2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Enhancing Conducting Gesture Analysis: Integrating Laban Movement Analysis with Tree Ensembles and Neural Networks

Sean Pierce and Herbert H. Tsang
Applied Research Lab, Trinity Western University
Langley, British Columbia, Canada
herbert.tsang@twu.ca

***Abstract*— Our research focuses on integrating Laban Movement Analysis (LMA) with neural network technologies for analyzing conducting gestures. While promising, this approach faces challenges in real-time speed and adaptability. To overcome these limitations, we propose using tree ensembles, conducting LMA classification on conducting gestures with time-invariant transforms. This study aims to outperform the previous neural network approach in terms of time and set a benchmark for comparison, contributing valuable insights to enhance real-time applications in conducting gestures.**

I. Introduction

Gestures are an important means of communication. Historically, gesture recognition has been focused on producing an output based on specific, conscious actions. Action variation is typically only used for model robustness. However, this variation holds untapped potential, and could improve the responsiveness of gesture recognition techniques. These small variations are especially important in artistic gesture such as movement for dancers and conductors.

To investigate the delicate nuances in motion, it is essential to isolate them from the overall gesture. Opting for an application with simple gestures, where the technique itself conveys most of the information, proves to be advantageous. Musical conducting serves as an evident and ideal choice. Even when conducting basic rhythms, musicians can infuse their music with a distinctive tone, making it a suitable domain for this study.

A. Laban Movement Analysis

Laban Movement Analysis (LMA) is a framework we employed for describing dance and whole-body motion. LMA classifies all motion into four independent binary categories: weight, space, time, and flow, though these may be extended depending on the application. Within each category, there are two descriptors representing exact opposites. Table I shows the possible types of motions can be described by LMA.

TABLE I: Different type of motions that can be described using LMA. We included these actions in our dataset [1].

	Space	**Weight**	**Time**	**Flow/Energy**
Dab	direct	light	quick	free
Flick	indirect	light	quick	free
Glide	direct	light	sustained	free
Float	indirect	light	sustained	free
Punch	direct	strong	quick	bound
Slash	indirect	strong	quick	bound
Press	direct	strong	sustained	bound
Wring	indirect	strong	sustained	bound

B. Tree Ensembles

Neural networks are narrow and deep, progressively learning important features, whereas tree ensembles are shallow and wide, extracting knowledge from base features using decision trees. Decision trees are common in diagnostic applications and serve as weak learners for machine learning tasks. Their simplicity prevents overfitting, leading to consistent performance in ensemble learning compared to other models [2].

Tree ensembles offer several advantages, including speed and human-interpretability. Their evaluation and training are fast due to the simplicity of boolean expressions. Moreover, their human-interpretable nature makes them suitable for applications where understanding the decision process is crucial, like in remote control scenarios.

However, tree ensembles have limitations as they cannot generate new features, making them less effective for complex problems requiring multiple logical steps. They excel in regression and simple categorization tasks but may not be suitable for more intricate challenges.

1) Random Forest: Random Forest is a basic tree ensemble method, utilizing bootstrap sampling for data selection and training decision trees accordingly. To make predictions, it aggregates the outputs from individual trees, using averaging for regression and voting for classification, a process known as bagging.

Random Forest was a leading candidate for LMA classification with accelerometer data in 2014 [3]. Recent studies demonstrate its comparable performance, but not surpassing, Convolutional Neural Networks and Long-Short Term Memory algorithms in LMA classification [4]. In a recent analysis of machine learning methods, Random Forest ranked as the second-highest scoring approach [5]. It has also shown excellent results when combined with Wavelet Packet Decomposition entropy for feature extraction [6].

2) Rotation Forest: Rotation Forest, a modification of Random Forest, aims to enhance the efficiency of using trees. One limitation of Random Forest is its inability to split data on different angles from the feature axes, which

978-0-7381-4408-5/23/$31.00 ©2023 IEEE

may lead to jagged approximations even for linear problems, necessitating multiple trees. Rotation Forest addresses this by providing better axes for splitting. It bootstraps 75% of the data, performs Principal Component Analysis (PCA) on each subset, and rotates the data along the resulting axis [7]. This rotation aligns with the direction of maximum variation, improving the tree splitting process.

