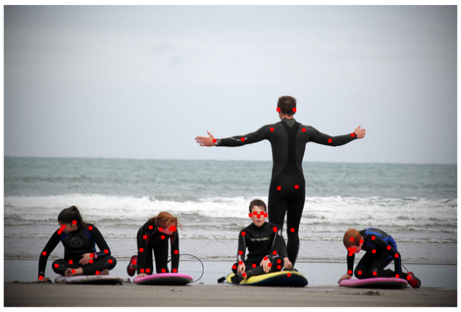
1 The Dataset

To perform the pose estimation, we need some data to train, validate and test our model. Throughout this section the used data is described and preprocessed.

1.1 The COCO Dataset

Figure 1: Example of image from the COCO dataset with labels



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

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 use. 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. Each joint corresponds to three sequential elements in the array, where the first index tells the x-location of the joint in the image, the second index tells the y-location of the joint in the image, and the third index is a flag, v, telling the visibility of the joint in the image. v has three outcomes; if $v = 0$, then the joint is not labeled, if $v = 1$, then the joint is labeled but not visible, and if $v = 2$, then the joint is visible and labeled.
The dataset is split 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, why it is unusable for our purpose, as our model will be doing supervised learning, where the labels are needed. 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. 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 ration, 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 for two reasons

1. 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 a when labeling a visible joint.

2. Each joint could correspond to multiple pixels in the image, hence why it is not correct to only use a single pixel as the location of the joint in the image, which is the current case.

1.2 Data Preprocessing

1.2.1 Creating the test dataset

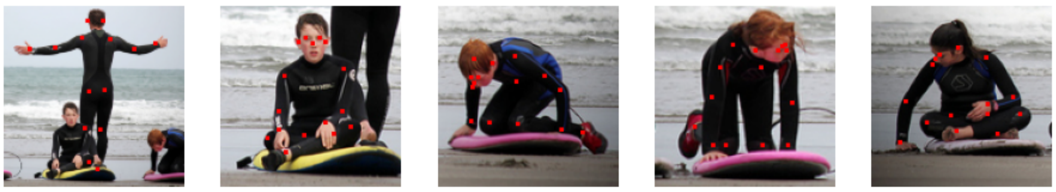
To create the dataset which will be used for testing, we take the training set, since it is the larger of the training set and the validation set, and sample 5.064 images randomly without replacement, to create a test set. This ensures that the test-set and validation-set are of the same size. This new 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 final model.

1.2.2 Preprocessing the images

Figure 2: Data distribution

	Amount of images	Percentage
Training set	5.064	3.658
Validation set	5.064	3.658
Testing set	118.304	92.684
Total	**138.432**	**100**

Figure 3: The results of processing the image from Figure 1 with the corresponding labels

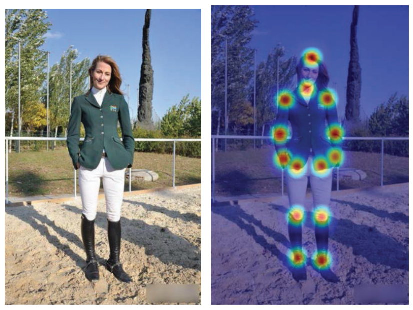


We start the preprocessing of the images by creating multiple bounding boxes, where each bounding box surrounds a single person, which is done by making use of the boundig box annotations provided by COCO. Then, each bounding box is transformed into a square by making the shorter sides have the same length as the longer sides - this is done to ensure that the aspect ratio of the image is kept, when it is later resized.
An issues can happen, where the bounding box still contains multiple people, which will confuse our model, since it does not know which person it should annotate. 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. This 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 current bounding boxes would 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, keeping the annotated person centered along the x-axis.
When all of the above is done, the image is finally croped to each bounding box, resulting in multiple squared images, each containing an unique person. Each of these squared images are then resized to a 256×256 image and saved. Doing all of these steps results in the distribution

of images displayed in Figure 2. In Figure 3 the results of processing the image from Figure 1 are shown with the corresponding labels.

1.2.3 Handling the labels

Figure 4: An example of the heatmaps of a single image fused together and put over the original image [1]



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

For each image of a single person our model outputs 17 heatmaps, one for each possible joint in the image, which tells the probability of the joint being in each pixel. An example of a heatmaps can be seen in Figure 4.
The heatmap of a single joint is created firstly by initializing an all-zero 2D array with size 256×256 for each of the 17 heatmaps. Next, in the ith 2D array at position (x_i, y_i), corresponding to the position of the ith joint, a 1 is placed - this 1 now corresponds to where the ith joint is placed in the image according to the keypoint annotation of the image.
Next, a Gaussian filter is used to smear out the image, where the standard deviation depends on the visibility of the joint; if the joint is visible, then the standard deviation is 0.5, whereas the standard deviation is 1 if the joint is not visible, since we are more unsure if the joint has been labeled correctly.
Lastly, as the model outputs 17 64×64 heatmaps, our heatmaps are resizes from a dimension of 256×256 to a dimension of 64×64.
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.