1 The Dataset

To perform the pose estimation, we need some data on which to train, validate and test our model. Throughout this section the used data and the relevant preprocessing is described. The section consists of two parts: Section 1.1, where a brief overview of the dataset is given, and Section 1.2, where the preprocessing of the data is described.

1.1 The COCO Dataset

Notice how the image contains multiple people, each with their own keypoints and amount of joints labeled

Figure 1: Example of an image from the COCO dataset with the keypoints drawn on [2]

The data needed for our model has to fit to our problem and has to be annotated, as our model will perform supervised learning. There are multiple datasets that fits these requirements. One of these datasets is the Common Objects in Context (COCO) dataset [2], which we will be using. The dataset contains annotations for different purposes, however, for our pose-estimation-task, only the keypoint annotations of human bodies are needed. An example of such a picture with the keypoints labeled can be seen in Figure 1.

The annotation of each person consists of an array with a length of 51, which annotates 17 keypoints of a person. Thus, each joint corresponds to three sequential elements in the array, where the first and second indices corresponds to the x and y-location of the joint in the image, and the third index is a flag, v, indicating the visibility of the joint in the image. v has three outcomes: if $v = 0$, the joint is not labeled, if $v = 1$, the joint is labeled but not visible, and if $v = 2$, the joint is visible and labeled.

The creators of the dataset has already split the data into three parts: a part used for training the model, a part used for validating the model and a part used for testing the model. However, the part used for testing the model is unlabel, hence, it is unusable for our purpose, as our model will be doing supervised learning. As both the training dataset and the validation dataset will be used for training and tuning the model, we will need to create our own hold-out dataset for testing to provide an unbiased evaluation of the final model.

The training and validation sets contains a total of about 123.000 various images. As we only need the images that contain humans, we will be discarding the images without any humans, leaving us with a total of about 66.808 images of humans doing various tasks, with a total of 149.813 humans annotated with keypoints. Each image can contain multiple people, which we need to handle before training our model, as we will be focusing on single-human pose estimation. Besides this, each image also has different resolution and aspect ratio, which we also need to handle, as our model requires the images to have a fixed resolution. Lastly, we

should also do some handling of the labels before training the model, as there could have been some inaccuracies, when the joints were labeled. This especially applies when $v = 1$, that is, when the joint is labeled but not visible, as there are more inaccuracies or uncertainty when labeling a non-visible joint than when labeling a visible joint.

1.2 Data Preprocessing

1.2.1 Creating the test dataset

To create the testing dataset we take the training set, since it is the larger of the training and validation set, and sample 5.064 images randomly without replacement, to create a test set. This ensures that the test-set and validation-set are of the same size. This new test set will not be used when training the model nor used when tuning the parameters. Instead, it will only be used to evaluate the very final model.

1.2.2 Preprocessing the images

	Amount of images	Percentage
Training set	124.040	$92,45\%$
Validation set	5.064	$3,77\%$
Testing set	5.064	$3,77\%$
Total	**134.168**$\%$	**100**$\%$

Table 1: Data distribution

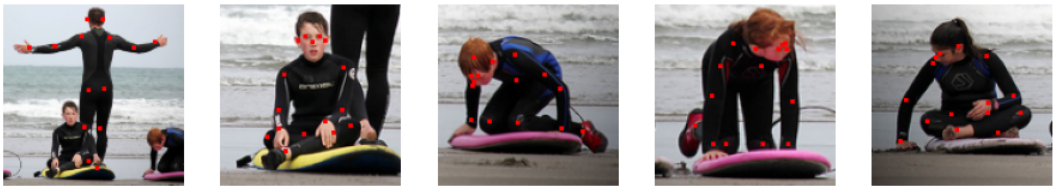


Figure 2: The results of processing the image from Figure 1 with the corresponding labels [2]

We start the preprocessing of the images by creating multiple bounding boxes, where each bounding box surrounds a single person, which is done by making use of the boundig box annotations provided by COCO. Then each bounding box is transformed into a square by making the shorter sides have the same length as the longer sides - this is done to ensure that the aspect ratio of the image is kept, when it is later resized.

It is possible for each bounding box to contain multiple people. This is a problem, as it will confuse our model, since it will not know which person to annotate. An example of this can be seen in the first image of Figure 2. To fix this we center the bounding box around the person it should annotate, making the model annotate the person in the center of the input image, which is done by centering the bounding box with respect to the outermost keypoints of the person.

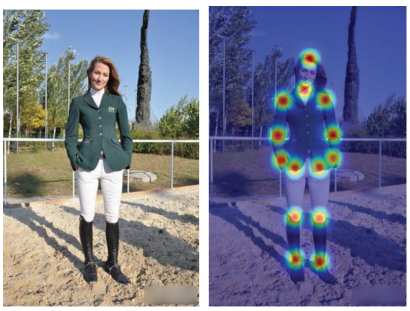
Since each keypoint does not necessarily lie on the edge of the person, the bounding boxes could result in not all of the pixels of the corresponding person being in the bounding box. For this reason, each bounding box is expanded with 10% in the height and width. If, however, the image cannot contain the expanded bounding box, the bounding box is then expanded as much as possible, while still being a square. If it is the case, that one of the corners of the

bounding box lies outside of the image, then the bounding box is moved either up or down, making the corner of the bounding box be inside the image and keeping the annotated person centered along the x-axis.

When all of the above is done, the image is finally cropped to each bounding box, resulting in multiple squared images, each containing an unique person at the center. Each of these squared images are then resized to a 256×256 image. We then center the rgb-values of each image by subtracting the mean rgb of all of the images from the training set from each image. Then, each images is saved as an `.png`.

By doing the data preprocessing as described above, we get the distribution of images displayed in Table 1. In Figure 2, the results of processing the image from Figure 1 are shown with the corresponding labels. Lastly, the data is shuffled to help the developed model generalize the data better.

1.2.3 Handling the labels



Left: The original image. Right: The heatmaps of all the keypoints, fused together to a single image.

Figure 3: An example of the heatmaps of a single image fused together and put over the original image [1]

For each image our model outputs 17 heatmaps, one for each possible joint in the image. An example of such heatmaps fused together can be seen in Figure 3. The ground truth heatmap of a single joint is created firstly by initializing an all-zero 2D array with size 64×64. Next, at coordinate $\left(x \cdot \frac{64}{256}, y \cdot \frac{64}{256}\right)$, where (x, y) is the annotated position of the joint in the original image, a 1 is places, representing the position of the corresponding joint in the output image - by placing the 1 in a heatmap of size 64×64 instead of in a heatmap of size 256×256, which is later resize to 64×64, we ensure, that the 1 is not lost once resized. Next, a Gaussian filter is used to smear out the heatmap, where the standard deviation of the Gaussian filter depends on the visibility, v, of the joint: if the joint is visible, that is if $v = 2$, then the standard deviation is set to be 0.5, whereas the standard deviation of the Gaussian filter is set to be 1 if the joint is not visible, as there there are more uncertainty with the labeling of such keypoints. We do all of this for all of the 17 joints for each image, resulting in the keypoints which will be used for developing our model.