In a comparison of ensemble learning methods, Rotation Forest achieved the highest score when utilizing REP trees [2]. Despite its success, Rotation Forest has seen limited use in gesture recognition since its proposal in 2006 [7]. However, recent applications in conjunction with surface electromyography demonstrated better results than using Random Forest alone [8].

3) AdaBoost: AdaBoost is a boosting method, not bagging. Unlike bagging, boosting coordinates weak learners to reduce errors faster, rather than waiting for errors to statistically disappear. AdaBoost selects learners based on their data classification performance, assigning them different weights based on their accuracy. To minimize overfitting, using trees with just two leaves (stumps) is recommended, allowing AdaBoost to systematically eliminate errors by selecting the best splitting axis and location each time. However, being a sequential algorithm, AdaBoost cannot be GPU-enhanced.

AdaBoost has shown success when combined with feature extraction techniques. Notably, it has been used with the two-dimensional Haar wavelet for image classification and in conjunction with SVM, Fourier transform of finger gradients, and modified Hu movements for invariant classification of Kinect sensor data. These applications demonstrate that AdaBoost performs well with feature extraction.

4) Extreme Gradient Boosting (XGBoost): XGBoost (Extreme Gradient Boosting) is an adaptation of the original Gradient Boosting algorithm, which utilizes gradient descent to train an ensemble. It starts with an arbitrary prediction and then trains a weak learner on the pseudo-residuals (differences between the prediction and ground truth) at each step. The new prediction is scaled by the learning rate and added to the previous prediction, gradually approaching the ground truth as more gradients are accumulated.

XGBoost significantly optimizes the algorithm by identifying similar value groups for data splitting, pruning trees to prevent overfitting, and utilizing approximate algorithms to speed up training time. Additionally, XGBoost cleverly executes some steps in parallel, enabling partial GPU acceleration.

The algorithm is quite popular, despite having only been proposed as early as 2016 [9]. This is probably due to XGBoost's parallelization support, ease of use, and high accuracy. The algorithm has achieved better results classifying wifi signals from antennas [10]. XGBoost was also the third best method in the aforementioned analysis [5]. From these results, it appears that XGBoost's performance is sensitive to the context of the problem.

Figure 1 show the qualitative rendition of results from the four tree ensemble algorithms and the three time-invariant transforms we used.

C. Time-Invariant Transforms

To handle semi-periodic data and began at different times, we used three time-invariant transforms for data extraction. This ensured uniform recognition of patterns appearing at various times and encouraged the model to focus on the form rather than the beat of the samples.

1) Fast Fourier Transform (FFT): The standard time-invariant transform, FFT, decomposes signals using complex exponentials. However, tree ensembles cannot handle complex outputs due to their non-orderable nature. To address this, two methods are available: normalizing the output (which maintains input size but is irreversible) or combining the real and imaginary parts (resulting in twelve frequency spectrums, but increasing input features). For our discussion, we will focus on the normalized approach to FFT.

2) Discrete Cosine Transform (DCT): DCT is simpler to work with than FFT, since it maintains the number of datapoints and only outputs real values. It is also used in compression algorithms, meaning the 16 frequency approximation should be more accurate for this transform.

3) Continuous Wavelet Transform (CWT): The Continuous Wavelet Transform uses localized wavelets, not trigonometric functions. This allows it to capture time information as well as frequency information. Therefore, this transform is only partially time-invariant. Instead of producing a time series, the Wavelet Transform outputs a matrix of weights. The matrix height is the wavelet size, and the width is the time axis. We will limit our discussion to the Mexican Hat wavelet for simplicity, though it is possible to use other wavelets as well.

We should expect CWT to perform well based on its ability to capture more than one type of information. In a recent study, wavelets were found to be the most effective method for gesture recognition using phone sensors [11].

