1 Improving the Model

1.1 Motivation

In Section **??** we explored the developed model from Section **??**. By doing so we found out, that the latent space of the model has some inconsistency of the placement of the training observations, as well as having some principal components that acts as noise, which could explain the not optimal performance of the model. By helping improving the latent space of the model, as well as removing some of the noise, we could improve the performance of the model, which can be done by making use of an autoencoder.

1.2 Configuration Details

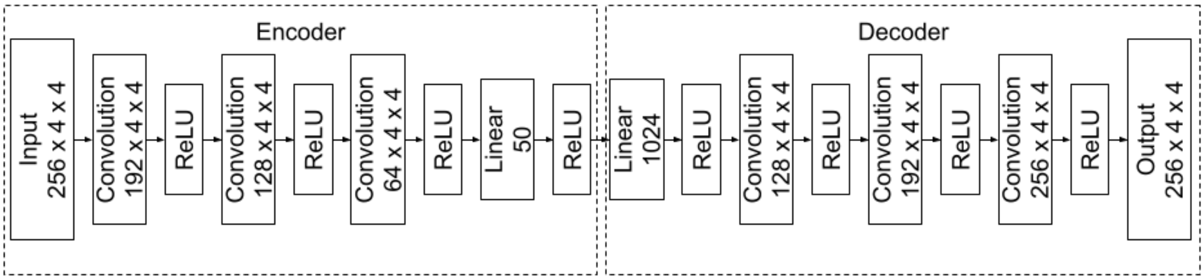

Figure 1: Visualization of the architecture of the developed autoencoder. The numbers in each box represents the dimensions of the output of the corresponding layer.

The developed autoencoder has been visualized in Figure 1. The autoencoder makes use of convolutional layers and linear layers to downsample the input down to only 50 dimensions. We chose 50 as the dimensions of the bottleneck, as we saw in Section **??**, that principal component 50 and above acts as noise.

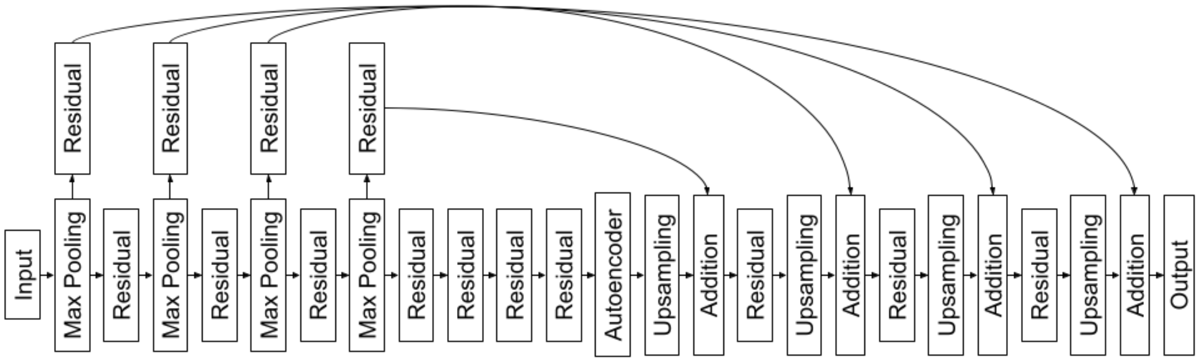

Figure 2: Visualization of the proposed combination of the hourglass and autoencoder for the stacked hourglass.

The autoencoder takes the output of the third residual of the bottleneck of our developed stacked hourglass as input, hence why the autoencoder will be placed after the third residual to form a new proposed hourglass for the stacked hourglass, as visualized in Figure 2.

The training of the new model consits of two parts to speed up the training. First the autoencoder is trained isolated. Then, the trained autoencoder is placed in the developed stacked hourglass, following the structure of Figure **??**, and the whole network is trained.

The autoencoder is trained using Stochastic gradient descent with Nesterov momentum, MSE as the loss function and a learning rate of $5e-4$, which is halved every 25th epoch. To increase the robustness of the autoencoder, we add noise sampled from

$$\mathcal{N}\left(0, x^2 e - 2\right)$$

to each training sample, where x is the value of the training sample. To help the model converge, we sample from a Glorot normal distribution, like in the case with the stacked hourglass.

After the autoencoder has been trained, the whole network is further trained by following Newell *et al.* [1] as described in Section **??**.

