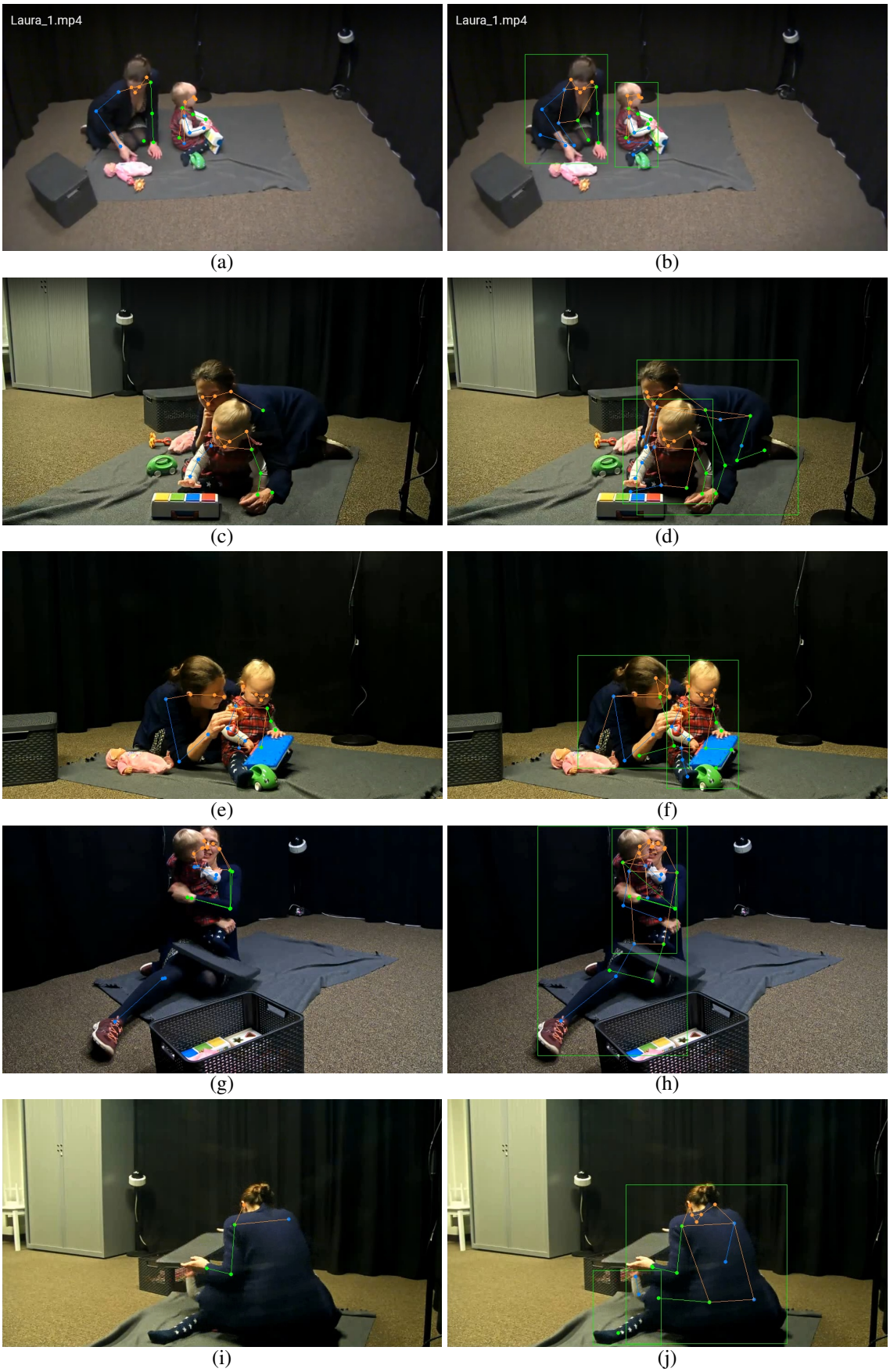
the knee should have been predicted somewhere between the elbows.
The images predicted in Figures 40g, 40h, 41a and 41b are very similar with a significant amount of occlusion and the same keypoints of the two different people being close together. In Figures 41a and 41b the adult's pose is predicted relatively well by both models but they both switched the left and right side of the child. In Figure 40g, HigherHRNet-W32 fails to locate most keypoints of the adult and seems to have predicted the child's nose on the adult's nose instead. In Figure 41b something similar happens, the child's left ear is placed on the adult's right ear by HRNet-W48. The models also use the adult's nose and eyes for the child's pose. It is a bit difficult to see, but the model also predicted the child's left elbow and wrist on the adult's elbow and wrist. Besides those keypoints, the predicted keypoints are relatively accurate.
Figures 41g and 41h are quite interesting as the adult's face is covered by the box's lid. HigherHRNet-W32 fails to (confidently) detect the adult as well as the child's right hip. HRNet-W48 performs a lot better as it does detect the adult and accurately predicts the lower body keypoints. Both the adult's and the child's eyes and nose do seem to be a bit off.
Figures 41i and 41j depict the child in a rare pose with the child's lower body being occluded. HigherHRNet-W32 merged the child's left elbow and wrist with the adult's pose and predicted the child's right wrist as the child's left wrist. As for HRNet-W48, I am not quite sure what happened but it looks as if it predicted the adult's pose double. Just as HigherHRNet-W32 it combined the child's left arm with the adult's right shoulder.

(a)

(b)