II. Methods

In this paper, we are extending our previous work on the CGLER framework by exploring new Neural Network algorithms [1]. The new addition is showed in Figure 2 flowchart.

A. Data and Preprocessing

The dataset we used was provided by the previous CGLER framework, and included both gyroscope and accelerometer data captured from smartphones [1]. Our inputs consisted of six time series per sample, with three for acceleration and three for rotation. Each sample was labeled with the quarter notes per measure (two, three, or four) and LMA categories (Flow, Shape, Time, and Weight). We expected the motion to be captured by only the narrow band of frequencies at which the whole arm can move. Therefore, we used only the 16 lowest of 128 frequencies for FFT and DCT, while for CWT, we used integer wavelet widths from 1–8.

First, the data was read from several files selectively, based on the LMA descriptor currently being classified. The input features were six time series with 128 datapoints.

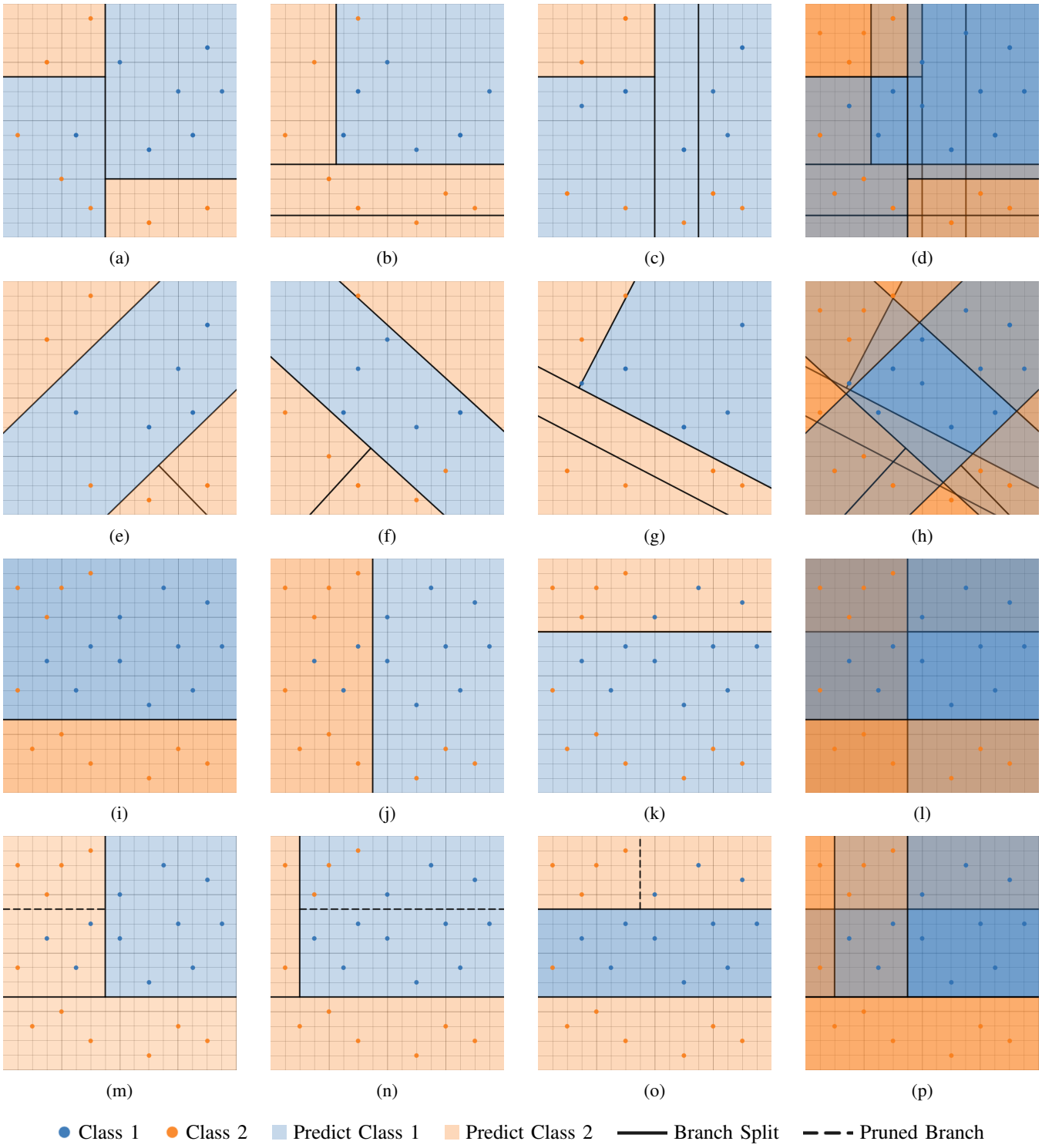

Fig. 1: A qualitative rendition of the tree ensemble algorithms used in this paper. Each row represents a different algorithm: Random Forest (a–d), Rotation Forest (e–h), AdaBoost (i–l), and XGBoost (m–p). In each iteration, the algorithms attempt to predict a class by splitting the tree along an axis, allowing only three splits in the figure. The first three columns display individual decision trees, with possible missing or overlapping points in Random Forest and Rotation Forest due to bootstrapping. In AdaBoost, some points may have higher weights, while XGBoost replaces points with their pseudo-residuals at each iteration. The last column represents the complete ensemble classifier formed by combining the decision trees.

