Assessing Squat Form Using 3D Pose Data

Rachel Adenekan
Department of Mechanical Engineering
Stanford University
adenekan@stanford.edu

Blake Pagon
Department of Computer Science
Stanford University
bpagon@stanford.edu

Marissa Lee
Department of Mechanical Engineering
Stanford University
mrlee1@stanford.edu

Abstract

Strength training exercises are essential for human health. However, when performed incorrectly, humans are at risk of injuring themselves. As such, they often invest in costly personal trainers and group fitness classes in order to have access to a fitness expert who will guide the individual in performing exercises safely. We aimed to increase access to affordable trainers by creating a tool that uses deep learning in order to classify squats as 'correct' or as one of six common types of 'incorrect.' Our key contribution was developing and comparing the performance of a bidirectional LSTM to that of a 1D CNN, using 3D pose data from 2D video. The 1D CNN performed better overall, with fewer false positives and a much faster training time. Future work can assess the performance of 1D CNNs with other activity assessments and with real-world data.

1 Introduction

Strength training exercises are essential for human health and rehabilitation. When performed incorrectly, these exercises can lead to injury. To avoid such injury, many individuals invest in costly personal trainers and group fitness classes to access expert guidance. These options are inaccessible to individuals with low income and are particularly difficult to use today, in the midst of a pandemic.

We aim to increase access to affordable training and rehabilitation by developing a tool that uses deep learning to classify exercise performance. This tool takes 2D video as input and can be deployed to personal devices, such as computers or phones, for quick activity assessment. A great deal of prior research has been conducted in the areas of activity recognition and activity assessment, particularly with mobile sensors including wearables and video.

In the video realm, researchers have developed models using both raw video and body keypoints from pose estimation as inputs [1,2]. Model architectures have included 3D convolutional neural networks (CNNs), 2D CNNs with recurrent neural networks (RNNs), and 2D CNNs with optical flow [2]. Some have been as simple as using a shallow network to learn from keypoints [4,5]. In a recent article, Ogata et al. developed a novel technique to assess exercise, using keypoint distance matrices. Applied to squats, they achieved 74% accuracy on their test set [1].

Using the model developed by Ogata et al. as a baseline, this work aims to contribute a comparison of two models. Specifically, we will use 3D pose data from 2D squat videos to identify whether squats are performed properly or improperly, comparing performance of a 1D CNN to that of an RNN. We will first attempt to regularize the 1D CNN model developed by Ogata et al; next develop a long short-term memory (LSTM) network to perform the same task; and lastly compare the regularized CNN model performance to that of the LSTM.

Ultimately, this work describes how CNNs and LSTMs perform comparably on this problem with respect to gross outcome metrics. However, the CNN is much faster to train and results in fewer false positives, so it is recommended for future approaches.

This work will contribute to a better understanding of prioritization of different model architectures in human activity assessment. Methods developed in this project can then be applied to activities beyond squats, improving safe exercise and affordable access to exercise.

2 Related Work

There are many ways in which deep learning can be leveraged for video analysis. Often, videos are assessed using 2D or 3D convolutional neural networks (CNNs). 2D CNNs capture spatial information and can be combined with temporal layers or temporal information, such as optical flow, to analyze videos. 3D CNNs capture both spatial and temporal information in a single layer and can also be combined with 2D layers and other temporal information [2].

Much of the recent research in activity analysis has used simpler approaches in deep learning to simplify model training. A major component in most of these approaches is pose estimation. Pose estimation involves identifying keypoints on humans and reducing the data to information just about those keypoints. This method removes confounding video features like backgrounds and reduces the dimensionality of input data. A timestep represented by a 256x256-pixel video, for example, can be represented as a 17x3 vector or a flattened 51x1 vector containing 3D coordinates for 17 keypoints. Pose estimation is often itself conducted by feeding video through a deep network, as in the case of the model developed by Kanazawa et al. [3]. Amazingly, models like this can estimate 3D information from 2D videos.

