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 be discussing our interpretation of the Stacked hourglass, including our approach as well as the results. In Section 1.4 we will argue and discuss why the autoencoder improved our initial model. Lasty, we will in Section 1.5 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.* [3] and Olsen [4], as they describe how stacking multiple hourglasses, increases the performance of the model.

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.* do 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 non-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 non-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 non-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 who 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. 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 as we did and received great results, making us believe, that our method for initialization should not be a problem.

Overall, our developed models seem to perform worse than the models developed by Olsen [4] and Newell *et al.* [3]. However, the comparisons are difficult, as the procedures have a lot of differences.

1.3 How Good was the Interpretation of the Stacked Hourglass?

In Section **??** we interpreted the Stacked hourglass developed in Section **??**. In the following we will be discussing our approach, as well as the obtained results.

1.3.1 Verification of the Skip-Connections

In Section **??** we verified the effects of the skip-connections of the Stacked hourglass. This was done by disabling the skip-connections of the already trained Stacked hourglass and comparing the corresponding results with results of the the Stacked hourglass with the skip-connections enabled. This approach however, is not optimal, as we will be having some biased results, due to the fact, that the network was trained with the skip-connections enabled. To get an full understanding of the effects of the skip-connections, we should have trained a Stacked hourglass with the skip-connections disabled, whose results should have been compared with the Stacked hourglass with the skip-connections enabled. We decided to not train a Stacked hourglass with the skip-connections disabled due to time constraints, as well as the fact, that our approach does give a brief view into the effects of the skip-connections.