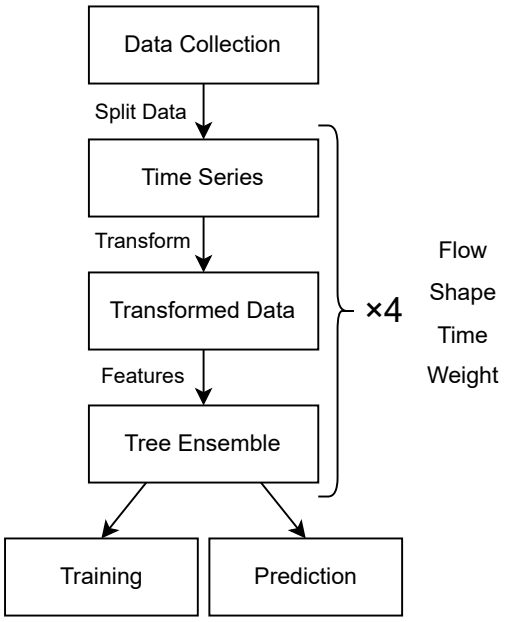

Fig. 2: The flowchart for a single train/test cycle. This was performed several times using K-Means.

We then performed feature extraction using a time-invariant transform.

B. Training and Testing

These input features were passed on to one of the tree ensemble algorithms. We used K-Fold cross-validation for training and testing. The K-Fold algorithm shuffles the data and splits it into k subsets. It trains $k-1$ of those subsets and tests on one, repeating k times. During K-Fold cross-validation, we calculated mean accuracy, Cohen's Kappa, and standard deviation of accuracy.

III. Results and Discussion

Table II summarized the results obtained from various combinations of transform and tree ensemble. Each of these provide an accuracy, Cohen's Kappa, and standard deviation of accuracy for each of the four LMA descriptors (flow, space, time, and weight). In the following sections, we compare the efficacy of each tree ensemble and transform. We then compare these results to the previous neural network approach.

A. Tree Ensemble Comparison

The topmost horizontal axis of Table II lists the tree ensemble methods we used.

Among the tested algorithms, AdaBoost demonstrated outstanding accuracies in the Flow, Time, and Weight categories. The combination of AdaBoost with transforms seemed to enhance its performance significantly. Random Forest performed comparably well, achieving the highest accuracy in the Space category. Surprisingly, XGBoost did not perform as well as expected and was often outperformed by Random Forest. The Rotation Forest algorithm, despite having longer training times, consistently showed the poorest performance across all categories.