Pose estimates and similar data have been used effectively in shallow networks by extracting summary features, such as mean, variance, and kurtosis over time. Kianifar et al. achieved 73% accuracy when classifying single leg squats as good, moderate, or poor with a decision tree classifier [4]. A disadvantage of this approach is the loss of temporal information that occurs with summary features extraction. Yet shallow networks can be fed a form of temporal information, through optical flow and pixel gradients. Pirsiavah et al., in predicting expert judge scores on Olympic diving videos, achieved a rank correlation of 0.41 using a support vector regression model with temporal features [5].

Deep models that take raw video as input have many parameters and are difficult to train. Shallow models that take summary features over time as input may lose valuable sequence information. In the middle lie deep models that take in reduced inputs that maintain temporal information. In 2019, Ogata et al. used the pose estimator developed by Kanazawa et al. to classify squats for a single individual. After extracting keypoint information, Ogata et al. created distance matrices, which represent the Euclidean distances between every pair of keypoints. Using these distance matrices, Ogata et al. were able to classify squats as having good form or as reflecting one of six classes of bad form, using a 1D CNN based on ResNet. They achieved 74% accuracy [1].

Ogata et al. used relatively small kernels in their model, which can make training easier but may also fail to capture longer-term dependencies that may exist in these videos [1]. An alternative to a 1D CNN when analyzing 1D temporal data is a recurrent neural network (RNN). A commonly-used RNN is a long short-term memory (LSTM) network, which can be useful in capturing long-term dependencies [6]. LSTMs have recently been used to successfully classify normal and pathological gait [7,8], stroke rehabilitation exercise quality [9], and daily activities [10]. Input features to these networks include kinematic time series extracted from RGB depth sensors [7,9], force plate time series from in-lab equipment [8], and accelerometer and gyroscope time series from wearables [10].

LSTMs are promising in classification tasks with time series inputs. In this project, we use the distance matrices published by Ogata et al. [1] and explore the effect of using an LSTM-based model in place of their 1D CNN-based model to assess squat form.

3 Data

We use a dataset of 2001 videos curated by Ogata et al [1]. The videos were captured using a static camera reference, and they feature the same individual performing squats for 10 seconds, equivalent to 300 frames, per video. This typically amounts to between 3 and 5 squats. These videos have a range of different backgrounds. Ogata et al. extracted 3D pose data from the motions in the video, using the pose estimation algorithm developed by Kanazawa et al. [3]. The 3D pose data consists of (x, y, z) coordinates for 19 different joints. This 3D pose data is then further processed by calculating the euclidean distance of each point from all other points. This leads to 171 different features for each frame. Therefore, a video example has shape (300, 171) corresponding to 300 frames and 171 features for each frame. We use these pose data and ground truth labels that are included with the pose data to train our model. The labels fall into the categories of "Inward Knees" ($n = 230$), "Round Back" ($n = 280$), "Warped Back" ($n = 312$), "Upwards Head" ($n = 272$), "Shallowness" ($n = 319$), "Frontal Knee" ($n = 295$), and "Good Squat" ($n = 293$) [1]. Figure 1 depicts two frames of a single video and keypoint detection.

Figure 1: (left and middle) Example frames from a 2D squat video. (right) Example of 3D pose data overlayed across video. (Courtesy of [1].)

4 Approach

Using an 80/10/10 split from our dataset, We 1) improved the 1D CNN published by Ogata et al. by simplifying the architecture [1], 2) trained a bidirectional LSTM on the same task (assessing squats with time series pose data), and 3) compared the two aforementioned models to assess the relative performances of the models. All models were implemented in TensorFlow [11].

4.1 CNN Model Development

The baseline model for these experiments is that developed and described by Ogata et al. [1]. When applied to the published dataset, this model achieves good accuracy on the seven-class problem but greatly overfits, with train and validation accuracies of 1.00 and 0.69, respectively. We reduce overfitting in this base model by experimenting with the removal of layers and with the introduction of dropout into the various layers of the network. A final 1D CNN model is selected based on validation set accuracy.

