Human Interaction and Emerging Technologies (IHIET-AI 2025), Vol. 161, 2025, 290–300
https://doi.org/10.54941/ahfe1005921

Deploying a Transformer-Based Model in Microservices Architecture: An Approach for Real-Time Body Pose Classification

Enrique Bances[1], Vedant Dalvi[1], Urs Schneider[1,2], and Thomas Bauernhansl[1,2]

[1]Institute of Industrial Manufacturing and Management, University of Stuttgart, Stuttgart 70569, Germany
[2]Institute for Manufacturing Engineering and Automation, Fraunhofer Society, Stuttgart 70569, Germany

ABSTRACT

Real-time body pose classification is essential in preventing injuries caused by repetitive strain or poor ergonomics. In industrial environments, ensuring worker safety often requires monitoring the poses of multiple individuals performing different tasks. However, analysing the movements of many workers simultaneously presents computational challenges, potentially impacting accuracy and latency. In this context, microservices architecture offers significant advantages for enabling individual application functionalities to operate independently. Also, this architecture allows systems to scale efficiently in response to specific workload demands by adding CPU, memory, and storage resources, improving system performance and resource efficiency. This study evaluates the scalability of a real-time body pose classification system deployed using a microservices architecture, comparing it against a traditional monolithic approach. The system used a transformer-based model designed to monitor awkward body positions and identify constraints in joint movements. The methodology involves offline training on sequential data representing body joint angles, collected using an IMU sensor-based motion capture (MoCap) system. The system streams joint angles wirelessly from participants performing logistic tasks, such as lifting and carrying sandbags, in an industrial setting. Once trained, the classification model is deployed in real-time, processing streaming data for live body pose classification. Inference results from both architectures are stored in a time-series database for performance analysis. Scalability tests were conducted by deploying services for varying numbers of participants (one, three, five, and ten) in parallel across both architectural setups. Data throughput, latency, and resource utilisation (CPU and memory usage) were monitored during load testing. The results show that the microservices architecture outperforms the monolithic architecture in scalability. When scaled to accommodate multiple participants, it achieved higher data throughput, reduced latency by 18-48%, and decreased CPU usage by 18-44%. These findings validate the effectiveness of microservices architecture in enhancing the performance and scalability of real-time body pose classification systems.

Keywords: Microservices, Transformer model, Body pose classification, Scalability, Sequential data

INTRODUCTION

Body pose classification (BPC) using inertial measurement unit (IMU) sensors is frequently used in industrial applications (Caputo et al., 2019), offering significant advantages for safety and ergonomics assessments (Salisu, 2023). These sensors are portable, cost-effective, and scalable, making them ideal for large-scale deployments requiring flexibility and quick setup (Ciklacandir and Isler, 2022).

Deep learning (DL) techniques using IMU sensor data have gained popularity in BPC. For instance, long short-term memory (LSTM) method is particularly designed to process sequential data and conduct time series analysis. LSTMs effectively capture and retain patterns of body movements over time, enabling accurate predictions of future states (Arras et al., 2019). LSTM applications span diverse fields, including sports science, where they enhance performance analysis (Kranzinger et al., 2023), and rehabilitation, where they support safety and ergonomics (Sherratt and Iravini, 2021). LSTMs are also integral to industries such as human-robot interaction (Jaramillo et al., 2022) and gesture recognition, enabling systems to understand and respond to human movements precisely (Kavarthapu, 2017).

Transformer models widely used in tasks such as natural language processing, machine translation, and computer vision (Zhan et al., 2023), are gaining attention in BPC due to their ability to handle long sequences, capture contextual relationships, and improve computational efficiency (Vaswani, 2017). Unlike LSTMs, which are designed for sequential data processing, transformer models utilise a self-attention mechanism to assess the importance of input data in parallel, resulting in faster training and inference times for long sequences (Vaswani, 2017). However, transformers require larger datasets and higher computational resources to perform optimally (Wen et al., 2022).