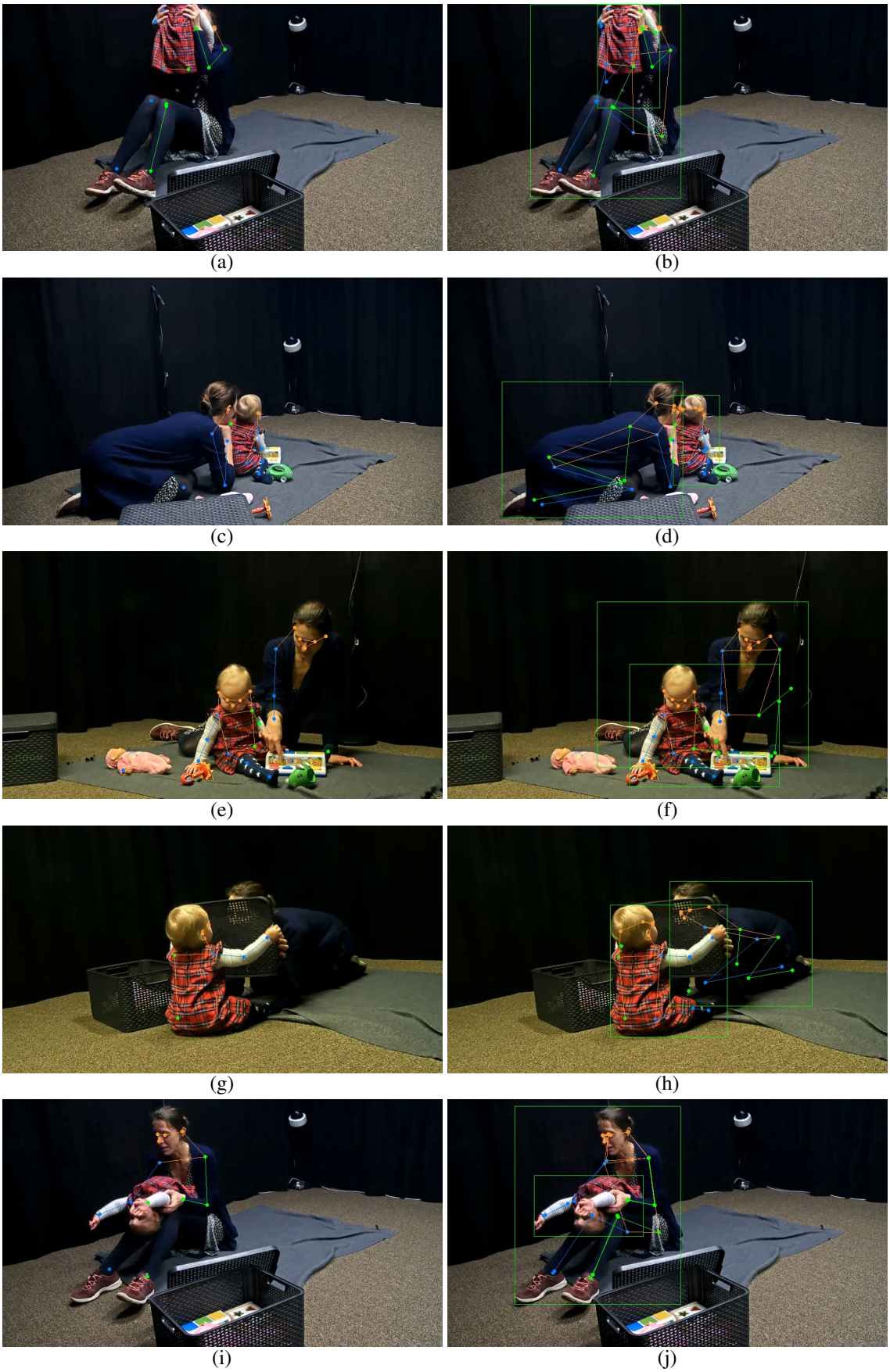
(c)

Figure 39. Three variants of the same synthetic image, (a) no blur, (b) Gaussian blur with a 3×3 kernel and (c) Gaussian blur with a 5×5 kernel



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 40. Predictions on images, from the public video, which are similar to those in the Youth image test set. I was permitted to use screenshots from this public video in my thesis. The left and right columns were predicted by the finetuned HigherHRNet-W32 and HRNet-W48, as defined in Section 4.4.1, respectively. HRNet-W48 used the ground-truth bounding boxes.



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 41. Predictions on images, from the public video, which are similar to those in the Youth image test set. I was permitted to use screenshots from this public video in my thesis. The left and right columns were predicted by the finetuned HigherHRNet-W32 and HRNet-W48, as defined in Section 4.4.1, respectively. HRNet-W48 used the ground-truth bounding boxes.

5. Discussion

In this section, I discuss the results from the previous section and interpret them where possible. First I discuss the results of mass producing models ad what could be improved. After that I discuss the synthetic dataset with some of its advantages and some points of improvement. Next, I wrote about the annotating of the Youth images test set and the quality of the data. Lastly, I discuss the results of the pose estimation models and how effective the synthetic data can be.

5.1 The mass-produced models

The synthetic dataset turned out rather well with the images portraying a lot of interaction with a significant amount of aesthetic variation. The 36 models used to create the interaction scenes do have some diversity, especially in height, but smaller changes could be made to increase the diversity. The ranges for certain variables had to be kept relatively strict as I did not have time to figure out the optimal range for each variable. The diversity in terms of sex, age and race are better.
There are equal distributions in terms of sex, age and race and this is an improvement compared to the older synthetic dataset used for pose estimation where they were not able to include as much diversity. Datasets such as SIM by Ludl et al. (2018) and the ones created by Sarafianos et al. (2016) and Liu et al. (2019) used far less unique models and mostly caucasian ones. mass-producing the models used to synthesise the data resulted in more problems than I anticipated. Programming a plugin for MakeHuman to mass produce models took more time than I anticipated as I was not familiar with MakeHuman's source code. Creating the clothing meshes, on the other hand, was not as difficult as I expected. Creating one shirt might have taken a bit longer, but then creating a version for females based on that one went well. There is also variation in the clothing meshes such as loose or tight trousers and there are options in terms of sleeve/leg length. It could still be improved as real data contains even more variation, but the variation in clothing is not the major shortcoming of the dataset.

5.2 Synthetic dataset

The synthesised images have a lot of variation in terms of lighting and materials, such as clothing. However, I am not sure whether the variation in the clothing materials is a good representation of clothing in the Youth images test set. There are also a lot of bright colours used as clothing materials. These materials do have a lower probability of being

used but I did not experiment with using certain distributions of dark versus bright colours. Additionally, there is no intra-person hair variation, the hair nor its material is randomised. There were also a limited number of haircuts and this lack in hair diversity might impact a pose estimator's performance.

The relatively low diversity in poses is the major limitation of the dataset. Because the poses of the models in each scene are static, there are only 80 unique poses while a real dataset, such as COCO or CrowdPose, contains at least a hundred times as many unique poses. This lack of pose diversity is a common problem in synthetic datasets. Many of them are based on motion capture data which generally has a limited amount of actions and usually few unique actors. By synthesising images the limited number of unique actors can be compensated relatively easily by just using many different 3D human models and clothing variations.

Using many different camera angles can give the impression of more pose diversity as the 2D distances and rotations between the keypoints change as the camera moves around. This camera randomisation is not too difficult while being rather useful for improving aesthetic diversity. Some of the camera variables such as the horizontal and vertical degrees were based on previous work by Liu et al. (2019)

The synthesised data was automatically annotated which is not only more efficient than hand-annotating images, but is also more consistent and accurate as the annotations are computed based on a combination of 2D and 3D information available to Unity. Hand annotating images relies on 2D information only and human annotators are not always consistent or accurate. This is the reason that the OKS metric (see Section 2.6.4) includes the k_i constant. Annotating the keypoints via the joints inside the 3D human model is easy and reliable, while the facial keypoints via invisible clothing meshes are more difficult as it required more work like creating the invisible clothing meshes and updating their location at the start of the synthesising process. However, these static keypoints are really useful for determining which keypoints are occluded and whether some keypoints are outside the camera frustum.

This information was used to move the camera around and ensure that enough keypoints are inside the frustum as pose estimators cannot learn as much from keypoints outside the camera frustum. Determining occlusions using a combination of the unlit coloured models and raycasts results in accurate occlusion detection as well as more information about the type of occlusion. This extra occlusion information is not used by the pose estimator but can be useful for other applications.

Lastly, the area computation is made really easy thanks to the option to colour objects with unlit colours. To compute the area after everything except one model is made a single colour, the program can just count the number of pixels that are not that specific colour and it has computed the area accurately. The bounding box is computed in a similarly straightforward manner and results in precise annotations.