Interestingly, in some cases, the axes identified by PCA proved to be less effective than the original ones, indicating that the problem might be too nonlinear for PCA to be an effective solution. Although kernel PCA exists, it primarily serves as a dimensionality reduction technique and does not facilitate axis rotation.

Overall, these findings highlight the importance of choosing the appropriate algorithm and transformation methods for conducting gesture analysis, considering the nonlinear nature of the problem.

Combined with CWT, Rotation Forest ran out of memory while it was training. The exponential growth of decision trees combined with the need to store rotation information makes Rotation Forest particularly memory intensive. For most transforms this was not much of a problem due to the small number of input features. However, CWT produced the most input features of all the transforms, since it captured features across both time and frequency space. This feature set was too large for Rotation Forest algorithm to process, at least on a home computer. Memory issues like this are an important consideration for future research, especially in applications intended to run on less proficient hardware such as a smartphone.

B. Transform Comparison

Table II displays the various transforms used for feature extraction along the leftmost vertical axis. Models utilizing transforms generally outperformed those without, demonstrating improved accuracy in almost every case. Compared the average result of no transform to combined all the transformed, we noted the overall accuracy increased from 77.79% to 80.13% for Random forest, 67.32% to 75.34% for Rotation forest, 75.41% to 80.71% for AdaBoost, and 76.61% to 79.57% for XGBoost.

Compared the various transforms, FFT has consistently performed better than the other transforms. These results indicate that incorporating more information does not necessarily guarantee better performance for tree ensembles. For instance, with the Discrete Wavelet Transform (DWT), additional features not only limited achievable accuracy but also significantly increased training time. It seems that there exists an optimal number of features that maximizes ensemble performance, beyond which minimal variations in accuracy occur. This suggests the presence of an upper bound on accuracy, and further improvements may only be achievable by introducing other types of features.

C. Overall Best Result

Overall, the best combination was AdaBoost with FFT with82.34% accuracy in average. It achieved the best accuracies in the Time and Weight categories. Additionally, the Flow category came within 0.35% of the best accuracy, and the Space category came within 2.25%.

D. Comparison with Neural Network Approach

Unfortunately, none of our accuracies or kappas surpassed those in the previous neural network approach [1]. There are several reasons for this:

1) Training Resources: The previous CGLER framework was trained using resources from Compute Canada [1]. Meanwhile, our method was trained on a home computer. Although the previous model had better accuracy, it likely had a longer training and prediction time. Unfortunately, the previous paper did not include a time benchmark, so a direct comparison was not possible.

2) Interdependence of Variables: As stated previously, tree ensembles do not perform well on datasets that require complex logic to classify. One of the issues with our dataset is the interdependence of the rotation and accelerometer data. A quick rotation can change the direction of acceleration in the phone's coordinates. This has the effect of switching axes at unpredictable intervals, creating different accelerations even if the motion is nearly identical. While a neural network may eventually learn a coordinate rotation or similar operation, a tree ensemble must use the data as-is. A simple transformation to absolute coordinates could improve accuracy significantly.

However, such a transformation comes with its own caveats. A gyroscope measures rotational velocity, not rotation angles. The velocity could be integrated numerically, but any small disturbances would add together. This issue is called drift, and it can be mitigated with more stable integration methods [12].

3) Feature Extraction: There may exist more useful feature extraction techniques than time-invariant transforms. While they can simplify alignment and combat noise through frequency selection, time-invariant transforms also remove valuable time information. Wavelet transforms combat this problem, but large feature sets complicate the classification process. The ideal transform for this scenario is one which contains a minimal number of both time and frequency features.

IV. Conclusion

Using tree ensembles with time-invariant transforms, we classified conducting gestures using the four primary categories of Laban Movement Analysis. Despite not matching the accuracy of the previous neural-network approach, its impressive performance with limited resources, accessibility, and fast training time make it an excellent choice for future research.

There are several opportunities for further investigation:

1) Longer Training: Three of the four highest accuracies were achieved by AdaBoost. Considering AdaBoost's robustness to overfitting, more extensive training of AdaBoost combined with FFT or DCT may improve results without additional changes.