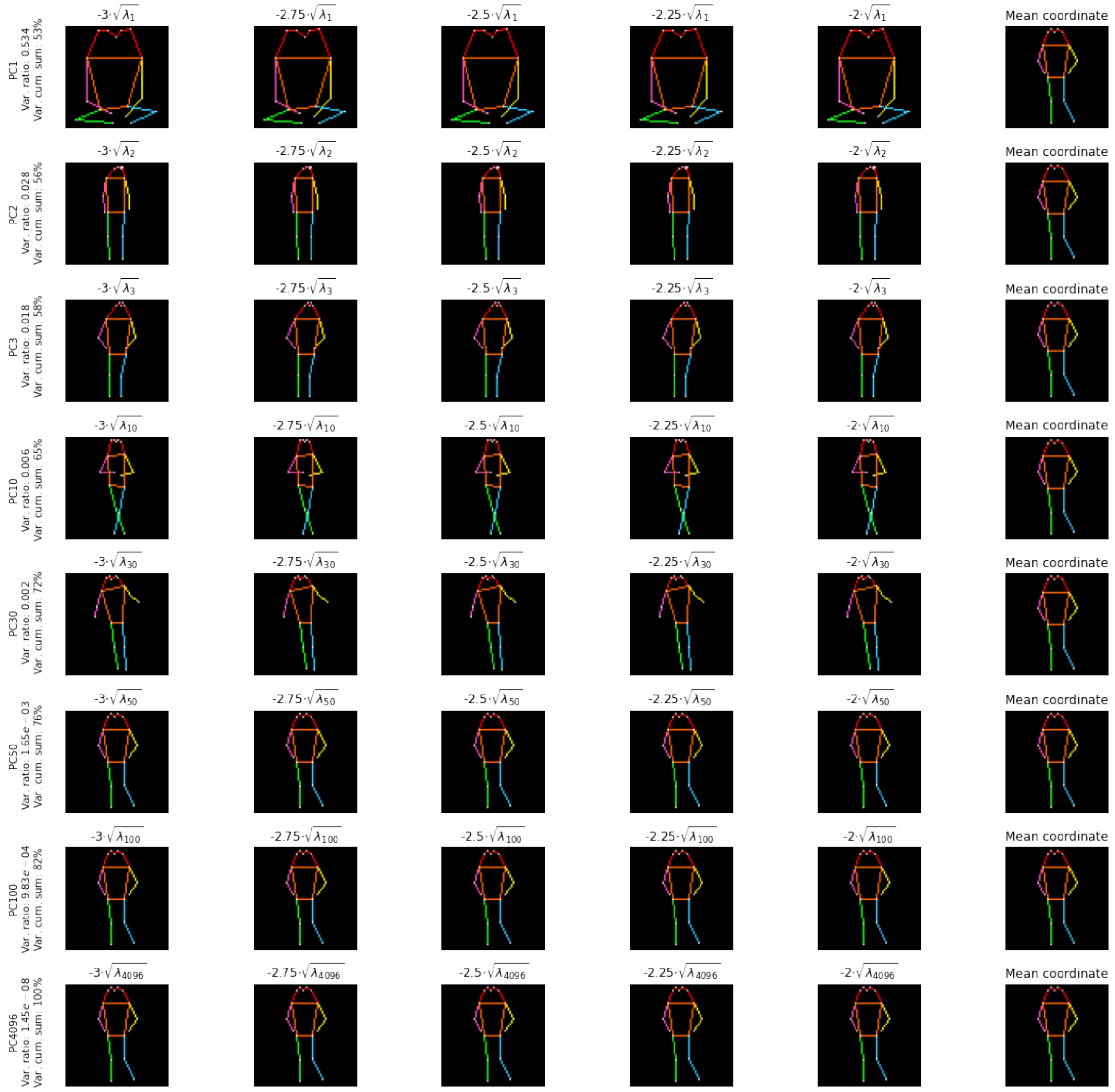


Figure 1: Nearest observation to the end of "walking" along various principal component with small step sizes.

1.3.2 Dimensionality Analysis of the Latent Space

In Section **??** we looked at the effects of various principal components of the latent space of the developed Stacked hourglass. We did this by "walking" along a given principal component with a step size of the form $c \cdot \sqrt{\lambda}$, where c is a constant and λ is the explained variance of the corresponding principal component. By looking at the results of walking in Figure **??** we see, that all of the results of walking in negative direction with a step size of $c = 3$ yield the same results as walking in negative direction with a step size of $c = 2$. This hints towards the increase of c being too great and we thus are missing some important information, which potentially could disproof our claim that principal component 50 and forward are just noise. However, we have in Figure 1 visualized the effects of "walking" with a smaller increase of c to explore the data in between $c = 2$ and $c = 3$. By doing so we can clearly see, that we did not miss any important information when we performed the dimensionality analysis in Section **??**, as the extra steps between $c = 2$ and $c = 3$ do not add extra information.

1.3.3 Clustering the Latent Space

In Section **??** we used clustering on the latent space of the developed Stacked hourglass to get an understanding of how the model works. Due to memory constraints we only performed the clustering on 10.000 training samples of the total 124.040 training samples. One could argue, that this would result in unreliable results, as only a minor subset of the dataset is used and clustering on the whole dataset could yield different results, such as the optimal amount of clusters would be more than 2 according to the Silhouette score. This is not easily verifiable due to the memory constaints, however, if we take a look at *Cluster 0* in Figure **??**, we can see, that the cluster can be somewhat split into two parts; one minor part in the top left and one greater part in the center/bottom left, hinting towards the true amount of clusters is actually 3 instead of 2. It is also important to note, that the principal components used for the visualization only explains about 15.3% of the variance in the data, making it possible, that the data actually consists of clusters which we cannot see. This does not verfy the argument of the unrelaible results, however, it does definitely support it. We however still decided to only use $k = 2$, as this was suggested by the Silhouette score, as well as this yielded usable results. Whether the increase of observations would have an effect on the clustering and if the data actually consists of more than 2 clusters is difficult to say, however, it is definitely possible.

1.4 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.

1.4.1 Clustering the Latent Space

In Figure 2, 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*.