5.3 Youth images test set

Annotating the image was more difficult than I anticipated. I did expect a lot of occlusions but because there are relatively many images where keypoints are outside the image bounds and cannot be annotated. There were various images where I would like to place a keypoint just outside the image bounds as that was the location where I would have annotated the keypoint if it was visible. For example, in one image I could see the lower eyelashes on the top of the image and so the eyes were a few pixels above the top. Annotating them would be incorrect as the eyes are not inside the image bounds. But not annotating them means that the model cannot use those keypoints to test its performance.
In other cases, the annotating was made more difficult due to the lighting conditions. The images are not that bright which, combined with dark clothes and dark background, can make it difficult to locate the joint. In some cases, the person can blend in with the background and the pose estimators struggle with predicting the poses of these people.
The COCO annotator tool could also be improved upon as skipping a keypoint, which is outside the image bounds or cannot accurately be located, requires multiple mouse clicks (more than just annotating a point). Sometimes it would go wrong and the annotator tool would suddenly skip multiple keypoints. Other times when I meant to skip a keypoint it would not skip them. These issues required me to remove incorrect annotations and re-annotate them as I would only notice after I had annotated other keypoints already. This slows down the annotating process, but it is probably easy to fix. E.g., having a clearly visible label somewhere on the screen showing which keypoint you are annotating and a skip-button to skip keypoints that cannot be annotated.
The Youth images test set contains 520 images with 1002 poses, which is not a lot compared to well-known datasets such as COCO or CrowdPose but this does not have to be an issue. The 1002 poses combined have 11,361 keypoints, which should be enough to make it a useful test set and is more than Ludl et al. (2018) had in their RARESIM test set of 43 images (see Section 2.9.2). However, their research was regarding pose estimation on a few very specific poses and thus there is also less need for a large test set.
The distribution of the visibility flags for the keypoints in the test set is very similar to that of the COCO validation set, which might be advantageous as the models trained on COCO have learned to predict the most visible keypoints well. There are also some images in the dataset where very few keypoints of a person are visible, e.g., only a knee. These images are difficult to annotate based on one image alone, but because the original image files contained four images (four camera views) in one file, I could use the other images to annotate keypoints more precisely. Images with very few keypoints visible are likely difficult for the pose estimators as they cannot use the other camera views.

5.4 Pose estimation results

The performance of all four models used for finetuning on the Youth images test has improved by training on a combination of synthetic and COCO training data, as can be seen in Table 14. The model with the largest improvement, HigherHRNet-W32, is also the only model for which I tried to find good hyperparameters, such as the ones described in the ablation experiments in Section 4.4.3. Before coming to the final baseline model for HigherHRNet-W32 I performed the data augmentation experiment, the freezing stage experiment and the experiment regarding the distribution of the synthetic-COCO images. The other ablation experiments were performed after I had collected the data in Table 14. Rather than trying to find good hyperparameters for the other three models, I opted to invest more time in ablation experiments. Due to time constraints, I could not do both, as training for 20 epochs took around 18 hours for the HigherHRNet-W32 model and similar amounts for the other three models.

The first thing one might notice is the large difference in performance between the bottom-up models and the top-down models. One explanation for this might be that the top-down models use the ground-truth bounding box during testing, as I mentioned in Section 4.4.1. When the top-down models are tested on the COCO test-dev or validation set, for example, they use a file with predicted bounding boxes. These bounding boxes are predicted by a person detector such as YOLO. These predictions are, of course, not perfect and probably less useful than the ground-truth bounding boxes. Using the ground-truth bounding boxes might inflate the performance as it is based on the annotation of the keypoints. However, these bounding boxes themselves are not perfect either, as I had to use a margin to ensure that there were no important body parts excluded from the bounding box.

I did try to use YOLO but as I only got my first results on July 3^{rd}, I instead focused on getting pose estimation results instead of person detection results. Nevertheless, some differences between the top-down and bottom-up models ought to be expected as top-down models perform better overall. Bottom-up methods do catch up on crowded scenes but I am not sure if the scenes in the Youth test set are crowded enough for that to happen or if the large difference is caused by something else. There is a lot of occlusion, but for the most occluded keypoints I was unable to annotate them so these cannot affect the performance. It might also be the case that the top-down models are 'forced' to annotate the exact right number of people in the image, while bottom-up models do not receive such information about the images.

Even though the Stacked Hourglass did not obtain the highest AP, it still performs well. Especially when you consider that the model is from 2016 (2016) and only has 9.4M parameters. Since 2016 there have been a lot of models that outperform Stacked Hourglass on datasets such as COCO. However, I am fairly certain that it is an updated version of the stacked hourglass as the MMpose website mentions that it has an AP of 0.746 on the

COCO validation set.
One oddity in Table 14 is the table are the low AP^M scores. This means that the models perform poorly on annotations with a small area, smaller than 9612. The images in the Youth test set are not taken from afar, so it is not necessarily the case that the models perform poorly on people in the distance or background as there are no people that far away. As the area is based on the smallest box containing all the keypoints, I suspect that the annotations in this category only have a subset of predominantly incomplete annotations.

	Small	Medium	Large	All sizes
0-3 keypoints	20	16	8	44
4-6 keypoints	2	27	58	87
7-9 keypoints	0	30	137	167
10-12 keypoints	0	25	220	245
13-15 keypoints	0	16	316	332
16-17 keypoints	0	6	126	132
Total	22	115	865	1002

Table 22. The number of keypoints per area category Small (< 1024), Medium (> 1023 and < 9216) and Large (> 9215).

When we look at the distribution of annotations based on their area category in Table 22, we can see that most annotations belong to the large category, about 86%. In the medium category, most annotations are actually somewhat complete, which is not what I expected. But because few annotations have a small area, the AP^M does not have a large effect on the AP. The AR of the models improved as well, meaning it can (confidently) detect more keypoints than before the finetuning.

5.4.1 Ablation experiments

When we look at the results of the ablation experiments we can get an idea of what aspects of the training procedure and data helps improve the models on the Youth images test set. The first ablation experiment is related to the data augmentation policy and the results can be found in Table 13. The APs of the three data augmentation variants, which all used the whole synthetic dataset, are very similar between 63.40 and 63.65. Meanwhile, the baseline model using the same HigherHRNet-W32, the standard data augmentation policy and a different data distribution, had an AP of 64.47 after 10 epochs.
Comparing it to the baseline does not make that much sense as the data distribution is different but it does give an idea of how much could be learnt in 10 epochs. Beforehand I expected that using the standard data augmentation method with the whole synthetic dataset might inhibit the model from improving as about one-third of the synthetic images

already has a camera roll, which has the same effect as a rotation augmentation. That is why I decided to use the R0 subset for all other ablation experiments as well as the final four models.

When we look at the results we can see that the data augmentation policy does not matter as much, probably because the rotation range is not too large and a bit less than 10% of the images have a roll value of 1. The reason for including the updated scaling is because the synthetic images have a resolution of 1200×634 and the model's input size is 512×512, meaning that the images get scaled down before being processed. The updated scaling range is meant to counteract this, but this scaling also affects the regular COCO training images and this might hurt the model's performance on those images. Regardless, the results show that when using the whole synthetic dataset with a 50-50 distribution of synthetic and COCO training data, the standard data augmentation policy is still the best for training the HigherHRNet-W32 when tested on the Youth images test set.

