Enhancing Home Physiotherapy: Machine Learning for Exercise Recognition and Error Detection

BAPTISTE MAQUIGNAZ, RAMI ATASSI, GARIK SAHAKYAN
EPFL, Switzerland

I. INTRODUCTION

Our project builds upon David Rode's research in Markerless Motion Capturing and Pose Estimation for therapeutic applications. Despite Markerless Pose Estimation exhibiting lower accuracy compared to Marker-Based methods, it presents a cost-effective and less cumbersome alternative, requiring no specific equipment or setup. David Rode's objective is to enhance home physiotherapy by enabling individuals to record their physiotherapy exercises at home and gain valuable insights into potential mistakes via an application. Ideally, this process would facilitate the automation of clinical assessments. Due to the limited quantity of data, they recorded individuals performing physiotherapy exercises and utilized Google's 'Blazepose' neural network to generate a new dataset depicting the evolution of body joint positions over time. Our project focuses on three key objectives: identifying exercises based solely on Blazepose joint positions, categorizing errors, such as improper form maintenance, and investigating how factors such as gender, weight, height or mobility influence accuracy.

II. THE DATASET

The study utilized four cameras, each capturing footage at a rate of 30 frames per second. These cameras included two frontal and two side cameras, positioned at varying heights, forming the basis for training Blazepose. The dataset predominantly consists of joint positions in an orthogonal spatial basis. There are 33 distinct joints, each providing estimated coordinates along the x, y and z axes. Additionally, the dataset includes temporal information for each video and specifies the camera used for recording as well as the participant that is performing the exercise. Each frame was manually labelled to identify the physiotherapy exercise and potential mistakes in the execution. The dataset includes seven unique exercises and seven distinct 'sets,' representing the two primary types of mistakes in execution, if they are present. Altogether, the dataset comprises 2.2 million joint positions evolving over time and space for various exercises, errors, (anonymized) individuals, and camera perspectives.

III. PREPROCESSING

Identification of the participant

It is important to highlight that even after anonymization, unique identification may still provide valuable insights into an individual's size or way of performing exercises. In cases where multiple videos feature the same person, the data may exhibit correlations that the model could learn from, potentially enhancing performance. However, the assumption that a unique identification will be provided depends on the final product's implementation. We therefore chose to discard participant's identity when training models. Note that this choice was also made in an attempt to minimize the amount of information needed about the patient.

Specifications of camera

The same rationale applies to camera specifications, since people will typically use their smartphone or webcam to record themselves. Consequently, considering the camera's angle and height was deemed unrealistic for incorporation into our approach. However, we learned that the application would typically give detailed instructions to users on how to position themselves. That information could then hopefully be used to improve performance in the final product.

Frame trimming

In order to calibrate, the videos start with each participant posing in a reference pose. They then clap in their hands, perform 5 repetitions of the exercise and go back to the reference pose. Calibration serves the purpose of enhancing performance in Blazepose joint tracking. Nevertheless, our focus lies on analyzing exercise repetitions. Consequently, there is a logical inclination to trim frames at both the beginning and end of the sequences. Based on video footage, it takes roughly 3 seconds at the start and 1 second at the end which corresponds to the 90 first and 30 last frame of the video. To ensure that we do not erase useful information, we erased the first 60 and last 30 frames. This implementation allows a small improvement in performance, leading to a 0.5% test accuracy increase when predicting exercise.

Handling corrupt data

In certain instances, Blazepose appears to encounter difficulty to accurately track the joint positions. In these cases, the entire skeleton becomes corrupted, resulting in the unavailability of joint positions. These occurrences transpire in approximately 1.6% of frames, mainly manifesting as isolated instances of corruption or small continuous sequence of corrupted frames. For the 'small' corrupted blocks, we

assume that movement can be approximated as linear. A logical approach is then to interpolate the missing joint positions based on the preceding and succeeding frames. It is essential to acknowledge that malfunctions in Blazepose may manifest when inferring positions becomes challenging, resulting in significant errors in adjacent frames. This, in turn, can lead to nonsensical outcomes in linear interpolation. However, this seems to be the best approximation to fill the missing data. For consequent lengths corruptions, we simply erase the whole Nan block. This decision is grounded in the understanding that interpolating several seconds of a sequence solely based on the previous and last frame is not likely to give good approximation. The critical length that we found by cross validation is 20 frames, namely if a continuous sequence of more than 20 frames is corrupted, we erase it from the dataset, otherwise we interpolate the missing frames linearily.

