1 Discussion

In the following we will be discussing our procedure as well as our results. We will start off in Section 1.1, where we will be giving a brief summary of our obtained results from Section **??**, **??** and **??**. We will then in Section 1.2 compare our results with the results of Newell *et al.* [3] and Olsen [4], as well as discuss the differences in the results. In Section 1.3 we will argue and discuss why the autoencoder improved our initial model. Lasty, we will in Section 1.4 comment on potential future work in relation to this project.

1.1 Summary of the Obtained Results

In Section **??**, we succefully implemented and trained a Stacked hourglass, consisting of a single hourglass. We did this by following the configuration details described by Newell *et al.* [3] and Olsen [4]. The developed model has a validation PCK accuracy of 0.433 and a test PCK accuracy of 0.441.

In Section **??** we gained an understanding of how the devleoped model works by exploring the different components of the model. We could verify, that the skip-connections of the model were used for recreating details that are lost during the encoder-phase of the model, as claimed by Newell *et al.* [3] and Olsen [4]. We then used PCA to gain an understanding of the structure of the latent space of the model. By doing so we came to the conclusion, that the first principal component of the latent space is used for deciding if a given person is sitting down or standing up,, as well as possibly discovering some redundancy in the model, as principal componet 50 and above seemed to act as noise. Lastly, we clustered the latent space to gain an understanding of how the model works. By doing so we learned, that the model knows the difference between fully-annotated people and not-fully annotated people, as well as knows the difference between stationary people and moving people. Here we also identified some possible reasons for inaccuracies of the model, as these groupings are not always correct.

In Section **??**, we used our knowledge of the model to improve the performance of the model. This was done by developing and training an autoencoder, which was placed in the model. We decided to make use of an autoencoder, as we knew from the previous section, that the latent space of the Stacked hourglass contained a lot of noise, as well as some missclassifications, which the autoencoder possibly could help on. By doing so, both the validation and test PCK accuracy increased to 0.467 and 0.473, respectively.

1.2 Comparison of Models

Description	# stacks	Testing accuracy
Olsen - $2A$	2	0.72
Olsen - $2B$	2	0.81
Olsen - $2M$	2	0.83
Our SHG	1	0.469
Our modified SHG	**1**	**0.576**

Table 1: Comparison of our developled models with Olsens developed model [4]. The testing accuracy is only based on fully-visible joints (that is, where $v = 2$)

To further understand the performance of our developed models, we have in Table 1 compared our models with the models developed by Olsen [4], which consisted of 2 stacks.

By looking at Table 1 we see, that the performances of the models developed by Olsen [4] exceeds the performances of our developed models. However, we also see, that all of Olsen's [4] models use more stacks, than our models, which could explain the better performance. This is also argued by Newell *et al.*, as they describe how stacking multiple hourglasses, increases the performance of the model [3].

Another reasoning behind the loss of performance compared with the models developed by Olsen [4] is, that Olsen's models were only trained on fully-visible joints [4], whereas our models were trained on both fully-visible joints and on non-visible joints. This could make Olsen's models [4] perform better on fully-visible joints, as this probably makes the model more focused on learning fully-visible joints during training. We would thus argue, that if we were to retrain our models on only fully-visible joints, the gap between the performance of our models and Olsen's models [4] would become smaller, and even very small in the case of the modified Stacked hourglass. This is however not possible to predict and thus only purely speculation, however, our thought experiment does support this claim.

Olsen's models and data are more similar to ours, than the ones used by Newell *et al.* [3], making the comparison more accurate. However, we can still compare our results with the ones obtained by Newell *et al.* [3] to get a further performance evaluation of our models.

Like in our case, Newelll *et al.* do also develop a model consisting of only a single hourglass. However, Newell *et al.* does not report the testing accuracy of this model. Instead, they report the evolution of the validation accuracy during training, for joints that are not associated with the head and torso. The validation accuracy seems to top at around 0.65 [3]. If we test our models on the joints from our testing data, that are not associated with the head or torso, we get a PCK accuracy of 0.32 and 0.38 for the Stacked hourglass and the modified Stacked hourglass, respectively.