In the second ablation experiment, I looked at freezing a number of stages, either 2, 3 or 4, because 0 and 1 would not work due to the GPU not having enough RAM. I expected freezing 3 stages would perform the best as it allows for enough learning whereas 4 frozen stages can only affect the last few layers in the head and not the HRNet backbone of HigherHRNet. I thought that freezing 2 stages would allow the model to be affected too much and start overfitting eventually. However, when we look at Table 16, we can see that the variant with 2 frozen stages actually performs better than freezing 3 stages. Perhaps the overfitting would come later but there is no indication for me based on the results that this is the case. I did correctly predict that the model with 4 frozen stages would perform poorly, but I did not anticipate a lower AP on the test set than the pre-finetuning model. The difference with the pre-finetuning model is not that large, only 0.25 percentage points, but it is a reduction nonetheless. Based on the fact that the variant with 2 frozen stages only works with a batch size of 8, took longer to train and only had a 0.23 higher AP, I decided to stick with freezing 3 stages as I had done before with the data augmentation ablation experiment.

The third ablation experiment that I performed was related to the synthetic and COCO training data distribution. Up to this point, I used a 50-50 distribution, which means that approximately half of all training images are COCO and the other half synthetic. I had no idea what a good distribution would be and therefore I opted to go for one that focuses more on the new synthetic data. The number of batches per epoch was also smaller as I set it to two times the number of images in the synthetic dataset used, which is 81,042 for the whole dataset and 54,084 for the R0 subset. During this experiment, I tried a different distribution which I refer to as 0.3, as approximately 30% of the images used for training were synthetic with the R0 subset. As Table 17 shows, using the 0.3 distribution results in higher performance than the 0.5 (or 50-50) distribution. The observant reader would realise that, while they both trained for 20 epochs, the 0.3 distribution processed almost

50% more images, albeit that those are COCO training images on which it has already trained for more than 200 epochs. The third variant in the table shows the performance of the 0.3 distribution variant after only 10 epochs, meaning that it has processed fewer images than the 0.5 distribution variant. And as this third variant has an AP that is 0.72 percentage points higher, it is safe to say that using a 0.3 distribution is the better option.
The fourth ablation experiment was performed to demonstrate that the model learns as much from the first 20 scenes as an additional 20. In other words, the performance improvement from 0 scenes, and 0 epochs, to 20 scenes and 20 epochs is roughly the same as from 20 to 40 scenes, both with 20 epochs. Of course, logically speaking there has to be a diminishing return in performance improvement at some point, but I expected that this would not really kick in after the first 20 scenes because the scenes are so different from one another. If there was no diminishing return after X scenes, or more broadly, X images, then models could achieve perfect performance by just increasing the amount of data ten- or a hundredfold. As Table 18 shows, this is not completely true. Going from 0 to 20 scenes improves the model's AP performance from 62.91 to 63.85, which is a 0.94 increase. The second improvement, from 20 to 40 scenes results in an AP of 64.72 or an improvement of 0.87. This is 0.07 less AP improvement than with the first 40 scenes.
When we look at the AP^M, which is difficult for all the models, the improvement with the first 20 scenes is 1.19 percentage points, while the second set of 20 scenes adds 1.49 percentage points. A possible explanation for this could be that the images of the scenes in the second set are better suited for improving the model's AP^M performance on the test set. But I cannot be sure as I have not trained the model using only the second set of 20 scenes. If training on these scenes alone results in the AP^M improving the most with these as the first scenes and less with the additional 20 then it would be a likely explanation. I suspect that adding more scenes will still be able to increase the model's performance, but at some point, the return for adding 20 scenes will be negligible, as the synthetic data is too different from real data. Meanwhile, the time to create 20 new scenes will not decrease as much.
The fifth ablation experiment is regarding the vertical degrees of the camera. As discussed in Section 4.2, I use three possible values (0, 12 and 24) for the vertical degrees while the test set does not have this specific variation. However, Figures 40 and 41 show that there is some difference between the four cameras in terms of vertical angle. I suspected that using only 0 or 12 degrees would still result in a performance improvement but because only a third of the images would be used, I expected a significantly lower performance improvement. However, Table 19 shows very small differences between the baseline, V0 and V12. In the case of V0, the difference is only 0.07 percentage points. Even though these two variants used about two-thirds less synthetic training data and more COCO training data, they still managed to improve the performance by at least 1.7 percentage points. When we look at the AP^{75} column in the table we can see that the baseline has a

higher score, at least 1.55 percentage points. I am not sure what causes this, but as the AP is a combination of AP^{50} to AP^{95} with steps of 5, thus if the AP^{75} is lower, then a different AP^{X} must be higher for these two variants than for the baseline.

The sixth ablation experiment is related to the Gaussian blur. As there is often motion blur in images, it might be advantageous to add some blur to images to make them look more like real images. Table 20 actually indicates the contrary, as the performance of the model trained on blurred synthetic data has an AP that is 0.45 percentage points lower. The Gaussian blur variant does still improve compared to the pre-finetuning model. Perhaps using the Gaussian blur as a data augmentation would be better than blurring all images. One explanation that I can give for the lower performance of the Gaussian Blur variant is that, before processing the image, the HigherHRNet models have to scale down from 1200×634 to 512×512 (keeping the aspect ratio the same), and thus the image loses (edge) details which is what the Gaussian blur does as well.

The last ablation experiment is regarding the different test set variants for which the results can be found in Table 21. It makes sense that the models achieve higher scores on the split variant as fewer details are lost when the images are processed. What I did not expect was that the performance improvements from the models would be smaller when the finetuned and pre-finetuning variants are compared to the non-split variant. The finetuned HigherHRNet-W32 only improves the performance with 0.99 and HigherHRNet-W48 only with 0.17 on the non-split test set. Most likely, the results may be a bit skewed in the sense that the performance of the HigherHRNet models on the test set does not increase steadily but goes up and down a little bit with more epochs of training. This means that, while after 20 epochs it performs well on the non-split variant, after 21 or 22 epochs it might perform slightly worse on the non-split and better on the split variant. The most noticeable observation made based on the table is that the AP^{M} improves so much, just by using the non-split variant. This is probably because, with the split variant, the model loses a lot of details, which is reduced by using the non-split variant and the medium-sized annotations require more details.

5.4.2 Qualitative results

The first thing I noticed, which also surprised me, about the qualitative results is that the HigherHRNet-W32 predicted a few keypoints in the images. HRNet-W48, on the other hand, has very complete predictions. The predictions of HigherHRNet-W32 tend to be more accurate while HRNet-W48 has a few severely inaccurate, and even unrealistic, predictions such as: predicting facial keypoints of the child on the face of the adult (Figure 40h), predicting the child's knees on the adult's knees (Figure 41b), predicting the adult's left elbow on her right knee (Figure 41d), incorrectly predicting the adult's hips and left knee and missing the clearly visible right ankle and knee (Figure 41f) and the double poses on the adult while failing to predict keypoints for the child (Figure 41j). Meanwhile,

the HigherHRNet-W32 did not detect the adult in Figure 41g, probably due to the box's lid and failed to see the child's visible right hip in the same figure.