4.2 LSTM Model Development

The model which we constructed to compare against the model proposed by Ogata et al. [1] is a bidirectional LSTM model. We chose a two-layer bidirectional LSTM model as previous work in gait classification shows success with this kind of network [7]. Our model works by taking the input and first feeding it through a bidirectional LSTM layer. The outputs of the forward and backward LSTMs are then concatenated before being fed into another bidirectional LSTM layer. The final output produced by the second bidirectional layer is passed through a dense layer with a softmax activation. A pictoral representation of the model is shown in Figure 2.

We train our model with the Adam optimizer and the categorical cross-entropy loss function. We experiment with learning rates, batch sizes, and hidden units in the LSTM layers via single-parameter sweeps and then reduce overfitting via experiments with regularization and dropout.

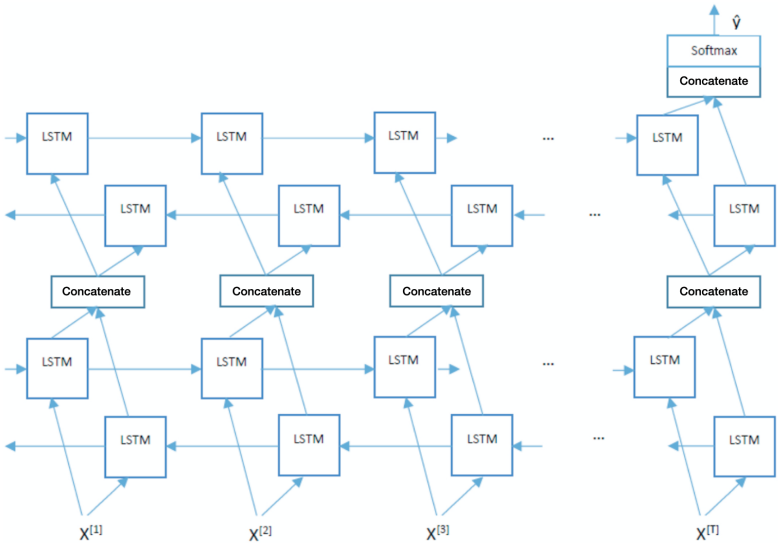


Figure 2: Our bidirectional LSTM model architecture. This model and figure are adapted from Khokhlova et al. [7] with the exception that we ultimately don't use a dropout layer for regularization.

4.3 Model Comparison

A main contribution of this project is the comparison of the performances of the two aforementioned models in classifying squats. We compare the performances of these models by examining training times, confusion matrices, receiver operator characteristics, and model performance in a simplified problem. The simplified problem involves combining two classes to attempt to improve the accuracy of predictions of "good" squats.

Training time is an important consideration if a deployed model will continue to update itself, and it provides insight into the feasibility of using models for future activity assessments. Confusion matrices provide information about model performance on an individual class basis, and they can provide insight into which classes are performing well and which are often-confused. Receiver operator characteristics display for us information on how model performance might change with different thresholds. Lastly, by simplifying the problem, we can gain insight into where our model is failing and how we might improve our model.

5 Experiments

5.1 CNN Model Development

The initial 1D CNN, described by Ogata et al., includes one convolutional layer with batch normalization and ReLu activation, four residual layers, an adaptive average pooling layer, and a dense classifying layer. Each residual layer contains three convolutional layers and a skip connection. See full implementation details in the manuscript by Ogata et al. [1]. We implemented model fitting with early stopping. Early stopping attempted to maximize validation set accuracy with a patience of 5 epochs.

Given the overfitting of this model, we first simplified the model architecture by reducing the number of residual layers. Two residual layers achieved less overfitting than four. Adding a dropout to the final dense layer with a dropout probability of 0.1 further reduced overfitting. Finally, adding dropout to the convolutional layers in the residual layers further improved the overfitting situation. No loss in validation set accuracy was observed. Results from these experiments are included in Table 1. The final simplified 1D CNN model, with 2 residual layers and dropout, achieves a test set accuracy of 72% (on par with the accuracy achieved by Ogata et al.).

	Baseline	Residual2	Dropout
Train Set Accuracy	1.000	0.884	0.760
Validation Set Accuracy	0.715	0.705	0.768
Epochs	27	20	36
Training Time (s)	93.3	48.3	86.9

Table 1: Results of CNN model simplifcation experiments. Baseline results are those from a recreated Ogata et al. model [1]. Results from Residual2 are those after decreasing the number of residual layers from 4 to 2. Results from Dropout are those after including dropout in the residual and dense layers.