Integrating transformer models with microservices architecture (MSA) enhances their parallel processing capabilities. MSA enables distributed deployment of model components as independent services, improving scalability, maintainability, and the ability to handle multiple data streams simultaneously (Li et al., 2021). This reduces inference time and supports real-time communication via frameworks like stream processing tools and message brokers, facilitating the development of complex, high-performance applications (Desai et al., 2020).

Finally, this paper leverages MSA to deploy transformer models, presenting a real-time BPC system's implementation, latency performance and scalability evaluation. The selected event-driven architecture supports integrating multiple external systems with diverse data sources, enabling concurrent, instantaneous processing of event streams (Bances et al., 2024).

DEVELOPING REAL-TIME BODY POSE CLASSIFICATION USING TRANSFORMER MODEL

The system implementation for BPC was structured into four phases. The motion capture (Mocap) suit was calibrated and prepare for use in the first

phase. During the second phase, data on body joint angles was collected from test subjects performing a specific logistic task such as lifting and carrying sandbags. The data was cleaned and processed to train a deep-learning model for BPC. In the third phase, a data streaming pipeline was set up from the Mocap suit using the IoT messaging protocol MQTT. The trained model was then deployed within microservices and monolithic architecture-based applications. Finally, in the fourth phase, experiments were conducted to compare and validate the various advantages of the microservices over the monolithic architecture.

Data Source

For this project, we used the Rokoko Smartsuit Pro II, equipped with IMU sensors placed throughout the body, including the neck, shoulders, scapulae, thorax, elbows, pelvis, wrists, hips, knees, and ankles. Data was collected via Rokoko Studio Legacy software after calibrating the suit. The recordings captured position, velocity, acceleration, and orientation metrics at various frame rates. The software enabled data editing and export in formats like FBX, BVH, CSV, and C3D. We exported the CSV data for model training and FBX data for analysing the test subjects' actions as animations.

Dataset Preparation and Model Training

The logistic task selected for data collection involved five test participants performing actions such as lifting, carrying, and placing a heavy object. We set up a test scenario with a storage bin filled with 8 kg sandbags positioned 170 cm from a table. The task was categorised into five sub-classes or postures: standing, lifting, carrying, placing, and unnatural/awkward positions (see Figure 1).

Figure 1: The selected classes or postures of the logistic task were used to train the model.

Collecting data from test subjects with varying body types and age groups was essential to ensure that the deep learning model generalises well to unseen data. Additionally, data collected from the same participant can differ because the positions of the sensors on the motion capture suit may not be consistent each time the subject performs the task or wears the suit again. The data from the motion capture suit was recorded at 60 FPS.

To collect data, each test subject performed the logistic task ten times, with slight variations in execution speed and body joint movements.

This approach was designed to build a robust dataset that enhances model generalisation capabilities. Additionally, data was collected class-wise to minimise overlap between postures.

Data from 50 task repetitions were collected and split into a 40:10 ratio for training and testing the deep learning models. Out of 47,409 data points, 37,827 were used for model training, while 9,582 were reserved for testing. The dataset is nearly balanced, though the standing class has more data points than the other classes (see Figure 2a).