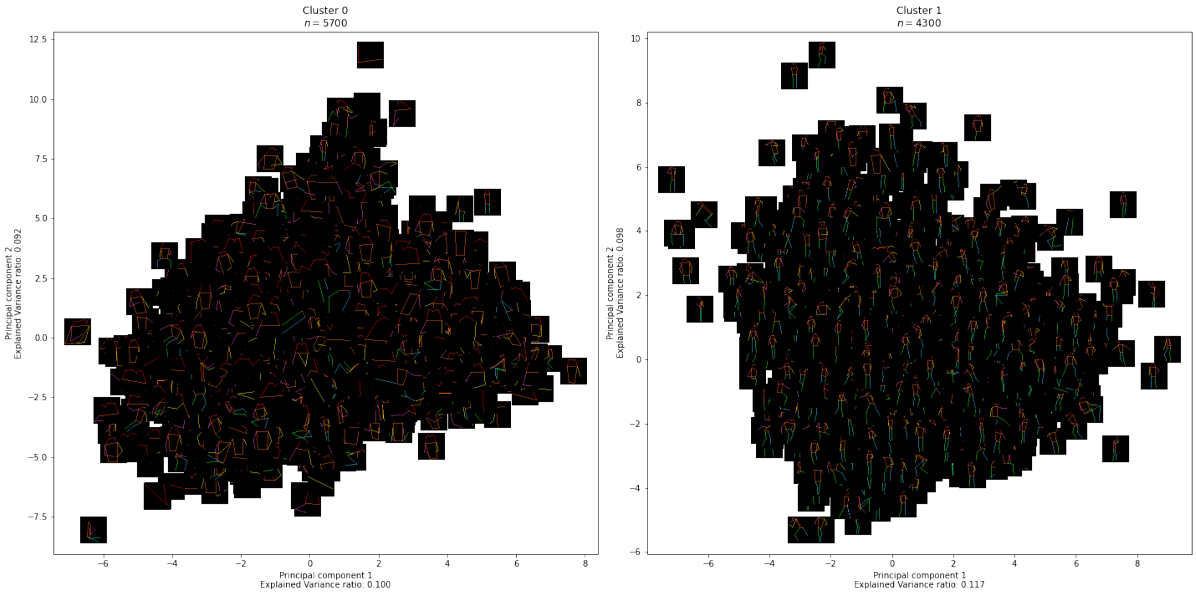


Figure 2: Results of performing K-Means clustering on 10.000 random samples of the latent space of the autoencoder

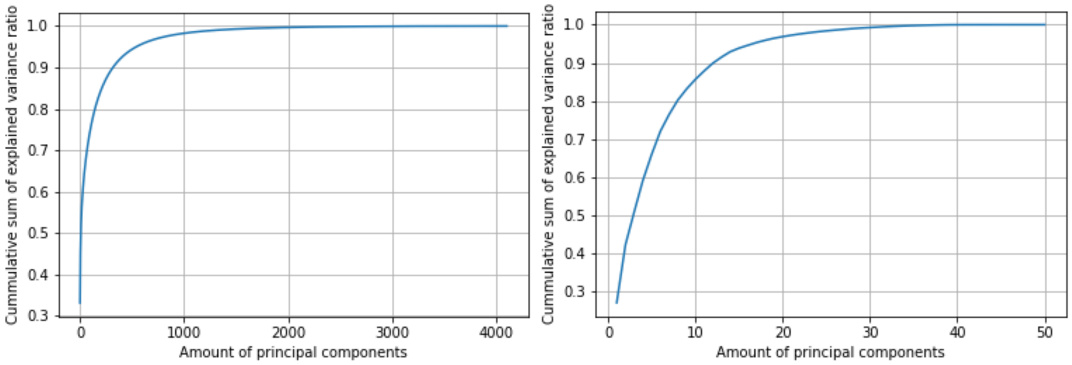
If we compare Figure 2 with Figure **??** we can see, that the clusters in Figure 2 contains fewer missclassifications than what was the case in **??**. More specifically we can see, that *Cluster 0* in 2, the cluster containing samples with a lot of keypoints missing, contains fewer samples that should 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 2 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.

1.4.2 Removing Noisy Features from the Latent Space

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 3 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.



(a) Stacked hourglass

(b) Autoencoder

Figure 3: Cummulative sum of the explained variance ratio per principal component of the latent space of the Stacked hourglass and autoencoder, respectively

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 **??**.

1.5 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.* [3] or Olsen [4] experience, 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 find 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 1.4, 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?