5.2 LSTM Model Development

With our model design in place, we had to tune our hyperparameters. The variables with which we chose to experiment were the learning rate, the batch size of training examples, and the number of hidden units in the LSTM (we use the same number for both forward and backward LSTMs). Clearly, there are too many combinations of hyperparameters to try a wide range of combinations, especially since training the LSTM model can take as long as an hour with a batch size of 16, and around 30 minutes with batch sizes of 64 and 128 depending on the number of hidden units in the LSTM.

A method as straightforward as a grid search wouldn't work because even trying all possible combinations of three learning rates, three batch sizes, and three hidden units values would result in 27 different models to train, and it would require around a day straight of computational work to train all of these models. To avoid this combinatorial explosion, a different approach was taken. We defined a base hyperparameter configuration and individually tuned values, one at a time, within that configuration. The yielded only 9 different models to train and compare.

We trained our model with the Adam optimizer and the categorical cross-entropy loss function, implementing early stopping with a patience of 50 epochs to avoid overfitting. The base hyperparameter configuration had a learning rate of 0.001, a batch size of 16, and 8 LSTM hidden units. We tested varying the learning rate to 0.01 and 0.0001; varying the batch size to 64 and 128; and varying the number of hidden units to 128 and 256. From this initial exploration, a couple of trends became clear. Firstly, a larger batch size significantly sped up model training and did not have much of an effect on accuracy. Secondly, a higher number of LSTM hidden units generally gave better results. See Figure 3 for an example of some results obtained during this initial phase of tuning.

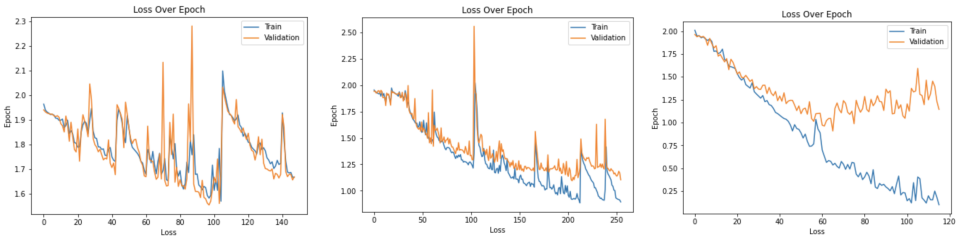


Figure 3: Examples from the initial sweep of hyperparameters. The leftmost figure is with a learning rate of 0.0001, a batch size of 16, and 8 LSTM hidden units. The middle figure is the same but with a learning rate of 0.001, and the rightmost figure is the same but with 256 LSTM hidden units.

From this, we tuned the learning rate to that which worked well with a larger number of LSTM hidden units and a larger batch size. After this step, we had a good configuration for the learning rate, the number of LSTM hidden units, the batch size, and the number of epochs to train. We then tried to regularize our model, as the loss and accuracy curves clearly showed overfitting during training. While regularizing, we also attempted to find a good learning rate schedule, as we noticed the loss curve tended to spike up and down after initially decreasing. We tried L2 regularization in the LSTM kernel weights as well as dropout layers and found that L2 regularization worked better. We also found that a constant learning rate of 0.0001 worked better than any learning rate schedule. Refer to Figure 4 for an example visual comparison of the loss curves.

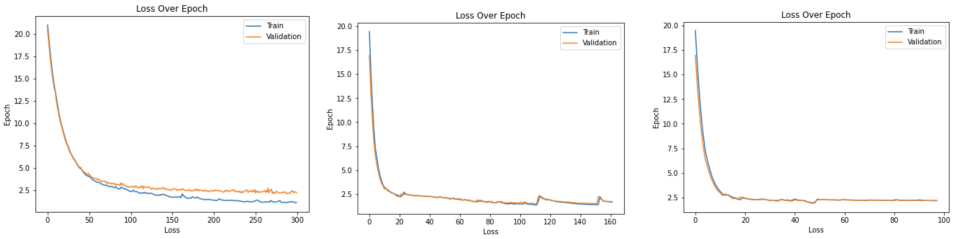


Figure 4: Examples from final tuning of the learning rate with L2 regularization. The leftmost figure maintains a constant learning rate of 0.0001, the middle figure starts with a learning rate of 0.0005 and decays with a base of 0.8 every 10 steps, and the righmost figure starts with a learning rate of 0.0005 and decays with a base of 0.9 every 10 steps.