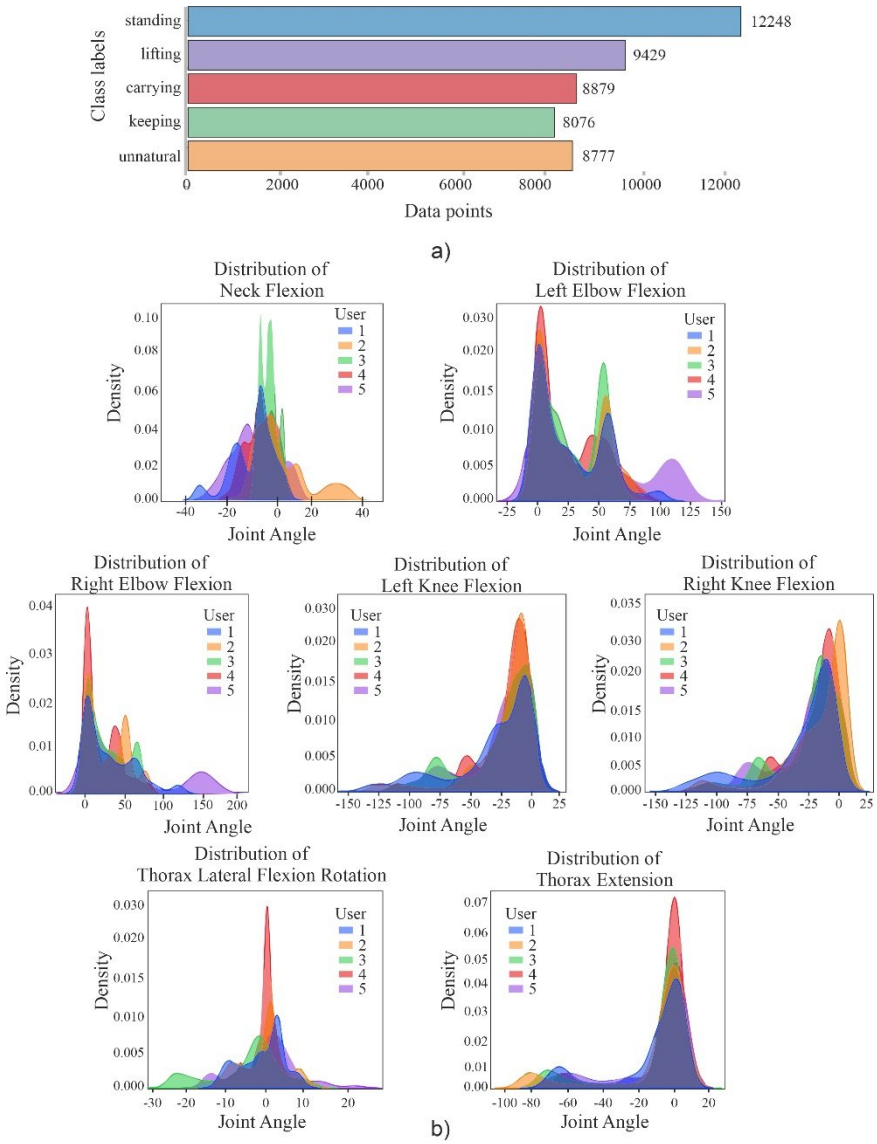

Figure 2: a) Class distribution of the collected data. b) Among the 51 potential features representing body joint activity levels across five users, kernel density estimation (KDE) identified seven as the most active during the logistic task. These features were selected for further analysis.

The suit captures 51 body movement features, including neck flexion, pelvis extension, and ankle dorsiflexion. To identify the most relevant movements, we analysed the distribution of each feature across all users using kernel density estimation (KDE) plots (Weglarczyk, 2018). These plots provide a clear visualisation of the probability density for continuous variables, enabling us to assess the range of motion for each body joint and observe joint behaviour during the selected task. Furthermore, the KDE plots reveal patterns and similarities in joint movements across the five users. After thoroughly examining the KDE plots for all 51 features, we selected seven fundamental movements that were the most active during logistic tasks: neck flexion, left and right elbow flexion, thorax extension, thorax lateral flexion rotation, and left and right knee flexion (see Figure 2b). The collected data for the seven features is then pre-processed through a series of steps, including checking for non-numeric and missing values, labelling, scaling, sequencing, and stacking. This process results in creating a 3D input tensor (see Figure 3), which is then split into training and test sets for the transformer model.