Data augmentation

Markerless position estimation is a very noisy science. Even with a good set up, camera and athlete performing the exercises, the skeleton sometimes shows a significant difference with the theoretical position. One of the main assumption for this project is that people should be able to film themselves at home with a very cheap set up and still get significant insights on the performance of their exercises. It is, therefore, a natural requirement to desire a noise-resistant model, as various factors contribute to noise. We chose to add Gaussian noise to the joint positions to mitigate the sensitivity of our models to noise. After experimenting with different variances, we found that a variance of 0.01 improves noise resistance without sacrificing the relevance of positions. Another intuitive idea would be to implement linear transformation such as simulating camera zoom or shifts to allow robustness to people with a poorly centered or cropped video footage. However, these situations are already addressed during standardization: for spatial positions, standardization can be viewed as re-centering the subject and scaling appropriately.

Standardization

The given dataset was already standardized so that the positions of the joint stand between -1 and 1 for each sequence and the centre of mass around the pelvis. This normalization results in very small column means and standard deviation. It functions similarly to 'classical' standardization, which would zero the mean and scales according to the standard deviation. By curiosity, we tried performing a 'classical ML standardization', but noticed an overall lower performance when doing so. For example, the frame by frame error rate on guessing which exercise was performed, decreased by 0.5%, when implementing further standardization. Consequently, we chose to simply keep the initial standardization.

IV. Modeling and training

Train / Test splitting

The dataset is composed of positions evolving in time, which means that successive frames are heavily correlated. In addition, the same footage was recorded from different cameras, implying further correlation between sequences. Not addressing these dependencies when splitting the train and test set would lead to biased performances. For example if we test our model on frames adjacent to the training frames, our accuracy would be falsely high. It would also be rigged for a sequence tested from a camera angle, when it was trained on another angle. This implies that the test set cannot be selected from dataset at random. To address this issue, we selected participants to be exclusively in the test set or in the train set. To split the train and test set, we chose 5 participants among the 25 (20%) to put in the test set and kept the rest for training. We then used data augmentation by adding noisy versions of sequences to balance the number of sequences for each exercise and set in the training set since it was heavily unbalanced. This simulates the fact that the application will encounter people that the network hasn't seen before. To illustrate the performance bias, we achieve a $80+\%$ accuracy using a Random Forest decision tree, by allowing the random split of camera sequences, compared to roughly 50% when properly separated.

Choosing architecture

In order to predict the physiotherapy exercise and its' correctness (labeled as 'set'), two distinct methodologies were identified. We could concurrently predict the combination of exercise and set using a unified model, or else train two distinct models to independently predict the exercise then the set, respectively. We initially trained two simple models composed of a single 3 linear layers of size 256 predicting both the exercise and set (size 38) and another composed of two similar MLPs of output size 7 since there were 7 different exercises and 7 different sets. It is noteworthy that the sets are labeled either as 'Correct' or with a letter from 'A' to 'F,' corresponding to specific types of errors based on the executed exercise. We noticed that the second method performed better with a great test accuracy on the exercise [V] and a slightly better performance on the prediction of the set than the first model. Because of the huge difficulty gap between the two task, the slight performance difference and the possibility of feeding the exercise prediction of the first MLP as additional input to the second MLP when training both separately, we favored implementing two separate models.

Models

Because we were unfamiliar with this type of dataset, we decided to run several models to compare performance and find the most fitting one.

- The first model that we implemented was the one discussed above: we used a MLP with 3 hidden linear layers of size 256 and ReLu non-linearity. This standard model is known as MLP 3x256. We played with drop outs, batch norm and augmented data to compare performance. We also tried running a MLP 5 x256 and 3x512 to experiment different depth and number of neurons.
- Dealing with heavily space-correlated data seemed a bit similar to an image, we naturally thought about using a CNN, we trained several model to improve feature selection before feeding the data to a MLP. We will call the first one CNN2B because it contains two blocks, each composed of increasing size convolutional layers (64 and 128) that are separated by 2D pooling layers. This simple model allowed us to tune the block size It is then fed to a MLP 3x256. This architecture was inspired from VGG19 and other famous CNNs. We will call the second model CNN3B, it is an up-scaled version with an additional block with convolutional layer of size 256.
- After doing some research [1], [2], we found out that a fitting model would be recurrent neural networks, and more precisely a Long Short-Term Memory (LSTM) model. LSTM has the particularity to remember previous inputs for a long period of time, making it suitable for our given dataset consisting of joint positions evolving over time. We used one LSTM blocks consisting of 256 units each, and having tanh activation function and sigmoid recurrent activation function. This was followed by a dense layer of 7 units, for performing the classification task using categorical cross-entropy loss.
- In parallel of the previous models, we have also tried some of the classics from ML, namely Logistic Regression, SVM and decision trees. Since they didn't seem appropriate for our data, we wanted to quickly verify this assumption. While logistic regression and SVM were indeed not relevant, decision trees such as Random Forest and Boosting Gradient were worth further investigation. Both came close to MLP accuracy and Random Forest (RF) in particular appeared robust to the changes in data preprocessing. Since it was easy and fast to train, we started to frequently use Random Forest to quickly evaluate different preprocessing features or to verify the correctness of our Neural Networks models.