Our final model uses a constant learning rate of 0.0001, 256 hidden units in each LSTM layer, and a batch size of 128. It achieves a test set accuracy of 74.0%.

5.3 Model Comparison

From this work, it is clear that both CNN-based and LSTM-based architectures learn to predict squat classifications from temporal distance matrices. In this section, we compare the performances of these models by examining training times, confusion matrices, receiver operator characteristics, and model performance in a combined-class problem.

5.3.1 Training Time

The final 1D CNN model took about 86.9 seconds to train with an NVIDIA Tesla K80 GPU over 36 epochs. The final LSTM model took considerably longer, requiring nearly 30 minutes to train with the same GPU over 247 epochs with an increased batch size. Time discrepancies are owed to parallel optimization performed with CNNs on NVIDIA GPUs.

5.3.2 Confusion Matrices

To better visualize our model's performance, we constructed confusion matrices (Figure 5). The CNN and LSTM models have similar gross performance. However, the CNN performs better in some metrics of particular interest in this task. For example, from the confusion matrices, we can observe that the CNN has fewer false positives for the 'good squat' class (Figure 5). This is important so that we do not ingrain a squatting model in an individual that will result to poor, injury-prone form.

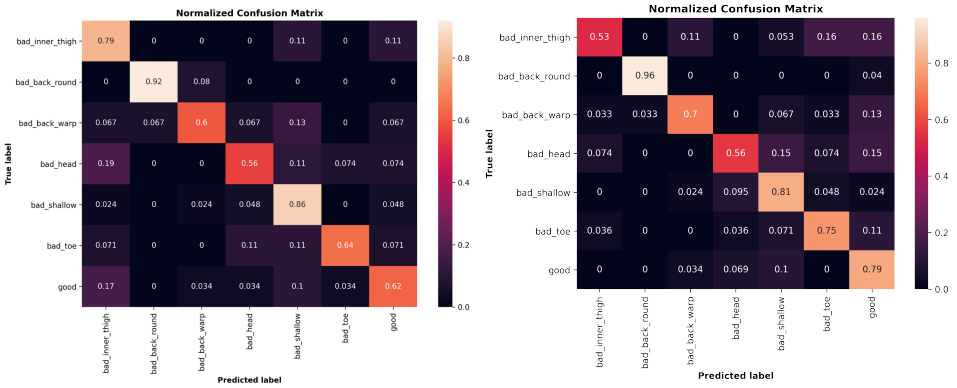


Figure 5: Normalized Confusion Matrices for the (left) 1D CNN and (right) LSTM.

While both models could benefit from tuning of certain classes, we note that both models are quite good at predicting the 'rounded back' and 'shallowness' cases. These are useful in giving feedback that could help avoid back injury and improve strengthening results. In both the CNN and the LSTM, we notice that the 'upwards head' and 'shallowness' classes are often confused for one another. This may be due to the two types of errors appearing similar in nature in the temporal distance matrix data. We explore further error analysis regarding these two classes in section 5.3.4 (Class Combinations).

5.3.3 Receiver Operator Characteristics

To dig a deeper into model results, we plot the receiver operator characteristics (ROCs) for the "good" squat class (Figure 6). An ROC is a plot of true positive rate against false positive rate over a range of thresholds at which to consider a prediction true. Ideally, true positive rate is 100%, and false positive rate is 0%, always. This would yield an area under the ROC curve of 1. The larger the area, the generally better the model.