2) Different Feature Extraction Techniques: Due to time constraints, we were only able to test three basic transforms. There may be other feature extraction techniques, including other transforms, which have better performance than our demonstration. Furthermore, it would be advantageous to use global rather than local acceleration coordinates, which can only be achieved using stable integration methods.

3) Combination with Neural Networks: Our model combines the speed of tree ensembles with the accuracy of neural network methods. It is expected to be faster than existing neural network-based approaches and more accurate than using a tree ensemble alone.

V. Acknowledgments

The authors would like to acknowledge the research grants from the following agencies supported the work presented here: the Natural Sciences and Engineering Research Council of Canada (NSERC) and Trinity Western University. This research was enabled in part by support provided by BC DRI Group and the Digital Research Alliance of Canada (alliancecan.ca).

References

[1] F. Tan, G. Woo, and H. H. Tsang, "CGLER: Laban effort framework analysis with conducting gestures using neural networks," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2020, pp. 1452–1459.

[2] S. Jukic, M. Saračević, and J. Kevric, "Comparison of ensemble machine learning methods for automated classification of focal and non-focal epileptic eeg signals," *Mathematics*, vol. 8, 09 2020.

[3] B. Kikhia, M. Gomez, L. L. Jiménez, J. Hallberg, N. Karvonen, and K. Synnes, "Analyzing body movements within the laban effort framework using a single accelerometer," *Sensors*, vol. 14, no. 3, pp. 5725–5741, 2014. [Online]. Available: https://www.mdpi.com/1424-8220/14/3/5725

[4] S. Wang, J. Li, T. Cao, H. Wang, P. Tu, and Y. Li, "Dance emotion recognition based on laban motion analysis using convolutional neural network and long short-term memory," *IEEE Access*, vol. 8, pp. 124 928–124 938, 2020.

[5] S. Bhushan, M. Alshehri, I. Keshta, A. K. Chakraverti, J. Rajpurohit, , and A. Abugabah, "An experimental analysis of various machine learning algorithms for hand gesture recognition," *Electronics*, vol. 11, no. 6, 2022. [Online]. Available: https://doi.org/10.3390/electronics11060968

[6] T. Li and M. Zhou, "Ecg classification using wavelet packet entropy and random forests," *Entropy*, vol. 18, no. 8, 2016. [Online]. Available: https://www.mdpi.com/1099-4300/18/8/285

[7] J. Rodríguez, L. Kuncheva, and C. Alonso, "Rotation forest: A new classifier ensemble method," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, pp. 1619–30, 11 2006.

[8] F. Peng, C. Chen, X. Zhang, X. Wang, C. Wang, and L. Wang, "sEMG-based gesture recognition by rotation forest-based extreme learning machine," in *2021 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 2021, pp. 1122–1127. [Online]. Available: https://doi.org/10.1109/RCAR52367.2021.9517479

[9] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 785–794. [Online]. Available: https://doi.org/10.1145/2939672.2939785

[10] X. Ding, T. Jiang, W. Xue, Z. Li, and Y. Zhong, "A new method of human gesture recognition using wi-fi signals based on XGBoost," in *2020 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, 2020, pp. 237 – 241. [Online]. Available: https://doi.org/10.1109/ICCCWorkshops49972.2020.9209953

[11] I. Trabelsi, J. Francoise, and Y. Bellik, "Sensor-based activity recognition using deep learning: A comparative study," in *Proceedings of the 8th International Conference on Movement and Computing*, ser. MOCO '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3537972.3537996

[12] X. Kong, W. Yang, H. Luo, and B. Li, "Application of stabilized numerical integration method in acceleration sensor data processing," *IEEE Sensors Journal*, vol. 21, no. 6, pp. 8194–8203, 2021.

TABLE II: Mean accuracy, Cohen's Kappa, and standard deviation of accuracy for all combinations of transforms and tree ensembles. Each 4x3 block represents a single model employing a specifc transform. The best results in each LMA category are in bold. Note that Rotation Forest with CWT experienced an out of memory exception, and therefore no results are available for this combination.