6. Conclusion

In this thesis, I have researched if it is possible to synthesise data of adult-child interaction for pose estimation which can improve a pose estimator's performance without the use of motion capture data. There have been studies in the past that looked at synthesising data for pose estimation but they mostly relied on the use of motion capture data. Child-specific datasets are usually not publicly available while pose estimators struggle more with estimating the poses of children as well as the occlusions caused by interactions. Public datasets for adult-child interaction do not exist. Thus, synthesising data might be a solution to tackle this problem. Using motion capture data for this would be difficult as motion capture data is generally collected of a single someone performing actions, but not of people interacting. Moreover, using motion capture data for synthesising adult-child interaction data requires the motion capture data to be of adult-child interactions as the collected data is bound to a specific body. Using the data of an adult for a child does not work as children's bodies are significantly different from adults' bodies.

I proposed to, instead, create static scenes of adult-child interaction using 3D models made with MakeHuman. Two of these models were positioned in a Unity world where the models were adjusted to take on specific poses which results in the two models interacting. I have created 40 such scenes which can then be used to synthesise data. I ensured to add aesthetic variation in terms of models (skin colour, age, height and sex), the models' clothing, lighting and camera positions. This resulted in a dataset of 40,571 2D RGB images which can be used to train a pose estimator. This is possible thanks to the automatic and precise annotating done by Unity which accurately collects the area, bounding box, and 2D and 3D locations of the keypoints which are necessary for the pose estimation.

To determine whether the data helps improve the performance of a pose estimator requires a test set. As there are no publicly available pose estimation datasets of adult-child interaction, I annotated a number of images to create the Youth images test set. This test set consists of 520 images of adult-child interaction which are relatively difficult to estimate the pose on due to occlusions, people blending in with the background and keypoints being outside the image bounds.

Four pre-trained state-of-the-art pose estimator models (HigherHRNet-W32, HigherHRNet-W48, HRNet-W48 and Stacked Hourglass) were finetuned on a combination of COCO training data and the R0 subset of the synthetic data. These finetuned models improved the performance on the Youth images test set when compared to the non-finetuned models. This finetuning resulted in AP improvements of 1.81, 0.72, 1.14 and 1.55 respectively for the four models as well as AR improvements of 2.52, 2.17, 1.35 and 1.25.

The main research question was:

Is it possible to synthesise adult-child interaction pose data to improve the performance of a human pose estimation model without using motion capture data?

To answer this question I also specified three sub-questions which I tried to answer as well. The first sub-question was:

What type of poses ought to be present when synthesising adult-child interaction pose to improve the performance?

Pose estimators struggle with interaction data, and specifically adult-child interaction data, due to the many occlusions such as self-occlusion and inter-person occlusion as can be seen in Figure 27 as well as in Figures 40 and 41. Therefore, the poses that ought to be present in the synthesised pose data exhibit both self-occlusion and inter-person occlusion as well as poses that one could realistically expect with the adult-child interaction. These are the poses that the pose estimators can learn the most from as these are the ones that the pose estimators struggle the most with. Synthesising interaction scenes with rare poses like gymnastics poses would not be so helpful and poses without any interaction do not make sense either.

The second sub-question asked:

How can this type of adult-child interaction pose data be synthesised without using motion capture data?

As there is no public motion capture dataset of adult-child interaction, synthesising adult-child interaction data would be rather difficult using the motion capture data which is currently available. Motion capture data of an adult could not easily be used with a child model and synthesising interaction data requires interaction motion capture data, which is rare. Besides the availability issues, motion capture datasets usually have low actor diversity, which means that there would be less model diversity than in my approach. Additionally, the pose diversity is not huge either, although this is also an issue with my approach. Mass-producing 3D human models and then adjusting their poses in Unity allowed me to create 40 scenes of adult-child interactions. I used these scenes to synthesise more than 40,000 images and created precise annotations automatically with information such as 2D and 3D keypoint locations, keypoint visibility flags and accurate bounding boxes and areas for the people present in the images.

The third sub-question is as follows:

What data aspects, besides the poses, are important to improve a pose estimator's performance on adult-child interactions?

The poses are very important for synthesising adult-child pose data, as it influences how much occlusion there is and if there is enough interaction. Nonetheless, other aspects of the synthesised data are important as well. What's more, the diversity in poses is lower than in real data and thus the other data aspects need to be diverse enough. Every dataset has its own bias and removing this bias is extremely difficult. By having enough diversity in

aspects such as materials, lighting, backgrounds, clothing and models, you can ensure that those will not be the main bias in the synthetic dataset. Only using a handful of models can quickly lead to a bias, e.g., using only Caucasian models has a bias which could negatively impact the performance on non-caucasian people in a test set. If a model trained on such a dataset would be used for applications that impact people's lives it could be disastrous. Biases due to low diversity in lighting or backgrounds will be less impactful, but could still hinder a model's learning. A lack of diversity in clothing and materials can also hinder a model's learning as it would not be a good representation of real-world data nor would it be varied enough such that the model does not overfit to the type of clothing and material used. As the synthesised data will undoubtedly have low diversity in terms of poses, diversity in camera position and rotation is paramount. Although this cannot resolve the lack of pose diversity, it can give the appearance of more pose diversity in the images. To answer the sub-question, the more diversity in the data, the better, with the diversity in camera position and rotation, clothing and materials being the most important.
Coming back to the main research question, by creating scenes with 3D human models and having Unity generate more than 40,000 images of 40 different scenes I have managed to synthesise adult-child interaction pose data without using motion capture data. By finetuning four pose estimator models on the synthesised data I have improved their performance on the Youth images test set which consists of adult-child interactions. The improved performance on the test set shows that it is possible to synthesise adult-child interaction pose data to improve the performance of a human pose estimation model without using motion capture data.

6.1 Limitations and future works

During my research, I was able to perform multiple ablation experiments although there are always additional experiments that can be performed to research if the models' performance could be improved more. One possible ablation experiment would be using keypoint-centric occlusion as discussed in Section 2.8.2 as I did not have enough time to experiment with this. The UDP++ model used this data augmentation approach and saw a 0.7 percentage points AP improvement. As occlusion augmentation methods are relatively new for human pose estimation, it would be worth investing if it could also help a model's performance on the Youth test set as there is a significant amount of occlusion. One issue of the Youth test images is that there are relatively many images where parts of a person are completely outside the camera frustum. Perhaps combining multiple images from one image file (different camera views of the same people) would improve the general accuracy of the pose estimation and specifically that of keypoints outside the camera frustum. As there are relatively few images in the test set, it might also be advantageous to extend it and create a validation set as well, as this would make the hyper-parameter search easier.

