Gait Mode Classification using a Temporal Convolutional Neural Network

CSC 499 - Independent Research in Computer Science, Fall 2025

Nicholas Sutton
Department of Computer Science
North Carolina State University
nwsutton@ncsu.edu

Jaemin Lee
Department of Mechanical & Aerospace Engineering
North Carolina State University
jlee267@ncsu.edu

Peng Gao
Department of Computer Science
North Carolina State University
pgao5@ncsu.edu

***Abstract*—This paper addresses the challenge of real-time human gait classification from motion capture data as a fundamental component of human teleoperation systems for quadrupedal robots. We present a learning-based approach using a Temporal Convolutional Neural Network (TCN) to classify human gaits from motion capture sequences. Our method leverages the temporal modeling capabilities of TCNs to capture the sequential patterns inherent in human locomotion. We compare our model to a Convolutional Neural Network (CNN) and a Bidirectional TCN without causal convolution to validate the performance of our approach.**

I. Introduction

A. Problem Statement

Motion-driven teleoperation is a promising control method for robots that operate in complex and dynamic environments, where fully autonomous methods may fail. Different embodiments such as quadrupedal robots offer significant advantages over humanoids for such applications, including superior stability, adaptability over challenging terrains, and numerous distinct gait types. A critical component of such teleoperation systems is the ability to accurately classify human gaits in real-time, enabling appropriate motion mapping from human to robot.

Significant developments have been made in transferring human motions to robots as a control method. However, most current literature focuses on retargeting for robots morphologically similar to the human demonstrator. Works that do explore cross-morphology teleoperation adopt an end-to-end learning approach which is limited in its flexibility due to the model being trained to both interpret the human motion and control the robot. A modular approach that separates gait classification from motion retargeting could provide greater flexibility and enable easier adaptation across different robotic platforms.

This work focuses on developing a robust human gait classification system as a foundational component for future teleoperation applications. We present a data-driven approach in which human motion features are classified using a Temporal Convolutional Neural Network (TCN). The model is trained on gait features derived from motion capture data of a human subject performing various motions. While designed with quadrupedal robot teleoperation in mind, this work specifically addresses the gait classification challenge, with deployment and motion retargeting left as future work.

B. Related Works

Various learning-based approaches have been developed for gait classification for robotic systems. Wang et al. [1] proposed a model-based classifier using inertial measurement unit (IMU) data for real-time gait recognition on lower limb exoskeletons. While Wang et al. [1] achieved high accuracy (97.91%) across 12 locomotion modes, their approach relies on a Convolutional Neural Network (CNN) which is primarily designed for modeling spatial features and does not explicitly model the temporal nature of locomotion.

Similarly, Sunny et al. [2] employed a temporal deep learning network to classify gait phases from lower limb joint angles. While the use of a temporal network in this paper showed immense promise in modeling gaits, their network was trained to classify gait phases, not distinct gait modes.

Kim et al. [3] employed Bi-LSTM networks for both phase and gait classification. This architecture captures short-term temporal patterns but may struggle with long-range dependencies due to the model's sequential data processing.

Temporal Convolutional Networks (TCNs) offer an alternative approach for modeling temporal sequences. Originally proposed by Lea et al. [4] for action segmentation tasks, TCNs use dilated causal convolutions to capture multi-scale temporal patterns while enabling parallel processing. This architecture has been shown to match or exceed LSTM performance on various sequence modeling tasks while offering computational advantages [5].

II. Methodology

A. Motion Capture System

Our motion data was captured using six OptiTrack PrimeX 13 cameras. Tracking subjects wore a total of 36 reflective

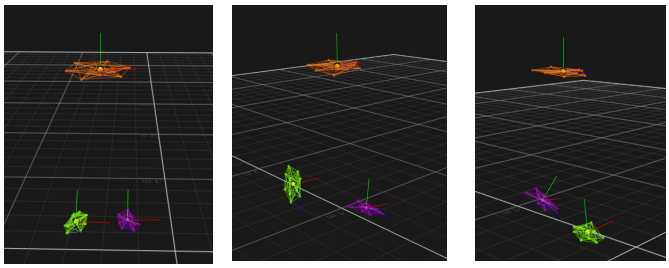


Fig. 1. Tracking subjects' waist and foot rigid bodies.

markers attached to the subject's waist and feet. Using the Motive builder tool, a rigid body was created for each body part using 12 markers for each rigid body (Fig. 1).

B. Data Preparation

The dataset was captured during two sessions at 240Hz and 120Hz, respectively. Both sessions consist of 8 individual gaits as well as mixed locomotion. The 8 gait types represented in the dataset are stand, walk, jog, quasi-static, step-up, step-down, stair ascend, and stair descend.