Handling non uniform size of input

An additional difficulty arose from the variance among inputs, which ranged from 345 to 2830 frames depending on the video length. We needed to adapt the data to each model :

- When evaluating MLP, we predict a label for each frame and then take the most recurrent one as prediction for the entire video.
- When using CNN, we start by appending a number of empty frames to assure that the sequence length is a multiple of the block size and then feed samples of frames as a fixed length 2D-array. We then take the most recurrent prediction to predict the video label.
- For both RNN and RF, we needed videos to be of equal lengths. We thus decided to truncate all videos to the size of the smallest video in both train and test set.

V. Results

Accuracy on exercise

It seems that predicting the performed exercise is a rather easy task that can be performed at high accuracy with an MLP 3x256 model. We chose an Adam optimizer and trained the model for a small number of epochs with step size of order 10^{-3}. We achieve a test accuracy of 100% without the use of dropouts or batch norm layers. The network is able to predict the correct label for 96.06% of the individual frames on participants that it never encountered before. Note that achieving perfect accuracy required fine tuning of parameters but the model can achieve $99.5 + \%$ with a single epoch.

Accuracy on set

With minimal preprocessing and no data augmentation, a naive run of a MLP 3x256 guesses both exercise and set at the same time with a test accuracy of 54.28% and an F1-score of 0.51. Taking this result as a basis, we will discuss different models in order to increase accuracy and F1-score.

When training MLPs, we chose Adam as optimizer, ReLu as activation function which seemed to be classical choices for MLP. We use a big batch size of 128 since the size of dataset was consequent and usually initialized step size with values of order 10^{-3} with a scheduler that reduces the step size as error gets smaller. It usually took about $30 - 50$ epochs to converge without dropouts. We found out that the data augmentation with noise improved accuracy [I]. While the performance did not exhibit a significant improvement with the exclusive use of dropouts, it noticeably increased the difficulty of achieving convergence to a low loss, and we did not succeed in fully training the network even for small drop out percentage. This could be due to the fact that the model is small relative to dataset [3]. However additionally adding batch norm layers facilitated convergence and seemed to improve performance. We then tried to tune the number of neurons and depth of the network to see the impact on performance and tried to run both a MLP 3x512 and a MLP 5x256 with augmented data. Note that most MLPs were stopped before fully converging to limit train set overfitting.
In order to train CNNs we kept Adam as an optimizer and

ReLu as activation function. We typically needed to use a step size of order 10^{-3} and a similar scheduler but a much smaller batch size of 32 since the amount of data is much smaller. We tested several CNN models that all had lower performance than MLP alone. An interesting observation was that adding depth and blocks to CNN2B seemed to worsen its performance. This seems to imply that the feature selection performed by convolution are not well fitted for our application or at least that we did not manage to find a good CNN model.

When training the RNN/LSTM model described above, we again used Adam as optimizer. We initially started with a learning rate of 10^{-2} and decreased it to 10^{-4}, which was the one that gave us the fastest convergence. By testing different drop out rates, the best results were obtained without any drop outs at all. The model converged after 250 epochs. Interestingly, we notice that this model does not give good results ($48\% \pm 3$ of accuracy) compared to previously discussed models. We still think that it was worth mentioning, since RNNs are often used for time series forecasting and classification in practice.

For Random Forest, both models with normal and noisy data were cross-validated to find their optimum parameters namely the number of estimator (decision tree) and max-depth of each tree. It was important to keep these parameter low to not overfit the train set.

Based on all the previous observations, we designed a custom MLP that we will call PhysioMLP. It is composed of three hidden layers of size 512 that uses drop outs with drop rate 0.3 and batch norm layers in between layers, (except the last one). The set up parameters is very similar to the other MLPs. PhysioMLP achieves 59.05% accuracy and 0.59 weighted average F1-score.

