# 1 Discussion

The following section discusses our obtained results from the pretraining and finetuning stages, some concerns regarding possible suboptimal configuration choices we have made, as well as suggests some possible future work. The section begins in section 1.1, where we discuss the obtain results. Then, in section 1.2, where we discuss why the models performed much better during the finetuning stage, than during the pretraining stage. Section 1.3 then discusses which model is the optimal choice. In section 1.4 we then discuss some pitfalls of our approach and how we could have made some better choices. Lastly, in section 1.5 we present some possible future work possibilities.

## 1.1 Results

In section **??** and section **??** we successfully implemented and tested the four architectures in various experiments. We will in the following be analysing and discussing the potential reasons for these obtained results.

### 1.1.1 Pretraining

We saw in section **??** how the 3-dimensional convolutional layer tend to perform much better in experiment 2, than it does in experiment 1 (of course with the exception of run 2.1 and 2.2). As the goal of experiment 1 is for the models to both learn translation and scaling, whereas the goal of experiment 2 is just for the models to learn translation, we can clearly see here how big of a difference the task of learning to also scale the data has on the results for the 3-dimensional convolutional layer. Of course, a tiny bit of randomness does also play in here, as the models for the two experiments were initialized with different weights. However, as the weights were sampled from the same distribution, as well as the 3-dimensional convolutional layer having very few weights, we find it hard to believe that this randomness has such a big impact on the results. Thus we claim that the performance differences between the two experiments is due to the task of scaling the data.

For the two architectures that are based on a bidirectional convolutional LSTM, a similar pattern is observed for the shifting scalar, however, this performance difference is much smaller than it was in the case for the 3-dimensional convolutional layer. We simply believe, that this improved performance is due to the bidirectional convolutional LSTMs predicting the position of the keypoints in a more sophisticated manner.

For DeciWatch there are no performance differences between experiment 1 and experiment 2 when considering shifting-scalar $s = 1$. However, this is expected, as unlike the other architectures, DeciWatch works directly on keypoint-coordinates instead of the heatmaps, making it scaling invariant and thus making it the two experiments equivalent to each other. On the other hand however, when considering shifting scalar $s = 2$, there is a clear performance difference between experiment 1 and experiment 2 for DeciWatch, which is very surprising for us, as the model is scaling invariant. However, if we inspect Figure **??** we can clearly see, that the initial performances of DeciWatch 2.1 and DeciWatch 2.2 are much farther away from each other, than the initial performances of DeciWatch 1.1 and DeciWatch 1.2 were from each other. Thus, we suggest that the final performance difference between DeciWatch 2.1 and DeciWatch 2.2 is simply due to the randomness in the weight initalization of the two models.

We further saw in section **??** the effects on the models of halfing the frame rate. For both the 3-dimensional convolutional LSTM and DeciWatch this had a negative effect, whereas it actually had a positive effect for the two models that are based on a bidirectional LSTM. For

DeciWatch we have to keep in mind, that it is already only considering every fifth frame, so by halfing the frame rate it is actually only considering every tenth frame, giving it a lot less context for it to process. For the 3-dimensional convolutional layer and the two models based on a bidirectional convolutional LSTM, we again suggest that the performance difference is due to the later models yielding their predictions in a more sophisticated manner than the simple 3-dimensional convolutional layer. In the case of when the shifting-scalar is $s = 2$ all models generally perform worse in experiment 3 than they do in experiment 1, which we simply believe is due to the data being so noisy, that the models need the finer details to infer their predictions. As the distance-threshold for PCK is increased however, the 3-dimensional convolutional layer actually perform better in experiment 3 than it does in experiment 1. However, we have to note here, that the performance difference is rather small and that it is in the case where the the distance-threshold is at its greatest. Thus, we believe the performance difference is due to the random initialization of the models or the models learning a rough estimate of the keypoints over learning a very fine estimation as this is very difficult when the data is very noisy. We further strengthen our last believe when considering, that the model is the lowest performing model when considering PCK@0.05 but the highest performing model when considering PCK@0.2 and thus generally learns to yield a rough estiamte of the position of the keypoints.