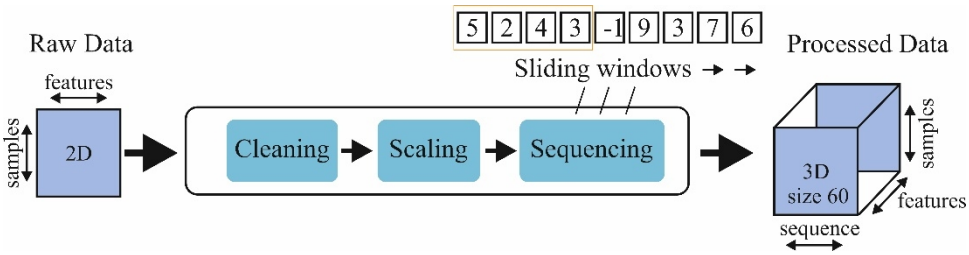

Figure 3: Following the scaling of the dataset, the next step in the data processing pipeline involved creating sequences from the scaled features and stacking these sequences. This transformation was necessary because deep learning models require input in the form of a 3-dimensional tensor, while the dataset is currently structured as a 2-dimensional array defined by the number of samples and features.

Transformer Model Training

The transformer architecture employs an encoder-decoder structure enhanced by a self-attention mechanism (Vaswani, 2017). Unlike other deep learning models, such as LSTMs, which process data sequentially, transformers utilise self-attention to process data in parallel. Transformers handle data in parallel using a mechanism know as self-attention. This mechanism allows the model to weigh the importance of each part of the input sequence independently, making it highly efficient and capable of capturing long-range dependencies. The implemented architecture consists of an encoder and a decoder, each comprising multiple layers. For sequential data classification tasks, only the encoder block is used to capture input sequences' underlying characteristics and contextual dependencies. The encoder transforms the input sequence or input tensor into a continuous representation. Positional encoding is added to the input embeddings to retain the order of the sequence, and multi-head attention allows the model

to focus on different parts of the input simultaneously. Figure 4 shows a representation of the implemented model.

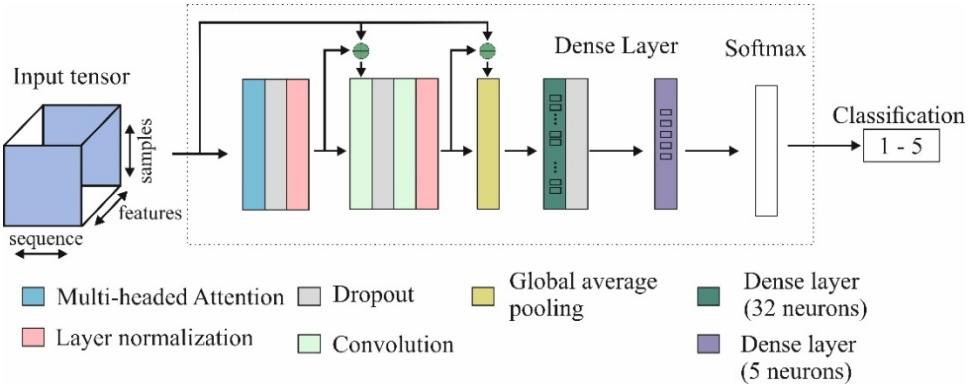

Figure 4: The architecture of the transformer model, illustrating the encoder block with multi-head self-attention mechanism, feed-forward layers, and the final output layer used for classification.

Table 1 shows the hyperparameters found for the best classification accuracy of the model. These parameters are set before the learning process begins, significantly influencing the training and performance of models.

Table 1. Hyperparameter used for the transformer model.

Hyperparameters	Value
Heads	1
Head size	16
Batch size	32
Learning rate	1e-4
Dropout	0.1
Optimiser	Adam
Activation function	SoftMax
Loss function	CE
Sequence size/ timesteps	60
Epochs	200

Training Results

The loss function is the Categorical Cross Entropy (CE) loss, also called Softmax loss, defined in Eq. 1. This loss function is a combination of a Softmax activation function and a cross-entropy loss, which is used in neural networks for multi-class classification. By minimising loss, the model learns to assign greater probabilities to accurate classes while decreasing probabilities for wrong classes, thereby improving classification accuracy. Figure 5a shows the training loss curve, showing that the training and validation losses decrease exponentially up to a point and converge after

this point. There is no significant gap between the training and validation loss, called a generalisation gap. No gap between training and validation loss indicates a good model fit to the training data and good generalization with unseen data.

$$CE = -\sum_{i=1}^{i=N} y\_true_i \cdot \log\left(y\_pred_i\right) \tag{1}$$

where *y_pred* is the predicted class and *y_true* is the actual class of an N-class classification problem.