There are a number of reasons that could explain the poor performance of our models compared with the models developed by Newell *et al.* [3].

1. We use a different dataset than Newell *et al.* [3]: Newell *et al.* uses the FLIC [5] and MPII Human Pose [1] datasets, whereas we use the 2017 Microsoft COCO dataset [2]. These datasets could potentially be different from the dataset that we use, making it an unfair comparison to compare our results with Newell *et al.*'s results [3]. For instance, the FLIC [5] and MPII Human Pose [1] could contain less occlusion than the Microsoft COCO dataset [2], which would decrease the performance of our model, as argued by Newell *et al.* [3].

2. Our preprocessing of not-visible joints was different than the preprocessing done by Newell *et al.* [3]: when we preprocessed the data, we made use of a Guassian filter to represent uncertainty in the annotation. For not-visible joints we used a standard deviation of 0.5, whereas we for visible joints used a standard deviation of 1. Newell *et al.* describes, how they for all joints used a standard deviation of 1 [3]. We decided to use a standard deviation of 0.5 for not-visible joints, as the uncertainty of the annotation of these joints is greater, than for visible joints. This could however also possibly have confused the model, as it thus implicitly also learns to predict whether a given joint is visible or not, making the complexity of the model greater, possibly decreasing the performance of the model.

3. We use a different PCK accuracy metric: Newell *et al.* state, that they use different PCK accuracy metrics - that is PCK accuracy metrics, where the normalization constant is different - at different times [3], however, they do not state which PCK accuracy metric

they used for validation during training. We chose a normalization constant equal to one tenth of the heatmap size, however, if Newell *et al.* [3] used a greater normalization constant, the predictions would have a bigger probability of being accepted as correct, making the accuracy of the developed model seem greater than it actually is. Likewise, Newell *et al.* [3] do not state the threshold radius they use. If they were to use a threshold radius greater than the one we use, their estimations have a greater probability of being counted as correct, than our estimations.

4. Newell *et al.* describes that they use batch normalizations to prevent the model from overfitting, however, they do not describe where they used these batch normalizations [3]. If we were to place our batch normalizations at places different than where Newell *et al.* [3] place theirs, we could make our model overfit much quicker than what Newell *et al.* [3] experiences, resulting in worse performance. This, however, is much less likely, as we decided to follow Olsen [4] for the placement of batch normalization and they achieved great results, hinting towards their, and thus also our, placements of the batch normalizations being correct.

5. We use different configurations: Newell *et al.* [3] does not state the decay rate ρ for RMSProp. We decided to use a decay rate of 0.99, as this is the default value in PyTorch, resulting in a slow decay of the accumulated squared gradient $\boldsymbol{r}$. If Newell *et al.* [3] use a different value for the decay rate, $\boldsymbol{r}$ would most likely decay differently, potentially resulting in different results. Likewise, Newell *et al.* [3] does not state how they initialize the parameters of their model, hence why, the differences in performance could be explained by a potentiel difference in how they initialize the parameters in their model and in how we initialize the parameters in our model. This is however, not likely to have a big impact on the results, as Olsen [4] performed the same method for initializing the parameters of their model, received great results, making us believe, that our method for initialization should not be a problem.

Overall, the comparison between our developed model and the model developed by Newell *et al.* [3] is difficult, as they have a lot of differences. The comparison between our model and Olsen's model [4] is thus more accurate, however, still inaccurate, as Olsen's models use 2 stacks instead of just 1 as in our case [4].

1.3 Why did the Autoencoder Improve the Stacked hourglass?

In Section **??** we argued the following two reasons for why using the autoencoder could improve the performance of the model: (1) inconsistency of the placement of the training observations, and (2) having principal components that acts as noise. In the following we will be analysing and discussing our results of using the autoencoder, leading to if these two reasons actually had an effect on the performance of the model.