We finally saw in **??** how the models for shifting-scalar $s = 1$ had the most difficulty with the wrists and ankles, whereas they performed the best on the ears and shoulders. However, we find this very expected, as the wrists and ankles tend to be the joints that have the highest amount of movements, especially in sports which our pretraining dataset is centered around, and thus are much more difficult to denoise than more static joints such as the ears and shoulders. On the other hand, in the case of using the shifting-scalar $s = 2$ we saw, that the models struggled the most with the elbows and knees, and performed the best on the ears, nose and the ankles. At first glance, one would find this rather unexpecting as the wrists and ankles are no longer the most difficult keypoints. However, we suggest that this could be caused by these keypoints being too close to the heatmap borders, such that when they are shifted a lot towards any of these borders, they will end up at the border, as they cannot exceed the border. Thus, the wrists and ankles are actually not shifted as far as the other keypoints, such as the knees and elbows which would then be the most difficult keypoints, as these keypoints are both shifted a lot and the corresponding joints contain a decent amount of movement.

### 1.1.2 Finetuning

In section **??** we saw how the 3-dimensional convolutional layer generally converged towards the highest validation accuracy, whereas the DeciWatch-based models and the models based on a bidirectional convolutional LSTM tended to plateau in validation accuracy, which even decreased over time for most of these models. This is most likely unexpected to some people, however, if take a look at the evolution of their training loss and validation loss through the finetuning of these models we can see, that the training loss is almost completely monotonically decreasing for all of these models through the whole finetuning-phase, whereas the validation loss is actually increasing. Thus for these models, where the validation loss is actually increasing, we are actually experiencing that they are overfitting. Further, the ClimbAlong-dataset is very small and these models are much bigger than the 3-dimensional convolutional layer (in terms of tuneable parameters), which means that they have a greater likelihood of "remembering" the ClimbAlong-dataset instead of learning its patterns and thus overfitting. We do however find it odd, that there are two Deciwatch-models that are not overfitting - DeciWatch 2.1 and DeciWatch 2.3. We are not too sure why these two models are not overfitting, however, we do find it interesting, that these two models were actually also the two worst performing model during the pretraining-stage.

In section **??** we saw how the models from the experiments with shifting-scalar $s = 1$ tend to yield better results than the models trained on data with shifting-scalar $s = 2$. We further saw, how the models from experiment 2 generally performs better than the models from experiment 1. Generally however, these performance difference are very minor, which makes sense as the finetuning-data is the same for the two groups. We do note the minor performance differences however, which probably means, that (1) the noise in the finetuning data is more similar to the data with shifting-scalar $s = 1$ than the data with shifting-scalar $s = 2$, and (2) the standard deviation of the peaks of the finetuning data is not changing as much as they do in our experiment 2.

One could look at Figure **??** and claim that the pretraining-stage has had no effect, as the validation accuracy starts off very low. However, we disagree with for two reasons. First off, we just noted that the choice of shifting-scalar from the pretraining-stage had an effect on the final results, so the pretraining-stage must have had an effect. Secondly, most of the models actually reach a very high validation accuracy after just a single epoch, which we believe is due to the pretraining. One could argue, that a similar pattern was observed during the pretraining-stage and is thus just a general pattern. However, we have to remember, that the pretraining-dataset is much bigger than the finetuning-dataset and thus just after one epoch in the pretraining-stage, the models have already been trained on a lot of samples, which explains this major performance-jump during the pretrianing-stage.

Lastly, one saw in section **??** how the models tend to perform the worst on keypoints related to the thumbs, pinkies and index fingers. We see two reasons for this observation. First off, these keypoints were not included in the pretraining dataset, meaning the models have learned to infer the position of these keypoints purely from the finetuning dataset. We believe this reason to only have a minor effect, as if we look at the performance of the models on the keypoints related to the heels and the toes, which were also not included in the pretraining dataset, we see, that the models generally performs only a tiny bit worse on these keypoints, that on the remaining keypoints. The second reason is, that the finger keypoints just have a lot of movement, as the related joints are the joints that are mostly used for bouldering. And as previously stated, keypoints that move a lot are more difficult to predict the position of.

## 1.2 Why did the Models Perform Better during Finetuning than during Pretraining?

All of the finetuned models outperforms their pretrained counterpart. We see a couple of reasons for this.

First off, the models have been trained om more data, as they have been trained on both the pretraining data and the finetuning data, which of course is always positive.