Furthermore, a confusion matrix was used to evaluate the classification model's performance. In the confusion matrix of the trained classifier, it can be observed that the model is classifying all five classes of the logistic task, which indicates that there is no bias in the trained model for any class of the classification problem (see Figure 5b). The results of the offline (not real-time) model training are summarized in Table 2. Despite the availability of limited data, the transformer model successfully learned the underlying characteristics of the input sequences of body joint angles and predicted the body poses with excellent accuracy. The trained model's weights were stored in HDF5 format and will be deployed later for real-time inference.

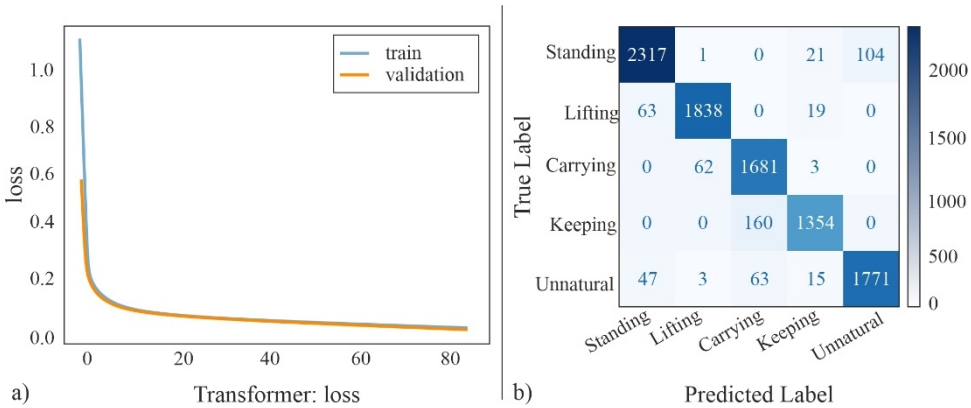

Figure 5: (a) The training loss curve shows the progression of loss reduction over epochs during the training of the transformer model. The curve indicates how the model's performance improved as it learned from the data, with a decrease in loss over time reflecting effective optimisation. (b) The confusion matrix displays the model's classification performance across different classes. Each cell represents the number of predictions made for a specific class, with the diagonal cells indicating correct classifications and off-diagonal cells representing misclassifications.

Table 2. Model training results.

Model	Model Parameters	Model Size (kB)	Test Accuracy (%)
Transformer	1019	3.98	94.11

Deployment of Real-Time Data Streaming and Inference Pipeline in Microservices

This phase consists of the data streaming and real-time inference pipelines. Body joint angles are continuously streamed from the motion capture system in the data streaming pipeline. A script application processes these angles and publishes them to an MQTT topic. Following this, two microservices are deployed on the server to handle real-time data processing and inference.

Microservices ensure flexibility and scalability, enabling the application to efficiently handle multiple models simultaneously. The process remains simple, as loading the respective model files in HDF5 format requires no additional coding, allowing for easy duplication of the same code across multiple models. The resulting performance metrics and inference outcomes are stored in separate databases, facilitating further comparative analysis and visualisation.

System Evaluation, Latency, and Scalability Test

An experimental evaluation was conducted to assess the system's latency and scalability. An emulator streams the collected training data to simulate simultaneous data streams for multiple users, addressing the limitation of having only two motion capture suits. To create data for new users, the original training dataset was augmented by adding Gaussian noise of varying intensities and scaling the data with different factors. The transformer model was deployed in monolithic and microservices architectures and tested simultaneously with data from one, three, five, and ten participants.