The raw data was exported as CSV files using OptiTrack's Motive software. The recorded data consists of the position for the waist (base), left foot (left), and right foot (right) rigid bodies relative to the world frame. From this information, we derive the remaining features.

The state vector at time t is defined as:

$$\boldsymbol{s}_t = \begin{pmatrix} \boldsymbol{r}_t & \boldsymbol{f}_t^{\text{left}} & \boldsymbol{f}_t^{\text{right}} & \boldsymbol{g}_t & \boldsymbol{c}_t \end{pmatrix} \in \mathbb{R}^{34} \tag{1}$$

where each component encodes specific biomechanical features:

Base kinematics: $\boldsymbol{r}_t \in \mathbb{R}^9$

$$\boldsymbol{r}_t = \begin{pmatrix} \boldsymbol{v}_t^{\text{base}} & \boldsymbol{\omega}_t^{\text{base}} & \boldsymbol{a}_t^{\text{base}} \end{pmatrix} \tag{2}$$

with linear velocity, angular velocity, and acceleration in $\mathbb{R}^3$.

Foot kinematics: $\boldsymbol{f}_t^i \in \mathbb{R}^9$ for $i \in \{\text{left}, \text{right}\}$

$$\boldsymbol{f}_t^i = \begin{pmatrix} \boldsymbol{v}_t^i & \boldsymbol{a}_t^i & \boldsymbol{p}_t^i \end{pmatrix} \tag{3}$$

comprising linear velocities, accelerations, and relative positions in $\mathbb{R}^3$.

Gait metrics: $\boldsymbol{g}_t \in \mathbb{R}^4$

$$\boldsymbol{g}_t = \begin{pmatrix} \boldsymbol{l}_t^{\text{step}} & \boldsymbol{w}_t^{\text{step}} & \boldsymbol{h}_t^{\text{step}} & \boldsymbol{h}_t^{\max} \end{pmatrix} \tag{4}$$

representing step length, width, height, and maximum foot height.

Contact state: $\boldsymbol{c}_t \in [0,1]^2 \times \{0,1,2\}$

$$\boldsymbol{c}_t = \begin{pmatrix} \boldsymbol{c}_t^{\text{left}} & \boldsymbol{c}_t^{\text{right}} & \boldsymbol{e}_t^{\text{support}} \end{pmatrix} \tag{5}$$

where c_t^i denotes contact probability and e_t^{support} encodes single-support (0), double-support (1), or flight (2) phases.

All continuous features are normalized to the range $[0,1]$ using min-max scaling and integer encoding was used for non-numeric features.

C. Temporal Convolutional Neural Network Architecture

Locomotion is inherently temporal, as the state of a person at any point in the gait cycle relies on their previous state. To model the temporal properties of locomotion, we adopt a Temporal Convolutional Neural Network (TCN).

The network consists of an input projection layer followed by a stack of residual temporal blocks with exponentially increasing dilation factors, culminating in a global pooling and classification layer.

The input to the TCN is a sliding window of sequential states over $\tau = 200$ timesteps and stride 50:

$$\boldsymbol{S}_t = \begin{pmatrix} \boldsymbol{s}_{t-\tau+1} & \boldsymbol{s}_{t-\tau+2} & \cdots & \boldsymbol{s}_t \end{pmatrix} \in \mathbb{R}^{\tau \times 34} \tag{6}$$

where each $\boldsymbol{s}_i \in \boldsymbol{S}_t$ is the state vector defined in (1) and $\boldsymbol{s}_t$ represents state at the current timestep.

The architecture begins with an input projection layer that maps S_t to 64 channels via 1×1 convolution with batch normalization. The core TCN architecture is composed of $L = 3$ stacked temporal blocks with dimensions $\{d_1, d_2, d_3\} = \{128, 256, 512\}$.

Each temporal block consists of two sequential 1D convolutional layers with kernel size $k = 7$ and dilation 2^i. To prevent information leakage from future timesteps, a padding of size $(k-1) \cdot 2^i$ and a chomping operation is applied to remove the rightmost padding. After each convolutional layer we apply batch normalization, dropout with rate $p = 0.15$, and a ReLU activation function. After the second convolution layer, we apply a Squeeze-and-Excitation (SE) block for channel-wise attention. The SE block first performs global average pooling to produce channel-wise statistics, then passes these through two fully connected layers with a reduction ratio of 16.

Each temporal block implements a residual connection to facilitate gradient flow and enable deeper architectures:

$$Y = \text{ReLU}(F(X) + X) \tag{7}$$

When the input and output channel dimensions differ, we apply a learned projection to the residual path:

$$Y = \text{ReLU}\big(F(X) + W_{\text{proj}}(X)\big) \tag{8}$$

where W_{proj} is a 1×1 convolution followed by batch normalization that projects X to match the output dimensionality.