1.3.1 Clustering the Latent Space

In Figure 1, we have visualized the results of performing K-Means on the latent space of the autoencoder with $K = 2$. Due to memory limitations only 10.000 training samples were used.

In Figure **??** we visualized equivalent results of performing the same procedure on the latent space of the Stacked hourglass. In correlation to Figure **??** we argued, that the two clusters contained some missclassifications, which could explain the poor performance of the model. More specifically, *Cluster 1*, the cluster containing samples with a lot of keypoints missing, contained a lot of samples that should ideally have been in *Cluster 0*.

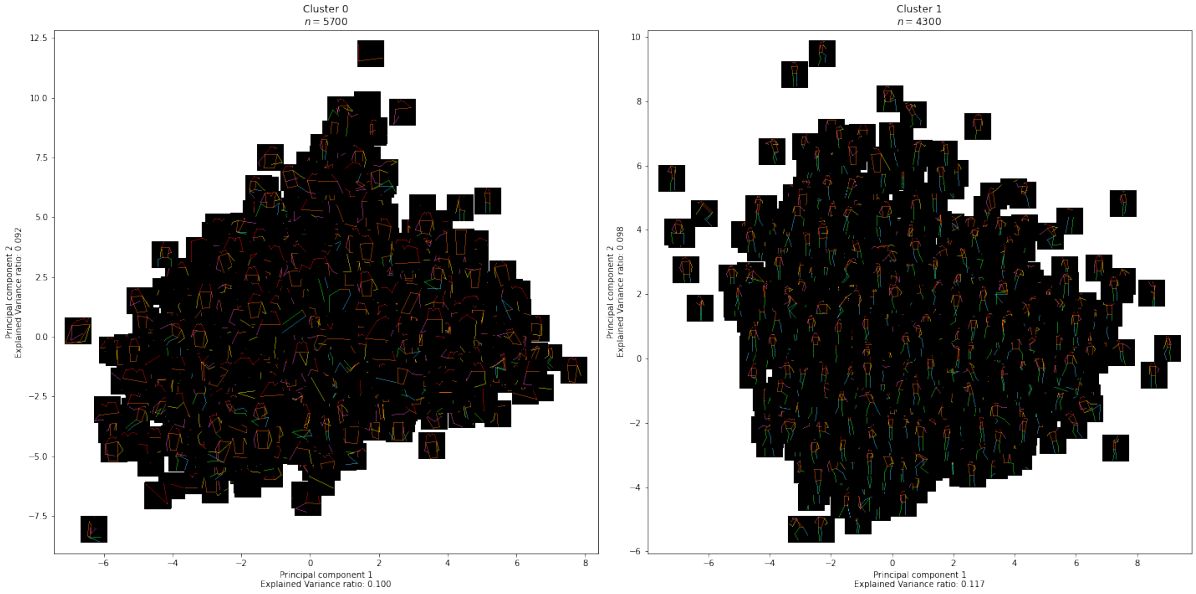
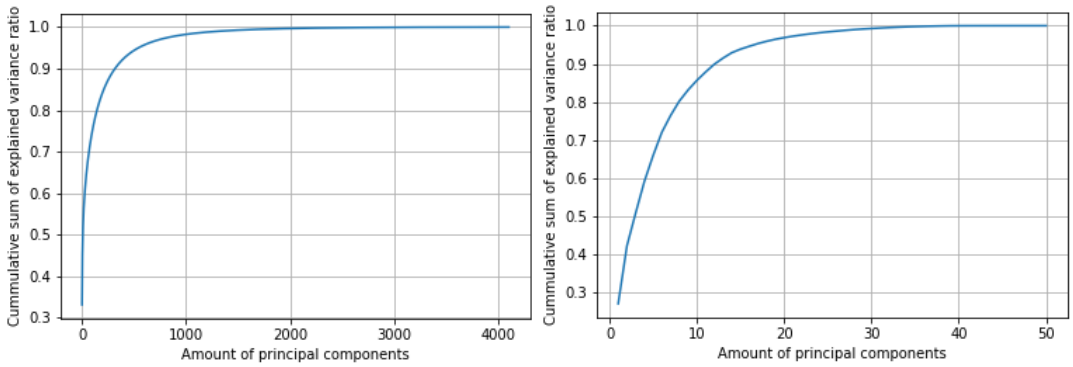