One more possible continuation of this research that I was not able to investigate is 3D pose estimation. As the 3D ground truth location is present in the annotation files it is relatively easy to use the data for this purpose.
The most significant limitation of the synthesised dataset is the relatively limited pose diversity. As there are 40 scenes, each with two models, there are only 80 unique poses. A dataset of real images will have far more diverse poses and training on a dataset with more diversity in terms of poses could improve a model's performance and prevent overfitting. However, a low diversity in poses is a common issue for synthetic datasets and is difficult to fix. The different camera angles do help somewhat as then the relative 2D distances and angles between two joints change. Nevertheless, the small number of unique poses in this dataset cannot come close to the number of poses in the Youth dataset. One interesting direction for further research would be the implementation of (small) random movements of the models. This could potentially be achieved by computing for each model what its contact points with the ground and others are. Then you could try to find possible rotations/translations of limbs such that the contact points remain. For limbs where there is no contact, it is even easier to implement these random movements and in the case of inter-person contact, it might also be possible to move the bodyparts of both models simultaneously. This approach does, however, require the ability to determine the positions of individual vertices or at least of smaller groups of vertices. I was not able to access these vertices during run-time to compute the locations of the facial keypoints and instead used invisible clothing meshes for this. Theoretically, it would be possible to create a whole load of small invisible clothing meshes corresponding to a few vertices and faces, but the number of meshes this would require is huge. I did not spend hours trying to access the vertices during runtime so there is likely a workaround for this issue.
As for the components used to create the dataset, there are some possible improvements as well. The human models made with MakeHuman could have additional diversity by including more skin materials and more clothing options. Having the clothing materials blend more with the background can potentially teach the pose estimator to better distinguish between person and background. This can then increase the model's performance on adult-child interaction data.
Although the models have 10 or 12 variants, the clothing options are relatively limited and this could harm the pose estimator's performance. The models themselves could also be more realistic as they do not have as many details as, for example, MetaHuman creator. But unless MetaHuman creator allows the creation of child models, it still will not be an alternative. The models from MakeHuman can, however, use better variable ranges. I had to limit them due to issues such as the models seeming to have a spinal disease or other unrealistic or rare abnormalities. With some testing, the ranges for some of these variables could be made larger and this would result in more diversity in the models' bodies.

A. Appendix

```
Distance mult: 0, Height: 0, Roll: 0, Tilt: 0, Yaw: 0
Camera_position: -2.380792, 20.10545, 253.8394
Camera_rotation: 0, 0, 0
Light: 64.52138, 229, 180
[0] Nose: 672.2695, 230.365; -18.97424, 39.76765, 127.7726; 5
[0] Left_eye: 679.554, 214.2132; -20.17429, 42.87166, 131.0336; 5
[0] Right_eye: 674.2396, 214.7766; -19.7513, 43.7896, 125.3709; 2
[0] Left_ear: 720.6173, 213.5383; -28.45725, 42.25681, 135.1372; 5
[0] Right_ear: 702.3464, 215.0417; -27.30899, 44.69552, 120.1067; 2
[0] Left_shoulder: 800.3864, 293.349; -50.55305, 25.55069, 121.8471; 5
[0] Right_shoulder: 673.0627, 294.549; -20.81694, 25.51825, 115.2933; 5
[0] Left_elbow: 785.5985, 393.9551; -43.93148, 2.653327, 130.9192; 5
[0] Right_elbow: 664.2808, 392.7881; -17.5002, 2.044215, 124.6957; 5
[0] Left_wrist: 726.9516, 374.6415; -28.15161, 8.201394, 142.3818; 5
[0] Right_wrist: 621.1188, 332.3646; -6.982358, 16.53977, 134.2055; 5
[0] Left_hip: 715.402, 354.0617; -39.89466, 7.732687, 75.3559; 2
[0] Right_hip: 700.5933, 297.0677; -35.12072, 26.26731, 75.13801; 2
[0] Left_knee: 604.8326, 354.2217; -4.067411, 6.765742, 62.21282; 5
[0] Right_knee: 604.4928, 330.1049; -3.998835, 15.02577, 56.10321; 2
[0] Left_ankle: 598.323, 352.1919; -1.681535, 5.013881, 24.88827; 2
[0] Right_ankle: 581.8498, 336.2919; 5.348451, 11.46414, 20.02241; 5
[1] Nose: 546.5117, 266.6232; 9.730734, 31.28599, 129.514; 5
[1] Left_eye: 547.2337, 258.0818; 9.792322, 33.46709, 127.1719; 4
[1] Right_eye: 537.3951, 256.3308; 11.64028, 33.46904, 130.8711; 4
[1] Left_ear: 534.2429, 254.0839; 13.48918, 35.04844, 121.3279; 2
[1] Right_ear: 506.1996, 248.9655; 18.53087, 35.04998, 131.4331; 5
[1] Left_shoulder: 525.1478, 286.0292; 16.08242, 27.49812, 118.4068; 4
[1] Right_shoulder: 484.6935, 287.0161; 23.62602, 26.64263, 130.0015; 5
[1] Left_elbow: 543.5295, 321.4499; 11.81336, 18.73558, 115.8302; 5
[1] Right_elbow: 478.3965, 329.1349; 24.25994, 17.22786, 133.552; 5
[1] Left_wrist: 548.8946, 325.3722; 9.475202, 17.93117, 126.4623; 5
[1] Right_wrist: 489.2464, 377.8474; 21.67874, 6.670029, 134.5645; 5
[1] Left_hip: 506.9172, 365.5545; 20.46196, 7.944632, 119.0987; 2
[1] Right_hip: 483.24, 368.2574; 24.79151, 7.944138, 126.0625; 4
[1] Left_knee: 546.9857, 400.6105; 10.07715, 0.22264, 124.8147; 5
[1] Right_knee: 518.3307, 326.2573; 15.90011, 17.80944, 130.9374; 5
[1] Left_ankle: 509.0659, 388.9229; 20.68327, 1.609621, 114.5785; 2
[1] Right_ankle: 504.1086, 391.0228; 19.04351, 3.343654, 131.167; 4
[0]: Female_34_159_Asi_SwTs_mass0027; (533, 156), (824, 409); RE; Area: 38422;
Shirt: Fabric-Design-Textile-Pattern-Geometric-Texture-2161331 (255, 255, 255),
Pants: SHERPA-white (125.5095, 26.43387, 10.40266),
Shoes: tennisshoes (79.80247, 79.80247, 79.80247), Eyes: brownlight_eyes
[1]: Male_2_70_Afr_SwTs_mass0005; (467, 211), (558, 410); ; Area: 12466;
Shirt: fabric-photo-cropped (74.74247, 34.82276, 149.1816),
Pants: SHERPA-white (24.77749, 116.8496, 84.52442),
Shoes: shoes06_diffuse (255, 255, 255), Eyes: green_eyes
```

Figure 42. An example of an annotation file, scene1_H0_V0_D0_He0_R0_T0_Y0.txt. I had to split the last two lines in order to fit the text on one page, so the lines starting with 'Shirt', 'Pants', 'Shoes' and 'Eyes' were originally all on the line containing the name of the models.

Bibliography

Andriluka, M., Pishchulin, L., Gehler, P., and Schiele, B. (2014). 2d human pose estimation: New benchmark and state of the art analysis. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*, pages 3686–3693.