Instead of standard global average pooling, we employ a dual pooling strategy. We apply both adaptive average pooling and adaptive max pooling to the output of the final temporal block, then concatenate the results.

The classification head consists of two fully connected layers with intermediate normalization and regularization. The first layer maps from the concatenated pooled features to 2048 dimensions with batch normalization, followed by ReLU activation and dropout with $p = 0.15$. The second layer produces the final 8 class logits.

D. Training Procedure

We prepared the data for training by first concatenating all the training sessions into a single large session. Then we divided the entire dataset into sliding windows S_t. Each S_t was labeled using majority voting based on the labels of the individual time steps in each window.

The training and test datasets were constructed using a stratified sample of the entire dataset with a split of 80% training data and 20% test data.

To combat the class imbalances in our data, we calculated the distribution of each class in the training dataset before each training session. The classes were then weighted based on their distribution in the training set. We then employed an Adaptive Moment Estimation (Adam) optimizer to adjust these weights during training.

To optimize the convergence of our model, we utilized the PyTorch ReduceLROnPlateau scheduler with a learning rate of $\eta = 0.0004$ as well as an early stopping mechanism that stopped training if the validation loss exceeded the previous minimum value.

III. Results and Discussion

A. Performance Metrics

We compare the performance of our TCN model to a standard CNN and a bidirectional TCN that does not perform causal convolution. To evaluate the performance of these models, we use the following metrics.

$$\text{Cross Entropy Loss} = -\sum_{i=1}^{C} y_i \cdot \log(p_i) \tag{9}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{10}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{11}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{12}$$

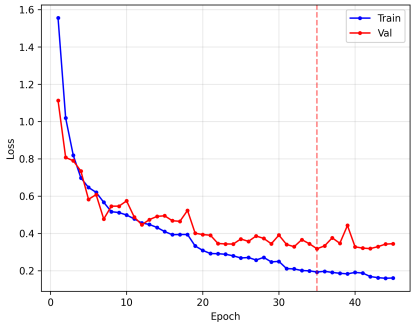
$$\text{F1-Score} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}} \tag{13}$$

TABLE I
Performance comparison of CNN, Bi-TCN, and TCN models. Acc. = Accuracy, Prec. = Precision, Rec. = Recall.

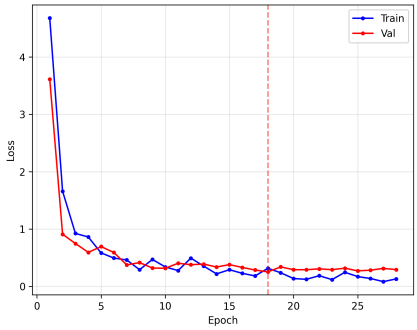
Model	Train Loss	Val Loss	Acc. (%)	Prec.	Rec.	F1
CNN	0.16	0.35	89.47	0.87	0.90	0.88
Bi-TCN	0.11	0.22	94.84	0.95	0.95	0.95
TCN	0.29	0.21	95.46	0.95	0.96	0.95

B. Evaluation

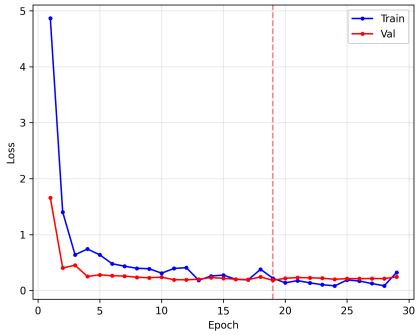
The TCN model achieves superior performance across all evaluated metrics compared to the baseline CNN model. The TCN achieved 95.46% accuracy compared to the CNN's 89.47%, representing a 6.03% improvement. This performance gain is further reflected in the precision (0.95 vs 0.87), recall (0.96 vs 0.90), and F1-score (0.95 vs 0.88), indicating that the TCN not only classifies more samples correctly but also maintains better balance between precision and recall.



(a) CNN Training Curve



(b) Bi-TCN Training Curve



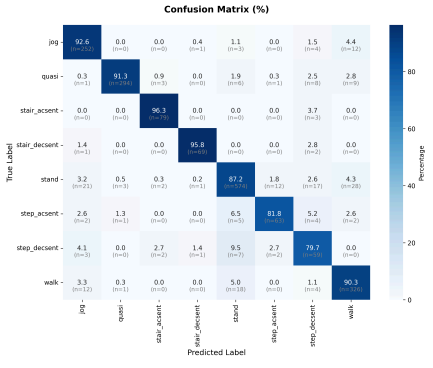
(c) TCN Training Curve

Fig. 2. Training Curves for (a) CNN, (b) Bi-TCN, and (c) TCN models.