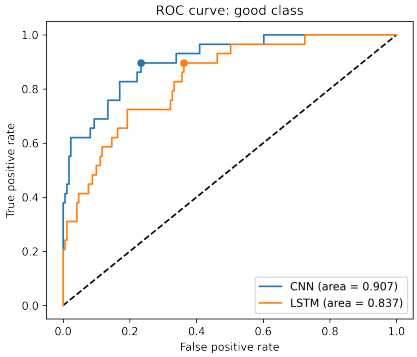
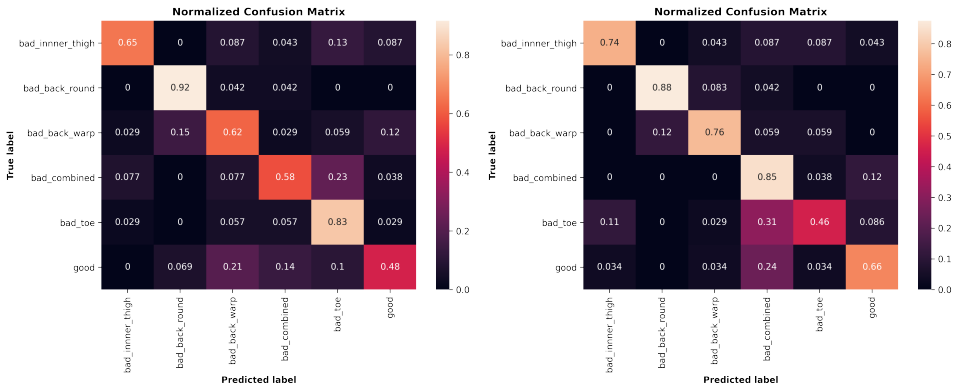


Figure 6: Receiver operator characteristics with operating points selected by Youden's index for the CNN and LSTM models. The dashed line represents results from a theoretical chance-based model.

Using area under the ROC curve as a metric, the CNN outperforms the LSTM. We can also look at the operating point on the ROCs that maximizes Youden's index. Youden's index optimizes true negatives and true positives. The selected operating points of the models each yield sensitivities of 0.897, but the operating point of the CNN outperforms that of the LSTM, with specificities of 0.766 and 0.637, respectively. Sensitivity is the proportion of positives that are identified as true positives , while specificity is the proportion of negatives that are identified as true negatives. In squat classification, it is important that we do not mislead our users into believing they are squatting properly when they are actually squatting improperly. It is better to accidentally tell someone they need to improve a good squat than to tell them they are performing a bad squat correctly. Consequently, we want to minimize false positives in classifying squats as good. A low false positive rate, as we can see in Figure 6, means a tradeoff with a high true positive rate. We can see that no matter the true positive rate of a model, the CNN will yield fewer false positives.

5.3.4 Class Combinations

In order to improve our the performance of both the CNN and LSTM, we combined two classes that were most often confused for one another in the original 7-class classification task. Specifically, we combined the 'upwards head' and 'shallowness' classes to obtain a 'bad combined' class. To maintain approximate balance in the dataset, we took just half of the examples from each of the original two classes when constructing the combined class. We then re-ran both models and observed that, on the whole, combining classes did not significantly improve either model's performance. Sensitivity of the classification of "good" squats decreased both models. Gross accuracies of the models decreased from the 7-class to 6-class tasks, with the CNN dropping from 71.5% to 67.8% and the LSTM dropping from 74.0% to 70.8%. These diminished results of the 6-class problem suggest that the models' errors in the 7-task problem are not due to certain classes being too similar to one another. Instead, perhaps the models would be further improved by using real-world, more varied data. A larger and more realistic dataset would enable our model to learn a more diverse set of errors and this would likely improve it's ability to correctly identify 'good' squats.



6 Conclusion

In summary, the purpose of this project was to assess squat performance using 3D pose data. We compared the performance of two models: a regularized 1D CNN (based on the 1D CNN used by Ogata et al. [1]) and a bidirectional LSTM. The models had similar performance in many aspects. However, the 1D CNN had fewer false positives for the 'good squat' class. This is important for training as telling someone that their squat is 'good' when it is not, could result in someone practicing and memorizing poor form, which could lead to injuries. Most importantly, the CNN was much faster to train, so it is selected as the of choice for this task.