1.3 Results

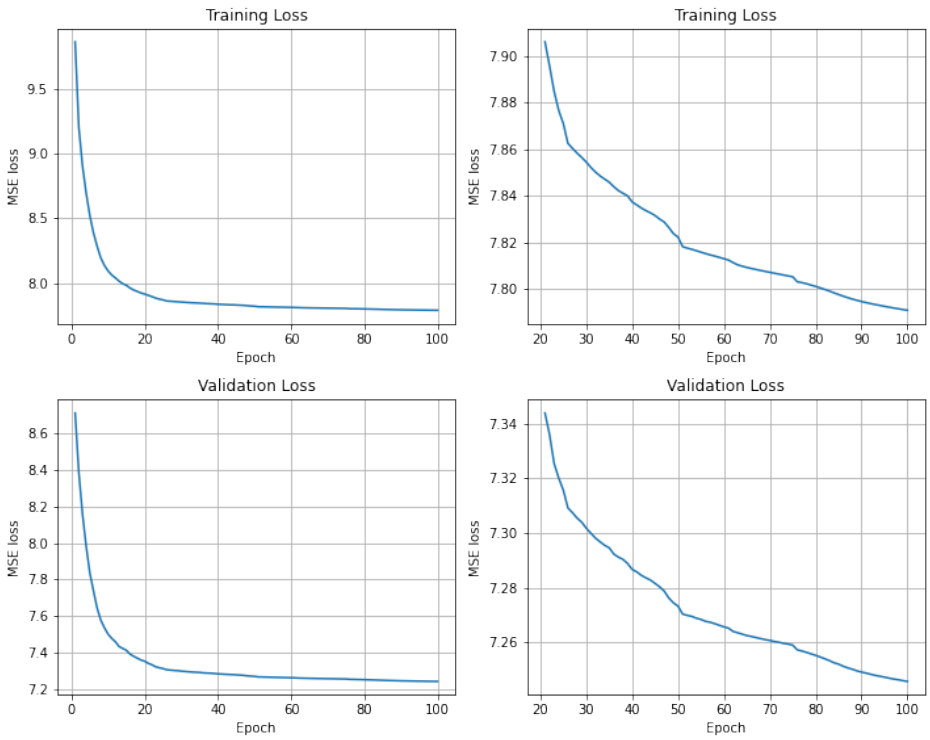

Figure 3: Visualization of the evolution of the training- and validation loss of the autoencoder during training. The left column shows all of the 100 epochs. Right column shows epoch 21 and foward

By training the autoencoder isolated, we get the evolution of the training- and validation loss visualized in Figure **??**. We can clearly see, how the model does not start to overfit, as in the case when we trained the stacked hourglass. We decided to stop the training of the autoencoder, as each update only yielded minor changes to the model. The evolution of training the stacked hourglass with the autoencoder has been visualized in Figure 4. By looking at the figure we can see how the combined stacked hourglass and autoencoder initially performs worse than the original stacked hourglass, however, as the training continues it beats the original stacked hourglass, resulting an validation PCK accuracy of 46.7% - an increase of 7.8% or 3.4 percentage points compared with the original stacked hourglass.

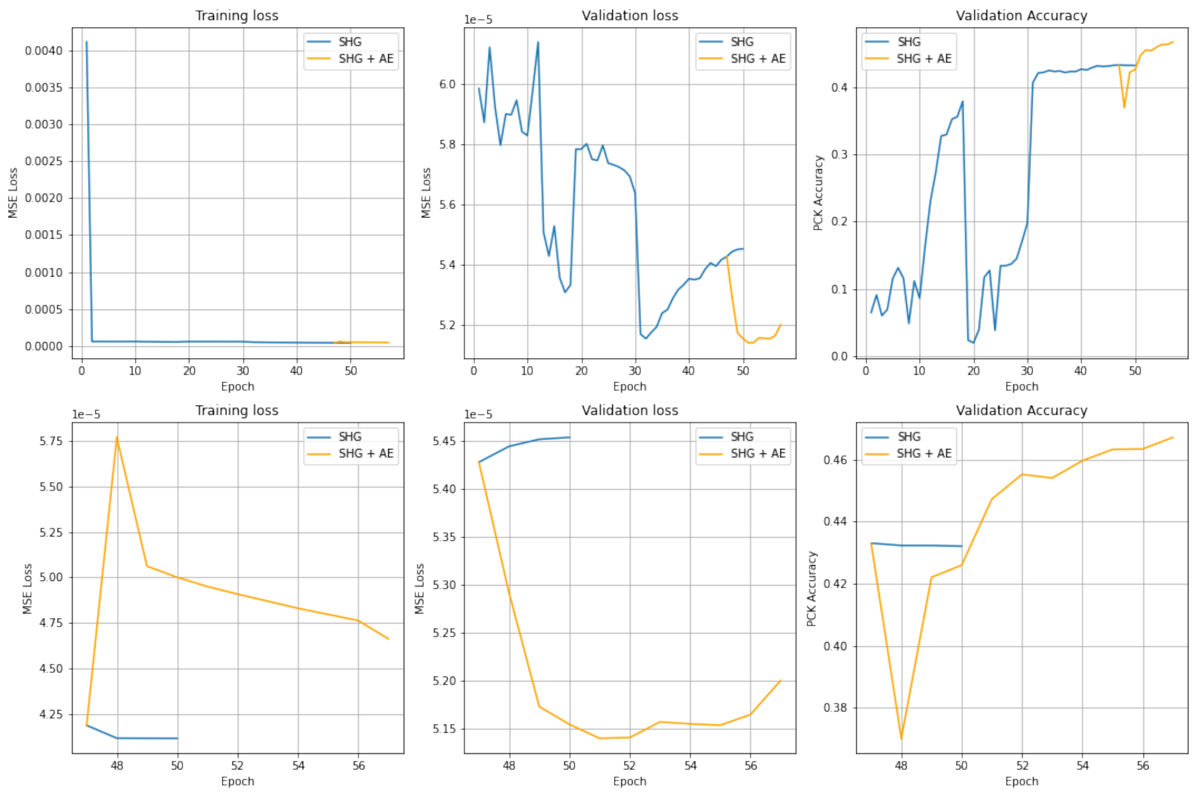

Figure 4: Visualization of the evolution of the training- and validation loss, as well as the PCK validation accuracy of the combination of the stacked hourglass and autoencoder, compared with the evolution of training the orignial stacked hourglass. The top row is of all of the 57 epochs. The bottom row shows epoch 47 and forward.