Request latency was measured as the time elapsed from receiving the streamed raw data to the end of the model's inference. Results, indicate that the microservices architecture achieved 18–48% lower latency across all experiments. Figure 6 illustrates a drastic increase in latency for the monolithic-based application, while latency remained stable and changed gradually with the microservices approach. These results further highlight the latency advantages offered by microservices.

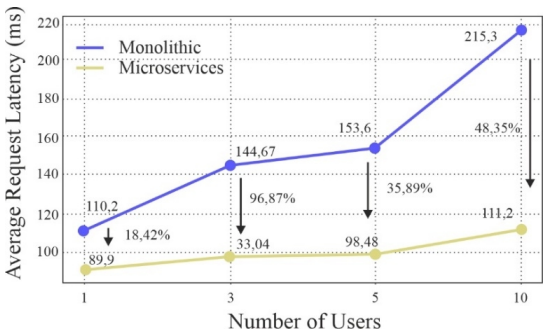

Figure 6: The latency results show the time the model takes to process and generate inferences from the streamed data. These results are measured in milliseconds and highlight the system's efficiency in real-time processing. Lower latency indicates faster response times, ensuring the model can perform real-time predictions with minimal delay.

Scalability evaluation measures the system's ability to handle increased data processing workloads without significant performance degradation. CPU usage was the metric to evaluate scalability in both architectures. The results show an 18–44% reduction in CPU consumption with the microservices architecture, highlighting its superior scalability and efficiency in managing higher workloads compared to the monolithic approach (see Figure 7).

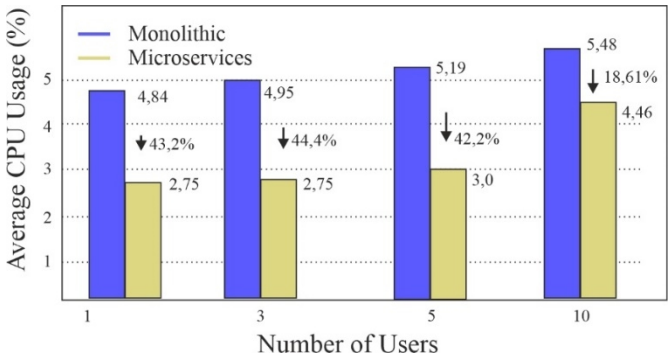

Figure 7: The scalability results demonstrate the system's ability to handle increasing inferences services per participant and data streams without significant performance degradation. In all cases, the CPU usage in the microservices is reduced.

CONCLUSION

This paper presents the implementation of a real-time body pose classification system based on a transformer model. The model was deployed in monolithic and microservices architectures and tested with data from one, three, five, and ten participants. An experimental evaluation was conducted to assess the system's latency and scalability. The results showed a latency reduction of 18–48% and an 18–44% decrease in CPU usage with the microservices architecture. These findings highlight the microservices approach's advantages in latency, scalability, and efficiency in managing higher workloads compared to the monolithic architecture.

This study also showcases the versatility and advantages of the transformer architecture. The architecture has parallelization, the ability to learn long-term dependencies and high scalability. This paper also shows that the transformer architecture can learn from a limited amount of quality and correctly processed data. However, the transformer model is computationally expensive to train on a larger dataset.

ACKNOWLEDGMENT

Supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2120/1 – 390831618.

REFERENCES

Arras, L., Arjona-Medina, J., Widrich, M., Montavon, G., Gillhofer, M., Müller, K. R.,... & Samek, W. (2019). Explaining and interpreting LSTMs. *Explainable ai: Interpreting, explaining and visualizing deep learning*, 211–238.

Bances, E., Schneider, U., Bauernhansl, T., & Siegert, J. (2024). Enhancing Ergonomics in Construction Industry Environments: A Digital Solution With Scalable Event-Driven Architecture. *Human Factors and Systems Interaction*, 103.