While both models could use improvement in identifying "good" squats, we note that both models achieved great performance in identifying some classes (e.g. the "rounded back" and "shallowness" cases). Correctly identifying these cases suggests that our model could be useful in reducing the risk of back injury and in increasing strength results from training.

To further improve our results, we attempted combining classes that were confused with each other during runs of the original seven class squat classification problem. However, we did not observe any significant improvement in results. Future work that could significantly improve model performance includes using a real-world and more diverse dataset. One limitation of the current dataset is that it was developed from pose data of a single subject performing squats. A real-world and varied dataset with multiple subjects and a more diverse set of errors would likely improve our performance. Additional steps that could yield improved performance include balancing the dataset and tuning hyperparameters via parametric sweep. Future work could also add attention to understand if the models learn from the same features in the temporal distance matrices.

We recommend first trying to use a CNN-based model in activity assessment approaches using deep learning and keypoint information. More study should be done to investigate if the results we found are consistent across activities. Results from this project can inform model choice for future activity assessment and save time and effort in future research. Our code is available at `https://github.com/marissalee20/squat-analysis`.

7 Contributions

All authors contributed to the discussion of the Introduction. R.A. and M.L. contributed to its writing. M.L. conducted the literature review and wrote Related Work. B.P. conducted the initial exploration of the dataset and wrote Data. R.A., M.L., and B.P. contributed to dataset preprocessing. All authors worked to recreate the model previously developed by Ogata et al. M.L. simplified the model, ran CNN experiments, and wrote results. B.P. and M.L. developed the LSTM model, and B.P. ran LSTM experiments and wrote results. R.A. developed confusion matrix evaluations and wrote results. M.L. developed ROC evaluations and wrote results. R.A. ran class combination experiments and wrote results. All authors contributed to discussion of results at all stages. R.A. wrote the conclusion and abstract. B.P. compiled supplementary material.

References

[1] R. Ogata, E. Simo-Serra, S. Iizuka and H. Ishikawa, "Temporal Distance Matrices for Squat Classification," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Long Beach, CA, USA, 2019, pp. 2533-2542, doi: 10.1109/CVPRW.2019.00309.

[2] J. Carreira and A. Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 4724-4733, doi: 10.1109/CVPR.2017.502.

[3] A. Kanazawa, M. J. Black, D. W. Jacobs and J. Malik, "End-to-End Recovery of Human Shape and Pose," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, 2018, pp. 7122-7131, doi: 10.1109/CVPR.2018.00744.

[4] R. Kianifar, A. Lee, S. Raina and D. Kulić, "Classification of squat quality with inertial measurement units in the single leg squat mobility test," 2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Orlando, FL, 2016, pp. 6273-6276, doi: 10.1109/EMBC.2016.7592162.

[5] Pirsiavash H., Vondrick C., Torralba A. (2014) Assessing the Quality of Actions. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8694. Springer, Cham. doi: 10.1007/978-3-319-10599-4_36.

[6] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," in Neural Computation, vol. 9, no. 8, pp. 1735-1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[7] M. Khokhlova, C. Migniot, A. Morozov, O. Sushkova and A. Dipanda, "Normal and pathological gait classification LSTM model," in Artificial Intelligence in Medicine, vol. 94, pp. 54-66, Mar. 2019, doi: 10.1016/j.artmed.2018.12.007

[8] A. Zhao, L. Qi, J. Li, J. Dong and H. Yu, "LSTM for diagnosis of neurodegenerative diseases using gait data," Ninth International Conference on Graphic and Image Processing (ICGIP), Qingdao, China, 2018, doi: 10.1117/12.2305277.

[9] M. H. Lee, D. P. Sierwiorek, A. Smailagic, A. Bernardino and S. B. i Badia, "Learning to assess the quality of stroke rehabilitation exercises," 24th International Conference on Intelligent User Interfaces (IUI), Marina del Ray, CA, 2019, pp. 218-228, doi: 10.1145/3301275.3302273.

[10] N. Y. Hammerla, S. Halloran and T. Plotz, "Deep, convolutional, and recurrent models for human activity recognition using wearables," Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI), New York, NY, 2016, pp. 1533-1540.

[11] M. Abadi, A. Agarwal, P. Barham, et al. "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.