Secondly, the models have been trained and tested on samples that have very similar to each other. On the other hand, the pretraining dataset consisted of people breakdancing, throwing a baseball, bench pressing and performing sit ups, which are all very different from each other, which could make it hard for the models to learn to generalize. Similarly, the samples from the pretraining dataset were all filmed from very similar camera ankles, whereas the samples from the pretraining data were filmed from all kinds of camera ankles, resulting in the same keypoints being placed at different locations

Further, the BRACE dataset was not fully manually annotated, but instead annotated using a pose estimator and only certain incorrectly predicted poses were manually annotated, where incorrectly predicted poses were detected by a machine learning model. However, if either the machine learning model for detecting incorrectly predicted poses or the pose estimator delivered suboptimal results, then the annotations would also be suboptimal and thus contain some noise, which is very difficult for our models to replicate. The finetuning dataset on the other hand was completely fully annotated by humans, making the annotations much more consistent and thus easier for the models to replicate.

We also see a problem with the Penn Action dataset, which can have an effect on the pretraining results. For the BRACE dataset, it is clearly documented, that the video sequences is filmed using 30 frames per second. For Penn Action, on the other hand, a similar information is not stated anywhere and the video sequences come as individual frames without the duration of the video sequences noted, hence why cannot neither compute the frame rate of these video sequences. This could cause some problems, as two windows of frames, one from each dataset, could span two different durations and thus capture two different duration of context, which can confuse the developed models. On the other hand, all of the video sequences of the finetuning dataset are all filmed using 30 frames per second, making the dataset much more consistent, which again makes the fitting easier.

Lastly, one major reason behind the performance differences behind the models on the pretraining dataset and the finetuning dataset is the perforamnce of the identity functions. If we compare the performance of the identity function on the pretraining dataset in Table **??** and Table **??** against the performance of the identity function on the finetuning dataset in Table **??** and Table **??**, we clearly see, that the identity function performs much better during the finetuning than during the pretraining. Thus, a lot less temporal smoothing has to be done by our temporal-inclusive models, making the learning a lot easier.

## 1.3 Which Model is the Best for Incorporating Temporal Smoothing?

As we have now fuly developed and tested all of our models on all three experiments, we can decide which models yields the best results.

As we know from section **??**, all of the models tend to deliver similar great results when considering PCK@0.2. However, generally DeciWatch does consistently deliver least optimal results when considering this metric. On the other hand, if we instead consider PCK@0.05, then DeciWatch is generally the model that delivers the greatest results and by quite a big margin, whereas either of the architectures based on a bidirectional convolutional LSTM delivers the worst results and the results of 3DConv are somewhere in between.

Thus, looking at the accuracies, if one is looking for the model that most consistently delivers great rough estimations of the placement of the keypoints, then either 3DConv 1.2 or either of the two architectures based on the bidirectional convolutional LSTM for any of the three experiments would be the optimal choice. On the other hand, if one is looking for the model that most consistently delivers the most accurate results, but do however occasionally yield some predictions that are pretty far off, then DeciWatch 1.1 or 1.2 would be the ideal choice. We have to however keep in mind, that 3DConv is much smaller than the remaining three architectures, so if one is also considering in the memory usage of the models, then 3DConv would be the optimal choice.

## 1.4 General Reflections

Generally, we believe that we made the correct choices throughout the execution of the project. However, looking back throughout the project, we do find some bad decision made by us, that could have been avoided or made in another way.

### 1.4.1 Pretraining

BRACE and Penn Action can be very semantic different from each other. For instance, in the BRACE dataset the person in the video sequence is very often upside down and performs some very quick movements, whereas the people in the Penn Action dataset is usually in some more usual poses an do not make movements that are as quick as the ones from BRACE. We tried to fix this by only including video sequences from the Penn Action dataset that had more poses where the person was not just standing straight up, however, even by doing so, the two datasets are still sematically different. Further, the video sequences from the two datasets are often filmed from two very different camera angles, resulting in the same keypoints being placed at different locations. If the two datasets are very different this will have an effect on the fitting of the models.