Model	Dropout	Data aug	Accuracy (%)
PhysioMLP with batch norm	Yes	Yes	59.05
MLP 3x256	Yes	Yes	56.29
	Yes	No	57.14
	No	Yes	58.04
	No	No	56.19
MLP 3x512	No	Yes	58.57
MLP 5x256	No	Yes	54.18
CNN2B	32	No	50.48
	64	No	50.91
	128	No	54.05
	256	Yes	56.01
CNN3B	128	Yes	48.52
RNN	No	Yes	48.10
Random Forest	No	No	52.57
	No	Yes	49.42

Table I
ACCURACY OF DIFFERENT MODELS ON ERROR SETS

VI. GOING FURTHER

Throughout the whole process and until the deadline, we continuously thought about new ways of improving our project, about testing new models, and about exploring other data selection and preprocessing methods. While most of them were time-consuming, and didn't yield the expected results, this iterative process let us with many ideas and not enough time to implement them. To enumerate a few ideas that would be worth exploring: we thought about augmenting the dataset by artificially simulating the joint positions that would have been recorded for small rotations of the subject in space (or equivalently small rotations of the cameras around the subject). This idea came from the intuition that people at home would probably have less precise set ups and feeding noisy rotations would likely improve the model's resistance to such imperfections. Another idea would have been to test Temporal Convolutional Network (TCN) and Hidden Markov Models (HMM) which seem to be efficient on time-dependent data. We are deeply convinced that the models used in our project could have been greatly improved with more knowledge and intuition in machine learning but we think that they constitute a good basis towards automated at-home physiotherapy diagnosis.

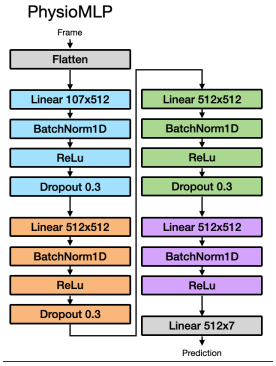

Figure 1. Architecture of PhysioMLP

VII. CONCLUSION

In conclusion, our project journey has been both rewarding and enlightening and has greatly deepened our understanding, allowing us to apply theoretical knowledge in a practical context. This project provided hands-on experience with different type of machine learning techniques, contributing significantly to our skill development. The considerable freedom allowed us to cultivate independence. Despite dedicating the majority of our time to experimenting with various models and figuring out what would or wouldn't work, and acknowledging that the accuracy of our models is still significantly improvable, we successfully applied a substantial portion of the concepts we learned this semester.

ETHICAL RISKS

A bad automated diagnosis by the model when spotting mistakes could lead to a bad execution of exercises or even injuries in the worst case. It is therefore important to create a flexible model that maintains good performance regardless of the person's gender, height, weight or way of moving. An important difference between our dataset and the real life application is the fact that the participants filmed in the dataset are unimpaired and in good health. This might typically not be the case for physiotherapy patient since they might be recovering from an injury, having health disorders or low mobility, contrasting with the dataset. There are many points to discuss on that subject but we will focus on the time of execution that could be increased dramatically for mobility restricted individuals. With bad resistance to this risk, the performance could suffer greatly and give bad diagnosis which would lead to a bad execution of the prescribed exercises. This is a significant risk since it seems quite likely to happen if the model is too sensitive to execution speed and could potentially cause physical harm if the bad execution is maintained for a long period.

To simulate sensitivity to execution speed, we trained a model on fast executions using the MLP 3x256 described above. We separated the test set in two parts, one containing fast executions (videos consisting of 1000 frames or less), and the other one containing slow executions (videos consisting of more than 1000 frames). Note, that counting the number of frames is a good measure of execution speed since each video consists of 5 repetitions of the same exercise. We tested the model on both slow and fast test sets, and obtained that slow exercise executions had 41.05% accuracy, while fast exercise executions had 60.87% of accuracy.

Unfortunately, we could not consider that risk in our model since it seemed heavily linked to the dataset, and it was impossible to record fresh footage. However, it is likely that the final application will also be trained on real users and that the basis dataset will grow as people use the application.

REFERENCES

[1] J. Brownlee. (Accessed 2023) Lstms for human activity recognition time series classification. [Online]. Available: https://machinelearningmastery.com/how-to-develop-rnn-models-for-human-activity-recognition-time-series-classification/

[2] S. ZAITSEFFe. (Accessed 2023) Classification of time series with lstm rnn. [Online]. Available: https://www.kaggle.com/code/szaitseff/classification-of-time-series-with-lstm-rnn

[3] shimao. (2017) Dropout makes performance worse. Accessed: December 10, 2023. [Online]. Available: https://stats.stackexchange.com/questions/299292/dropout-makes-performance-worse