Figure 1: Results of performing K-Means clustering on 10.000 random samples of the latent space of the autoencoder

If we compare Figure 1 with Figure **??** we can see, that the clusters in Figure 1 contains fewer missclassifications than what was the case in **??**. More specifically we can see, that *Cluster 0* in 1, the cluster containing samples with a lot of keypoints missing, contains fewer samples that should have been in *Cluster 1*, than what was the case earlier. Likewise, the distribution of the amount of samples in each cluster is also more balanced, leading to believe that fewer missclassifications are indeed happening.

On the other hand, the 10.000 samples used in Figure 1 and **??** are not the same. Both figures contains 10.000 random samples out of the total 124.040 samples, meaning the two figures most likely are not alike, however, probably have some overlap. This leads to us believe, that the two figures are not exactly comparable, however, will still give a decent insight into the clustering of the two latent spaces.

1.3.2 Removing Noisy Features from the Latent Space



(a) Stacked hourglass (b) Autoencoder

Figure 2: Cummulative sum of the explained variance ratio per principal component of the latent space of the Stacked hourglass and autoencoder, respectively

In Section **??** we argued, that the latent space of the developed Stacked hourglass contained a lot of principal components that acted as noise. By using an autoencoder this noise should be removed, since the autoencoder learns the most important features of the input data, as the output of the autoencoder is only an approximation of the input.

In Figure 2 we have visualized the cummulative sum of the explained variance ratio per principal component of the latent space of the Stacked hourglass and the autoencoder for the training data. We can clearly see how the autoencoder needs a lot less principal components to represent the data, than the Stacked hourglass. If we assume, that 5% of variance of the data acts as noise, the autoencoder needs 16 principal components to explain the remaining 95% of the variance of the data, where the Stacked hourglass needs a total of 541. This means, that by using the autoencoder we have removed a lot of redundancy in the form of noise, essentially denoising the data, which could explain the improvement of the performance.

To sum up, it seems, that we by using the autoencoder in the Stacked hourglass, have improved the structure of the latent space of the model, as well as removed the noise of the latent space, which seems to have improved the performance, as argued in Section **??**.

1.4 Future Work

If we were to work further with this project, it would be ideal to explore the effects of stacking multiple modified hourglasses end-to-end. By doing so we would not only hope that the performance of the model to increase further, but we would also hope we could obtain the same accuracy as Newell *et al.* experiences [3], however with fewer stacks. For instance, we could hope that by stacking 2 modified hourglasses, we would achieve the same results as Newell *et al.* [3] achieves with 4 hourglasses. In correlation to this, it would also be ideal to examine if any redundancy is added when multiple modified hourglasses are stacked. For instance, if we were to stack 2 modified hourglasses, maybe only 25 filters are needed in the second hourglass.

Secondly, one could check for redundancy in the other parts of the model. In Section **??** and Section **??**, we found redundancy in the bottleneck of the model, however, we did not search for any redundancy in the other parts of the model, such as the encoder, decoder and skip-connections, leaving some potential future work. In addition to this, one could retrain a new Stacked hourglass without skip-connections, unilke our procedure in Section **??**, where we simply disabled the skip-connections of an already trained network. This would give less biased results, as well as better insight into the effects of the skip-connects, in comparison to our procedure.

Lastly, it would be interesting to get an understanding of how much of a performance increase one could obtain by using the autoencoder in the hourglass. We described in Section 1.3, how the latent space of the autoencoder still has some missclassifications, hinting towards some potential increase of performance. If one could correct these missclassifications, how much of an increase in performance would this result in?