1 Stacked Hourglass, PCA and K-Means

In this section the various algorithms and architectures used throughout this thesis is described and explained in details.

1.1 Stacked Hourglass

When performing the pose estimation in section **REFERENCE MANGLER**, we will be implementing and using the *stacked hourglass* described by Newell *et al.* [1]. The following description and explanation of the architecture is based on an interpretation of Newell *et al.* [1] and Camilla Olsen [2].

1.1.1 Reasoning behind using the Stacked Hourglass

We have decided to make use of the Stacked hourglass described by Newell *et al.*, as it is an architecture that has shown state-of-the-art results. At the same time the architecture of the network is similar to the architecture of *autoencoders*, making the architecture useful for encoding the data into a lower dimension, which can be useful in section **REFERENCE MANGLER**, when we will be doing the interpretation of the model.

1.1.2 The Residual Module

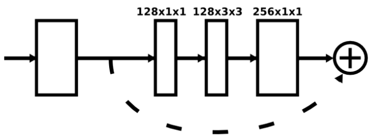

Figure 1: Visualization of the residual module [1]

The Stacked hourglass makes heavily use of so-called *residual modules*, one of which is visualized in Figure 1. The module works by taking an input, which is sent through a 1×1 and a 3×3 convolution, each with 128 channels. Then, the 128 output featuremaps are sent through a 1×1 convolution with 256 channels. Lastly, element-wise addition is then used to add the 256 output featuremaps to the input of the module, which the module then returns. All convolutions are followed by an acitvation function and are *same convolutions*, meaning the output featuremaps are of the same dimensions as the input featuremeaps.

1.1.3 The Hourglass

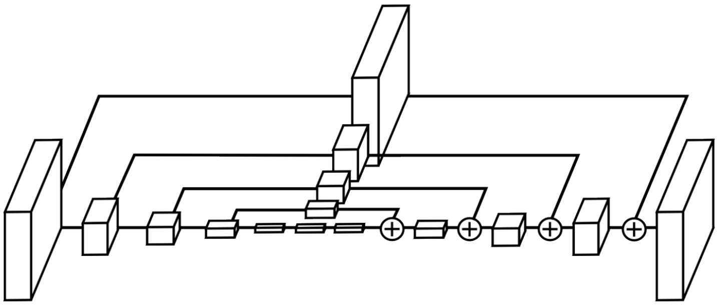

Figure 2: Visualization of a single hourglass [1]

The Stacked hourglass consists of hourglasses, where each hourglass is split into an encoder, where the featuremaps is downsampeld, and a decoder, where the featuremaps are upsampled. The hourglass is symmetric, in the sense, that it has an equal amount of downsampling layers in the encoder as there are upsampling layers in the decoder. In Figure 2 a single hourglass han been visualized, where each box is a residual module.

The hourglass works by using residuals and max poolings to process features down to a low resolution. Then, nearest neighbor upsampling is used to upsample the featuremaps until the featuremaps have the same dimensions as the input of the hourglass. Before each max pool in the encoder, the network branches off and applies a residual. The output of this residual is then added back element-wise to the corresponding level in the decoder, which helps to ensure that lost information from the encoder is kept. This is then fed into a residual in the decoder.

Between the encoder and decoder the network has a bottleneck, where no downsampling or upsampling happens, instead only residuals are processing the featuremaps.
After the decoder two 1×1 convolution layers er applied to produce the final predictions of the network.

1.1.4 The Stacked Hourglass

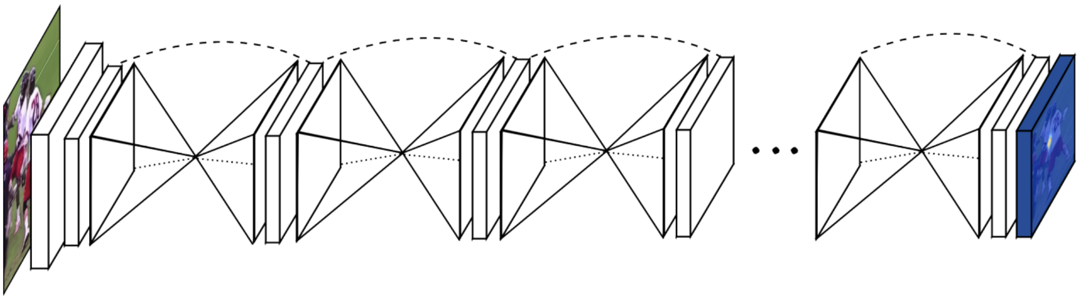

Figure 3: Visualization of the Stacked hourglass [1]

The full network is build by stacking multiple hourglasses end-to-end, making the output of one hourglass be the input of the next hourglass, as shown in Figure 3, which makes each hourglass reevaluate estimates. To evaluate each hourglass, intermediate supervision is used by applying a loss to each hourglass' intermediate prediction.

The input of the network is a 256×256 RGB-image. To lower the memory usage, the network starts off with a 7×7 convolution layer with stride 2, followed by a residual module and max pooling to bring the resolution down to 64×64, which is then input to the first hourglass.

By the end the whole network outputs n heatmaps corresponding to the n joints it should predict for a single person. The prediction of a joint is thus the maximum activation of the corresponding heatmap.

1.2 Principal Components Analysis (PCA)

1.3 K-Means Clustering

K-Means is a unsupervised method used for clustering observations into K groups of similar observations, such that no observation occurs in multiple clusters. In the middle of each cluster is a synthetic observation (that is, not a real observation), called the *centroid*, which is defined

Algorithm 1 K-Means [3]

```
Require: Input data $\boldsymbol{X}$
Require: Amount of clusters $K$
Let $\boldsymbol{\mu}_k$ be a matrix of the coordinates of the centroids of the $k$ clusters. Usually intially set with random values
Let $\boldsymbol{Z}_{nk}$ be a matrix of binary indicator variables that is $1$ if object $n$ is assigned to cluster $k$ and $0$ otherwise
while $\boldsymbol{Z}_{nk}$ is changed between each iteration do
    for each observations $\boldsymbol{x}_n \in \boldsymbol{X}$ do
        Find the centroid, $k$, that $\boldsymbol{x}_n$ is closest to
        Let $\boldsymbol{Z}_{nk} \leftarrow 1$ and $\boldsymbol{Z}_{nj} \leftarrow 0$ for all $j \neq k$
    Update centroids: $\boldsymbol{\mu}_k \leftarrow \frac{\sum_n \boldsymbol{Z}_{nk}\boldsymbol{x}_n}{\sum_n \boldsymbol{Z}_{nk}}$
```

as the mean of the cluster. The pseudocode of the algorithm has been visualized in Algorithm 1. The algorithm is an iterative process, which works firstly by assigning each observation to the closest centroid. Next, each centroid is updated accordingly. This is done until the assigning of each observation is unchanged [3].

K-Means is guaranteed to converge to a local minimum of the total distance between the objects and their corresponding centroid, however, it is not guaranteed to reach the global minimum. This only depends on the initial position of the centroids. To partly overcome this problem it is common to run the algorithm multiple times with different random initial positions of the centroids and use the best solution as the final output [3].