Bourdev, L. and Malik, J. (2009). Poselets: Body part detectors trained using 3d human pose annotations. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1365–1372. IEEE.

Brooks, J. (2019). COCO Annotator. `https://github.com/jsbroks/coco-annotator/`.

Burdi, A. R., Huelke, D. F., Snyder, R. G., and Lowrey, G. H. (1969). Infants and children in the adult world of automobile safety design: pediatric and anatomical considerations for design of child restraints. *Journal of Biomechanics*, 2(3):267–280.

Cao, Z., Martinez, G., Simon, T., Wei, S.-E., and Sheikh, Y. (2019). Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP:1–1.

Cao, Z., Simon, T., Wei, S.-E., and Sheikh, Y. (2017). Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7291–7299.

Carreira, J., Agrawal, P., Fragkiadaki, K., and Malik, J. (2016). Human pose estimation with iterative error feedback. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4733–4742.

Catalin Ionescu, Fuxin Li, C. S. (2011). Latent structured models for human pose estimation. In *International Conference on Computer Vision*.

Chen, P., Liu, S., Zhao, H., and Jia, J. (2020). Gridmask data augmentation. *arXiv preprint arXiv:2001.04086*.

Chen, W., Wang, H., Li, Y., Su, H., Wang, Z., Tu, C., Lischinski, D., Cohen-Or, D., and Chen, B. (2016). Synthesizing training images for boosting human 3d pose estimation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 479–488. IEEE.

Chen, Y., Wang, Z., Peng, Y., Zhang, Z., Yu, G., and Sun, J. (2018). Cascaded pyramid network for multi-person pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7103–7112.

Cheng, B., Xiao, B., Wang, J., Shi, H., Huang, T. S., and Zhang, L. (2020). Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5386–5395.

Cheng, Y., Wang, B., Yang, B., and Tan, R. T. (2021). Monocular 3d multi-person pose estimation by integrating top-down and bottom-up networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7649–7659.

Covre, N., Nunnari, F., Fornaser, A., and Cecco, M. D. (2019). Generation of action recognition training data through rotoscoping and augmentation of synthetic animations. In *International Conference on Augmented Reality, Virtual Reality and Computer Graphics*, pages 23–42. Springer.

DeVries, T. and Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*.

Ding, Y., Deng, W., Zheng, Y., Liu, P., Wang, M., Cheng, X., Bao, J., Chen, D., and Zeng, M. (2022). Iˆ 2r-net: Intra-and inter-human relation network for multi-person pose estimation. *arXiv preprint arXiv:2206.10892*.

Eichner, M., Marin-Jimenez, M., Zisserman, A., and Ferrari, V. (2012). 2d articulated human pose estimation and retrieval in (almost) unconstrained still images. *International journal of computer vision*, 99(2):190–214.

Ferrari, V., Marín-Jiménez, M. J., and Zisserman, A. (2008). Progressive search space reduction for human pose estimation. *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

Fieraru, M., Khoreva, A., Pishchulin, L., and Schiele, B. (2018). Learning to refine human pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 205–214.

Fischler, M. A. and Elschlager, R. A. (1973). The representation and matching of pictorial structures. *IEEE Transactions on Computers*, C-22:67–92.

Gal, Y. and Ghahramani, Z. (2015). Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*.

Garbin, C., Zhu, X., and Marques, O. (2020). Dropout vs. batch normalization: an empirical study of their impact to deep learning. *Multimedia Tools and Applications*, 79(19):12777–12815.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

Golda, T. (2019). Image-based anomaly detection within crowds. In *Proceedings of the 2018 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory. Ed.: J. Beyerer, M. Taphanel*, volume 40, pages 11–24.

Groos, D., Adde, L., Støen, R., Ramampiaro, H., and Ihlen, E. (2021). Towards human-level performance on automatic pose estimation of infant spontaneous movements. *Computerized Medical Imaging and Graphics*, 95:102012.

Hashemi, J., Spina, T. V., Tepper, M., Esler, A., Morellas, V., Papanikolopoulos, N., and Sapiro, G. (2012). A computer vision approach for the assessment of autism-related behavioral markers. In *2012 IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, pages 1–7. IEEE.

Hassan, M., Choutas, V., Tzionas, D., and Black, M. J. (2019). Resolving 3d human pose ambiguities with 3d scene constraints. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2282–2292.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hesse, N., Bodensteiner, C., Arens, M., Hofmann, U. G., Weinberger, R., and Sebastian Schroeder, A. (2018a). Computer vision for medical infant motion analysis: State of the art and rgb-d data set. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0.

Hesse, N., Pujades, S., Romero, J., Black, M. J., Bodensteiner, C., Arens, M., Hofmann, U. G., Tacke, U., Hadders-Algra, M., Weinberger, R., Müller-Felber, W., and Schroeder, A. S. (2018b). Learning an infant body model from RGB-D data for accurate full body motion analysis. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer.

Hesse, N., Schröder, A. S., Müller-Felber, W., Bodensteiner, C., Arens, M., and Hofmann, U. G. (2017). Body pose estimation in depth images for infant motion analysis. In

2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pages 1909–1912. IEEE.

Holt, B., Ong, E.-J., Cooper, H., and Bowden, R. (2011). Putting the pieces together: Connected poselets for human pose estimation. In *2011 IEEE international conference on computer vision workshops (ICCV workshops)*, pages 1196–1201. IEEE.

Huang, J., Shan, Z., Cai, Y., Guo, F., Ye, Y., Chen, X., Zhu, Z., Huang, G., Lu, J., and Du, D. (2020a). Joint coco and lvis workshop at eccv 2020: Coco keypoint challenge track technical report: Udp+. In *ECCV Workshop*.

Huang, J., Zhu, Z., Guo, F., and Huang, G. (2020b). The devil is in the details: Delving into unbiased data processing for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Huang, J., Zhu, Z., Huang, G., and Du, D. (2020c). Aid: Pushing the performance boundary of human pose estimation with information dropping augmentation. *arXiv preprint arXiv:2008.07139*.

Huang, X., Fu, N., Liu, S., and Ostadabbas, S. (2021). Invariant representation learning for infant pose estimation with small data. In *2021 16th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2021)*, pages 1–8. IEEE.

Hussain, F. (2018). Learn opengl. `https://www.oreilly.com/library/view/learn-opengl/9781789340365/843ddbf2-3a57-4f00-928f-7de5d42415ad.xhtml`.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR.

Ionescu, C., Papava, D., Olaru, V., and Sminchisescu, C. (2013). Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1325–1339.

Johnson, S. and Everingham, M. (2010). Clustered pose and nonlinear appearance models for human pose estimation. In *bmvc*, volume 2, page 5.

Johnson, S. and Everingham, M. (2011). Learning effective human pose estimation from inaccurate annotation. In *CVPR 2011*, pages 1465–1472. IEEE.

Josyula, R. and Ostadabbas, S. (2021). A review on human pose estimation. *arXiv preprint arXiv:2110.06877*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krizhevsky, A., Sutskever, I., and Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25.