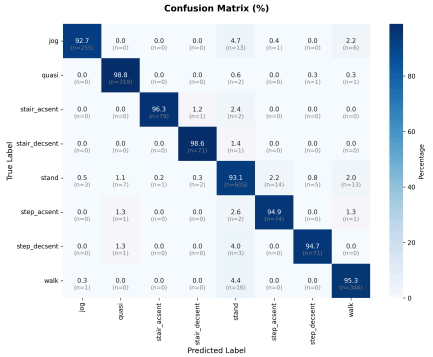
The performance improvements of the TCN over the Bidirectional TCN were smaller but still apparent. The TCN saw an accuracy improvement of 0.62% and has a smaller gap between the training and validation loss. The true benefit of the causal TCN is better displayed when deployed online. The Bidirectional TCN has access to future information that would not be available during real-time classification. For that reason, the TCN provides clear benefits when deployed for online classification.

C. Training Dynamics

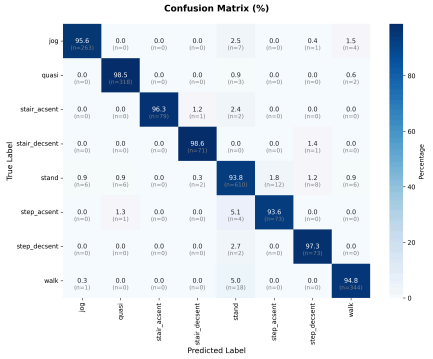
The training curves (Fig. 2) demonstrate effective convergence for all models. Compared to the CNN, the two TCN-based models converged significantly faster, reaching convergence after approximately 17 epochs versus 35 epochs for the CNN. The CNN also exhibited substantial overfitting with a training loss of 0.16 and validation loss of 0.35.



(a) CNN confusion matrix



(b) Bi-TCN confusion matrix



(c) TCN confusion matrix

Fig. 3. Confusion matrices on test set comparing (a) CNN, (b) Bi-TCN, and (c) TCN model performance across eight gait classes.

The TCN achieved the lowest validation loss (0.21) among all models, indicating strong performance on unseen data despite having a higher training loss (0.29). This pattern suggests the TCN's regularization and dropout effectively prevent overfitting during training. The Bi-TCN achieved the lowest training loss (0.11) with a validation loss of 0.22. While both TCN variants demonstrate effective learning, the causal TCN's ability to achieve comparable validation performance makes it more suitable for real-time deployment where future information is unavailable.

D. Per-Class Performance Analysis

The confusion matrix (Fig. 3) reveals critical performance patterns across the eight gait classes. While the TCN model achieved the highest per-class performance overall, all models consistently struggled with the stand and walk classes. This weakness stems directly from our data labeling process. Our model does not explicitly handle transitions between gait phases, creating ambiguous boundaries when the robot shifts from standing to walking or vice versa. This limitation is further evidenced by the fact that the most common misclassification across all gait modes was the stand class. These transition frames contain mixed feature values that belong to neither pure standing nor pure locomotion, making them inherently difficult to classify under our current discrete labeling scheme.

IV. Conclusion

In the paper, we present a TCN-based classification model for recognizing human gait modes from motion capture data. A sliding window of state history is used to capture the temporal nature of locomotion. The model is trained to recognize 8 gait modes based on 34 derived gait features. Our model is compared to a CNN and a TCN without causal convolution and performs better in all metrics used for our study. Particularly, our TCN achieves 95.46% accuracy. The results of our performance evaluation demonstrate that explicit temporal modeling through TCNs significantly outperforms spatial-only approaches for gait classification, while causal convolutions enable real-time classification without sacrificing accuracy.

References

[1] J. Wang *et al.*, "Integral real-time locomotion mode recognition based on Ga-CNN for Lower Limb Exoskeleton," *Journal of Bionic Engineering*, vol. 19, no. 5, pp. 1359–1373, July 2022, doi: 10.1007/s42235-022-00230-z.

[2] S. Mary Sunny and A. P. Parameswaran, "Intelligent Human Gait Phase Classification Using Machine Learning Models," *IEEE Access*, vol. 13, no. , pp. 142879–142899, 2025, doi: 10.1109/ACCESS.2025.3598038.

[3] Y. Kim *et al.*, "Deep Learning-Based Recognition of Locomotion Mode, Phase, and Phase Progression Using Inertial Measurement Units," *Journal of Bionic Engineering*, vol. 22, p. , 2025, doi: 10.1007/s42235-025-00723-7.

[4] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, "Temporal Convolutional Networks: A Unified Approach to Action Segmentation," *CoRR*, 2016, [Online]. Available: http://arxiv.org/abs/1608.08242

[5] S. Bai, J. Z. Kolter, and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," *CoRR*, 2018, [Online]. Available: http://arxiv.org/abs/1803.01271