Caputo, F., Greco, A., D 'Amato, E., Notaro, I., & Spada, S. (2019). Imu-based motion capture wearable system for ergonomic assessment in industrial environment. In Advances in Human Factors in Wearable Technologies and Game Design: Proceedings of the AHFE 2018 International Conferences on Human Factors and Wearable Technologies, and Human Factors in Game Design and Virtual Environments, Held on July 21–25, 2018, in Loews Sapphire Falls Resort at Universal Studios, Orlando, Florida, USA 9 (pp. 215–225). Springer International Publishing.

Ciklacandir, S., Ozkan, S., & Isler, Y. (2022, September). A comparison of the performances of video-based and imu sensor-based motion capture systems on joint angles. In *2022 Innovations in Intelligent Systems and Applications Conference (ASYU)* (pp. 1–5). IEEE.

Desai, V., Koladia, Y., & Pansambal, S. (2020). Microservices: Architecture and technologies. *Int. J. Res. Appl. Sci. Eng. Technol*, *8*(10), 679–686.

Fernandes, C., Matos, L. M., Folgado, D., Nunes, M. L., Pereira, J. R., Pilastri, A., & Cortez, P. (2022). A deep learning approach to prevent problematic movements of industrial workers based on inertial sensors. In *2022 International Joint Conference on Neural Networks (IJCNN)* (pp. 01–08). IEEE.

Jaramillo, I. E., Jeong, J. G., Lopez, P. R., Lee, C. H., Kang, D. Y., Ha, T. J.,... & Kim, T. S. (2022). Real-time human activity recognition with IMU and encoder sensors in wearable exoskeleton robot via deep learning networks. *Sensors*, *22*(24), 9690.

Kavarthapu, D. C., & Mitra, K. (2017). Hand gesture sequence recognition using inertial motion units (IMUs). In *2017 4th IAPR Asian Conference on Pattern Recognition (ACPR)* (pp. 953–957). IEEE.

Kranzinger, C., Bernhart, S., Kremser, W., Venek, V., Rieser, H., Mayr, S., & Kranzinger, S. (2023). Classification of Human Motion Data Based on Inertial Measurement Units in Sports: A Scoping Review. *Applied Sciences*, *13*(15), 8684.

Menolotto, M., Komaris, D. S., Tedesco, S., O'Flynn, B., & Walsh, M. (2020). Motion capture technology in industrial applications: A systematic review. *Sensors*, *20*(19), 5687.

Lee, S., Koo, B., Yang, S., Kim, J., Nam, Y., & Kim, Y. (2022). Fall-from-height detection using deep learning based on IMU sensor data for accident prevention at construction sites. *Sensors*, *22*(16), 6107.

Li, J., Liu, X., Wang, Z., Zhao, H., Zhang, T., Qiu, S.,... & Cangelosi, A. (2021). Real-time human motion capture based on wearable inertial sensor networks. *IEEE Internet of Things Journal*, *9*(11), 8953–8966.

Li, S., Zhang, H., Jia, Z., Zhong, C., Zhang, C., Shan, Z.,... & Babar, M. A. (2021). Understanding and addressing quality attributes of microservices architecture: A Systematic literature review. *Information and software technology*, *131*, 106449.

Salisu, S., Ruhaiyem, N. I. R., Eisa, T. A. E., Nasser, M., Saeed, F., & Younis, H. A. (2023). Motion Capture Technologies for Ergonomics: A Systematic Literature Review. *Diagnostics*, *13*(15), 2593.

Sherratt, F., Plummer, A., & Iravani, P. (2021). Understanding LSTM network behaviour of IMU-based locomotion mode recognition for applications in prostheses and wearables. *Sensors*, *21*(4), 1264.

Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J., & Sun, L. (2022). Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*.

Weglarczyk, S. (2018). Kernel density estimation and its application. In ITM web of conferences (Vol. 23, p. 00037). EDP Sciences.

Zhang, S., Fan, R., Liu, Y., Chen, S., Liu, Q., & Zeng, W. (2023). Applications of transformer-based language models in bioinformatics: A survey. *Bioinformatics Advances*, *3*(1), vbad001.