Li, J., Wang, C., Zhu, H., Mao, Y., Fang, H.-S., and Lu, C. (2019). Crowdpose: Efficient crowded scenes pose estimation and a new benchmark. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10863–10872.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Lin, T.-Y., Patterson, G., Ronchi, M. R., Cui, Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, L., and Dollár, P. (2022). Coco - common objects in context.

Liu, H., Liu, F., Fan, X., and Huang, D. (2021). Polarized self-attention: Towards high-quality pixel-wise regression. *arXiv preprint arXiv:2107.00782*.

Liu, J., Rahmani, H., Akhtar, N., and Mian, A. (2019). Learning human pose models from synthesized data for robust rgb-d action recognition. *International Journal of Computer Vision*, 127(10):1545–1564.

Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., and Black, M. J. (2015). Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16.

Ludl, D., Gulde, T., Thalji, S., and Curio, C. (2018). Using simulation to improve human pose estimation for corner cases. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3575–3582. IEEE.

Luo, Z., Wang, Z., Huang, Y., Wang, L., Tan, T., and Zhou, E. (2021). Rethinking the heatmap regression for bottom-up human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13264–13273.

Matthews, O., Ryu, K., and Srivastava, T. (2020). Creating a large-scale synthetic dataset for human activity recognition. *arXiv preprint arXiv:2007.11118*.

Medeiros, A. J., Stearns, L., Findlater, L., Chen, C., and Froehlich, J. E. (2017). Recognizing clothing colors and visual textures using a finger-mounted camera: An initial investigation. In *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility - ASSETS '17*.

MMPoseContributors (2020). Openmmlab pose estimation toolbox and benchmark. https://github.com/open-mmlab/mmpose.

Moon, G., Chang, J. Y., and Lee, K. M. (2019). Posefix: Model-agnostic general human pose refinement network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7773–7781.

Newell, A., Huang, Z., and Deng, J. (2017). Associative embedding: End-to-end learning for joint detection and grouping. *Advances in neural information processing systems*, 30.

Newell, A., Yang, K., and Deng, J. (2016). Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer.

Papandreou, G., Zhu, T., Chen, L.-C., Gidaris, S., Tompson, J., and Murphy, K. (2018). Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In *Proceedings of the European conference on computer vision (ECCV)*, pages 269–286.

Papandreou, G., Zhu, T., Kanazawa, N., Toshev, A., Tompson, J., Bregler, C., and Murphy, K. (2017). Towards accurate multi-person pose estimation in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4903–4911.

Park, S., Lee, S.-b., and Park, J. (2020). Data augmentation method for improving the accuracy of human pose estimation with cropped images. *Pattern Recognition Letters*, 136.

Perez, L. and Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*.

Prechtl, H. F. (1990). Qualitative changes of spontaneous movements in fetus and preterm infant are a marker of neurological dysfunction. *Early human development*.

Pytel, R., Kayhan, O. S., and van Gemert, J. C. (2021). Tilting at windmills: Data augmentation for deep pose estimation does not help with occlusions. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 10568–10575. IEEE.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

Ruggero Ronchi, M. and Perona, P. (2017). Benchmarking and error diagnosis in multi-instance pose estimation. In *Proceedings of the IEEE international conference on computer vision*, pages 369–378.

Salah, A. A. (2021). Designing computational tools for behavioral and clinical science. In *Companion of the 2021 ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, pages 1–4.

Sapp, B. and Taskar, B. (2013). Modec: Multimodal decomposable models for human pose estimation. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3681.

Sarafianos, N., Boteanu, B., Ionescu, B., and Kakadiaris, I. A. (2016). 3d human pose estimation: A review of the literature and analysis of covariates. *Computer Vision and Image Understanding*, 152:1–20.

Sciortino, G., Farinella, G. M., Battiato, S., Leo, M., and Distante, C. (2017). On the estimation of children's poses. In *International conference on image analysis and processing*, pages 410–421. Springer.

Shorten, C. and Khoshgoftaar, T. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6.

Shrivastava, A., Gupta, A., and Girshick, R. (2016). Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 761–769.

Singh, K. K., Yu, H., Sarmasi, A., Pradeep, G., and Lee, Y. J. (2018). Hide-and-seek: A data augmentation technique for weakly-supervised localization and beyond. *arXiv preprint arXiv:1811.02545*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Sun, K., Xiao, B., Liu, D., and Wang, J. (2019). Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5693–5703.

Tompson, J., Goroshin, R., Jain, A., LeCun, Y., and Bregler, C. (2015). Efficient object localization using convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 648–656.

Tompson, J. J., Jain, A., LeCun, Y., and Bregler, C. (2014). Joint training of a convolutional network and a graphical model for human pose estimation. *Advances in neural information processing systems*, 27.

Toshev, A. and Szegedy, C. (2014). Deeppose: Human pose estimation via deep neural networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition*.

Toyoda, K., Kono, M., and Rekimoto, J. (2019). Post-data augmentation to improve deep pose estimation of extreme and wild motions. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 1570–1574. IEEE.

Unreal Engine (2021). MetaHuman Creator Body User Guide. `https://docs.metahuman.unrealengine.com/en-US/UserGuide/Body/`.

Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M. J., Laptev, I., and Schmid, C. (2017). Learning from synthetic humans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 109–117.

Wang, C., Zhang, F., Zhu, X., and Ge, S. S. (2022). Low-resolution human pose estimation. *Pattern Recognition*, page 108579.

Wang, Y., Liang, W., Shen, J., Jia, Y., and Yu, L.-F. (2019). A deep coarse-to-fine network for head pose estimation from synthetic data. *Pattern Recognition*, 94.

Wang, Z., Li, W., Yin, B., Peng, Q., Xiao, T., Du, Y., Li, Z., Zhang, X., Yu, G., and Sun, J. (2018). Ms coco keypoints challenge 2018. *Joint Recognition Challenge Workshop at ECCV 2018*.

Wei, S.-E., Ramakrishna, V., Kanade, T., and Sheikh, Y. (2016). Convolutional pose machines. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 4724–4732.

Xiao, B., Wu, H., and Wei, Y. (2018). Simple baselines for human pose estimation and tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 466–481.

Yang, Y. and Ramanan, D. (2011). Articulated pose estimation with flexible mixtures-of-parts. In *CVPR 2011*, pages 1385–1392.

Yang, Y. and Ramanan, D. (2013). Articulated human detection with flexible mixtures of parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2878–2890.

Yurtsever, M. E. and Eken, S. (2022). Babypose: Real-time decoding of baby's non-verbal communication using 2d video-based pose estimation. *IEEE Sensors Journal*.

Zhang, F., Zhu, X., Dai, H., Ye, M., and Zhu, C. (2020). Distribution-aware coordinate representation for human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7093–7102.

Zhong, Z., Zheng, L., Kang, G., Li, S., and Yang, Y. (2020). Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008.

Zhou, J., Jiang, Z., Yoo, J.-H., and Hwang, J.-N. (2021). Hierarchical pose classification for infant action analysis and mental development assessment. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1340–1344. IEEE.

Zhu, A., Snoussi, H., and Cherouat, A. (2015). Articulated pose estimation via multiple mixture parts model. In *2015 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–5. IEEE.