			Tree Ensemble											
			Random Forest			Rotation Forest			AdaBoost			XGBoost		
			Overall accuracy	Cohen's Kappa	Standard Deviation	Overall accuracy	Cohen's Kappa	Standard Deviation	Overall accuracy	Cohen's Kappa	Standard Deviation	Overall accuracy	Cohen's Kappa	Standard Deviation
Transform	None	Flow	**83.48%**	0.6697	0.0007	**74.18%**	0.4837	0.0005	**80.25%**	0.6050	0.0008	82.39%	0.6478	0.0006
		Space	68.37%	0.3675	0.0012	56.17%	0.1234	0.0016	65.60%	0.3120	0.0007	63.59%	0.2718	0.0008
		Time	75.83%	0.5167	0.0010	65.83%	0.3167	0.0009	76.13%	0.5225	0.0006	79.75%	0.5951	0.0017
		Weight	**83.48%**	0.6696	0.0002	73.08%	0.4616	0.0003	79.65%	0.5930	0.0008	**80.70%**	0.6139	0.0014
	FFT	Flow	**84.73%**	0.6945	0.0008	80.60%	0.6120	0.0010	86.12%	0.7224	0.0005	**85.62%**	0.7125	0.0004
		Space	68.52%	0.3703	0.0009	64.11%	0.2823	0.0004	68.13%	0.3627	0.0004	69.67%	0.3933	0.0020
		Time	81.67%	0.6333	0.0013	74.51%	0.4902	0.0015	**88.58%**	0.7716	0.0003	84.36%	0.6873	0.0010
		Weight	84.48%	0.6895	0.0005	**81.00%**	0.6198	0.0008	86.52%	0.7303	0.0005	84.38%	0.6875	0.0003
	DCT	Flow	85.82%	0.7164	0.0006	81.34%	0.6269	0.0016	**86.47%**	0.7294	0.0007	84.33%	0.6866	0.0011
		Space	**70.38%**	0.4077	0.0009	64.74%	0.2947	0.0023	66.08%	0.3215	0.0015	68.56%	0.3713	0.0009
		Time	78.92%	0.5784	0.0004	74.71%	0.4941	0.0008	84.26%	0.6853	0.0002	77.55%	0.5510	0.0020
		Weight	**86.07%**	0.7214	0.0005	**81.69%**	0.6338	0.0006	86.27%	0.7254	0.0005	**85.42%**	0.7084	0.0008
	CWT	Flow	85.62%	0.7124	0.0004	Out of Memory			**85.17%**	0.7035	0.0004	**85.12%**	0.7025	0.0007
		Space	69.33%	0.3866	0.0002	Out of Memory			61.39%	0.2278	0.0012	64.02%	0.2804	0.0015
		Time	80.20%	0.6039	0.0014	Out of Memory			84.41%	0.6882	0.0010	80.88%	0.6176	0.0016
		Weight	**85.77%**	0.7154	0.0005	Out of Memory			**85.17%**	0.7035	0.0005	84.93%	0.6985	0.0016
	Avg	Avg (none)	77.79%	0.5559	-	67.32%	0.4346	-	75.41%	0.5081	-	76.61%	0.5322	-
		Avg (w/ transform)	**80.13%**	**0.6025**	-	**75.34%**	**0.5067**	-	**80.71%**	**0.6143**	-	**79.57%**	**0.5914**	-
		Avg (FFT)	79.85%	0.5969	-	75.06%	0.5011	-	**82.34%**	0.6468	-	81.01%	0.6202	-
		Avg (DCT)	80.30%	0.6060	-	75.62%	0.5124	-	**80.77%**	0.6154	-	78.97%	0.5793	-
		Avg (CWT)	**80.23%**	0.6046	-	-	-	-	79.04%	0.5808	-	78.74%	0.5748	-

FFT = Fast Fourier Transform, DCT = Discrete Cosine Transform, CWT = Continuous Wavelet Transform