We also see some problems in the pretraining datasets individually. For the BRACE dataset, it is clearly documented, that the video sequences is filmed using 30 frames per second. For Penn Action, on the other hand, a similar information is not stated anywhere and the video sequences comes as individual frames without the duration of the video sequences noted, hence why cannot either compute the frame rate of these video sequences. This could cause some problems, as a two windows of frames, one from each dataset, could span two different durations and thus capture two different duration of context. Further, some of the keypoints of Penn Action was marked as being non-visible. For these keypoints there were no consistent procedure of how they were annotated, as the position of these were sometime estimated by a human and at other times there were just placed near one of the corners of the frame. To handle this, we simply went with the easiest solution of just completely removing these keypoint, leaving the corresponding heatmap empty, however, in hindsigt we do see how this could have a negative impact, as the groundtruth keypoint could suddently seem to "disappear" as the keypoint went from being visible to being non-vinsible. For the BRACE we also see an issue, as the dataset has not been fully manually annotated, but instead annotated using a pose estimator and only certain incorrectly predicted poses were manually annotated, where incorrectly predicted poses were detected by a machine learning model. However, if this machine learning model for detecting incorrectly predicted poses delivers suboptimal results, then the annotations would also be suboptimal and thus contain some noise, which is very difficult for our models to predict.

For the preprocessing of the pretraining dataset, the goal was to preprocess the data, such that it simulated the output of the already developed pose-estimator. This was done through multiple steps, where we used multiple values. For instance, we expanded the bounding-boxed created from the keypoints by $10\%$ in each direction, we used a standard deviation in the range $\{1, 1.5, 2, 2.5, 3\}$ for the Gaussian filter on the input data, and we shifted the input by sampling from a normal distribution with mean $\mu_{in} = 1$ and standard deviation $3$ or $6$, depending on the experiment. These values were all some values that we came up with. We could instead have picked these values in a more sophisticated manner such that the data would be more similar to the ClimbAlong data. This could for instance have been done by approximating them from the ClimbAlong dataset. We could further have made this standard deviation keypoint-dependent, such that it would have modelled whether how much movement each keypoint makes. However, we do not believe this to be much beneficial, as (1) we generally already receive some great results which are difficult to beat, and (2) for the experiments where we increased the added noise, we did not see an improvement on of the keypoints that carry a lot of movement, for instance the pinky, hence why moveement does not require a more shifted keypoint.

For the last part of the preprocessing of the pretraining dataset, we split the data into a training, validation and test set, consisting of 60%, 20% and 20% of the data respectively. This was done by making sure that none of the windows were repeated and that none of the windows were overlapping between the three datasets. By doing so we ensured that the evaluation did carried minimal bias, as the models had not seen any of the frames in the validation and test set during its training. However, different frames of the same video sequence could appear across the three sets, which could maybe carry some bias, as the same person would then appear in the three sets. However, we do not believe this to happen in our case, as the pretraining dataset does not include the actual frames but instead only include the annotations.

### 1.4.2 Finetuning

When creating the target heatmaps for the finetuning data, we occasionally met a problem, where the groundtruth keypoints was placed outside of the predicted bounding-box, which was used as target bounding-box. As the goal of our models is not to correct the predicted bounding-box, but instead just to correct the position of the predicted keypoints, we had to make a choice of what to do for keypoints that were supposed to be placed outside of the bounding-box. We generally saw two choice, as we could either completely remove the keypoint and just leave the heatmap empty, or we could move the keypoint such that it was somewhere inside the bounding-box. As we expected the first solution to have a negative impact, as the models would just learn that the keypoint was missing, we simply went with the second solution.

## 1.5 Future Work

If we were to work further with this project, we first find it interesting to experiment with other machine learning methods. We would for instance find it very interesting to test the effects of letting DeciWatch process all frames, instead of only processing every fifth frame. Further, we find it interesting whether or not it would improve the results if DeciWatch was adapted, such that it made use of Vision Transformers as introduced by Dosovitskiy *Et al.* [1].

Further, we see some potential work in trying to avoid the overfitting during the finetuning-stage that we experienced in section **??** and find it interesting how much this would improve the final results. This could for instance be done by (1) incorporating data augmentation to increase the size and variation of the dataset, for instance by rotation the data or adding some noise, (2) decrease the complexity of the models, such that they have less tuneable parameters, or (3) test various regularization techniques such as weight decay or dropout.

Lastly, we suggest that the models could be retrained multiple times. In section 1.1 we blaimed some of the results on the random initialization of the weights of the models. By retraining the models multiple times, the likelihood of this randomness being the reasoning for these results decreases and thus we will be sure whether or not our results are due to the randomness of the weights.