2024 33rd IEEE International Conference on
Robot and Human Interactive Communication (RO-MAN)
August 26-30, 2024. Pasadena, California, USA.

Spiking Neural Networks for Improved Robot-Human Handoffs

Nathaniel Gyory[1], Wallace E. Lawson Jr[2], and J. Gregory Trafton[3]

2024 33rd IEEE International Conference on Robot and Human Interactive Communication (ROMAN) | USGov | DOI: 10.1109/RO-MAN60168.2024.10731389

Abstract— **This paper demonstrates the effectiveness of learning based models for accurate, and reliable robot to human handoffs in various HRI scenarios. Specifically we bench marked a neuromorphic spiking neural network and a time series k-nearest neighbors classifier against traditional hand crafted force threshold methods. These models use linear force in the x, y, and z direction, as well as torque about the x, y, and z axis at the end effector of the robot arm to make handoff predictions. This paper demonstrates that these learning based methods are more robust to noise which occurs during operational use. We applied our algorithms to both stationary handoffs (stationary robot) and moving handoffs (robot walking). We believe that our evaluation is the first to examine walking handoffs. We evaluated all models in tests which determined the accuracy, precision, recall, f1, and average execution time for handoff events, noise events, and no event tests. We find that the SLAYER spiking neural network model performed the best across both walking and stationary handoffs for the majority of the evaluation criteria. Our results suggest that neuromorphic spiking neural networks are strong contenders for applications in time series, event based HRI applications.**

I. INTRODUCTION

Handoffs are an essential element of joint human-robot teaming tasks. Handoffs are defined as the point at which the holder of an object releases its grip in order to transfer object control over to the receiver. If an object can not be quickly and effectively shared between a robot and human, then the team will be limited in its ability to complete tasks. With tool/object sharing, the robot-human team can more easily collaborate and effectively complete a desired goal. However, it is not enough for a handoff to simply work well, it must be extremely reliable. Handoffs between human-robot teams must have excellent accuracy, work in various environments, and be resilient to noise. If a handoff model is not able to perform up to these standards, then the robot will become a liability and hurt joint collaboration efforts.

Human-robot handoff models are separated into two distinct groups: human to robot (H2R), and robot to human (R2H) [21]. H2R handoffs are classified as handoffs where the object giver is the human and the receiver is a robot. The latter, (R2H) is when the giver is a robot and the receiver is the human. For this body of work, we focus on R2H handoffs.

In order to conduct a R2H handoff, a robot must be aware of changes to its environment so that it can properly determine when to release the object in its possession. Traditionally this is done using various onboard robot sensors. Force sensors are commonly used on robot arms to determine when an object is being influenced by an outside force. Thresholding techniques use these force sensor readings to determine when a human receiver has grasped the object in the robot's possession [6], [25], [19], [1], [15], [22], [16], [8], [13], [20]. In these papers, the authors choose a user specified threshold λ which describes the force magnitude required to trigger an object release. If the force at the end effector is $< \lambda$, the robot will continue to hold the object. If the force at the end effector is $\geq \lambda$ then the robot will release the object. While some thresholding methods utilize the total force magnitude [6], others analyze the force which is exerted on specific axes. In a paper from Parastegari et al. [22], the authors use thresholds along specific axes which correspond to the predicted pull directions from a receiver. The authors observed that these axes captured the majority of the forces from the human grasp event, and provided a more targeted threshold to trigger a release.

While the thresholding heuristic is straightforward, easy to implement, and effective in controlled environments, it unfortunately suffers from a number of problems. The first is that it requires a hand picked threshold. Depending on the specific robot arm, environment, and the tool that it is holding, a user must choose a different threshold value for all of these scenarios. Other authors attempt to remove the need to choose a user specified threshold value by using a perceived load force value [11] [17] [24] [5]. The perceived load force will change when a human receiver grabs the handover object to initiate a handoff. However, neither perceived load force nor thresholding models learn the underlying characteristic force changes which classify a handoff; they instead create a coarse grained judgement on what high forces/reduced load might mean when sensed at the end effector. If the robot arm is bumped or the robot is placed on a moving platform (elevator, ship, etc), external forces could potentially trigger false handoff events.

Similar to force threshold methods, some researchers use the spatial displacement of the robot arm end effector to determine if a handoff has occurred [4] [3]. For example, one method chose a displacement of 1 cm to signal a handoff. While this may work well in controlled environments, if the robot arm is accidentally bumped then it will drop the object which it is carrying. This is problematic because these models would not work in real world scenarios, outside of controlled environment testing.

Other R2H handover models rely on the use of computer vision in order to determine when a human has grasped the

The following authors are with the U.S. Naval Research Laboratory, Human and Machine Intelligence Group, NRL, 4555 Overlook Ave SW, Washington DC 20375.
[1] Nathaniel Gyory nathaniel.p.gyory.civ@us.navy.mil
[2] Wallace Lawson wallace.e.lawson2.civ@us.navy.mil
[3] Gregory Trafton greg.j.trafton.civ@us.navy.mil

object. Work by Aleotti et al. [2] uses a kinect camera to generate a point cloud of the scene. When a hand enters the scene and the tool's point cloud makes contact with the hand point cloud, the robot will then release the object. Other researchers use hand gesture recognition to classify hand positions which indicate a receiver has grasped the object [18]. While this may work if the human participant is paying full attention, it may be problematic if the user does not have a firm grip on the object. Since the robot does not record force data it must make a guess based on the point clouds to determine if the receiver is in control of the object. For this reason, using only computer vision methods for R2H handoffs presents problems for achieving consistent and reliable R2H handoffs.

Authors Grigore et al. [12] use both force sensor data and computer vision in order to determine when a robot is ready to release its object. These models determine when a human is in the frame, if they are looking at the object, and if there is force exerted on the end effector. These joint models help reduce the chance that noise from the environment create false handoff events. Since the model confirms a person is in the frame, they are looking at the object, and there is force acting upon the arm, the model is more confident that a handoff event is occurring. However the model has not actually learned the underlying characteristic force changes which depict a human is in control of the handover object. It is simply making assumptions based upon related observations. If a person bumps the robot arm while looking at the object, all requirements for a handoff would occur and the object would incorrectly be dropped.

Given the possible concerns highlighted above, we examined the feasibility of learning based methods for classifying handoff events. We were inspired by work from Shrestha et al. [23] who developed a neuromorphic deep learning framework called SLAYER. The authors demonstrated that their SLAYER framework excelled at classifying time series, event based data. Since force readings from handoffs consist of time series events, a SLAYER based deep learning model should be an excellent fit. In addition to developing a neuromorphic model, we also examined how well a popular learning based method, k-nearest-neighbors, would perform. K-nearest neighbors is an excellent baseline classifier to compare the SLAYER model to because it is a well known and widely used model [33]. Several adjustments to the model (described below) allow KNN's to perform well, specifically with time series data [26].

In addition to exploring new learning based models for handoff classification, we also wanted to explore stationary vs walking handoffs. To the best of our knowledge, there are no handoff models for moving mobile robots; and all current handoff research is conducted with stationary manipulators [6], [25], [19], [1], [15], [22], [16], [8], [13], [20]. There is research by Tulbure et al. [29] where the authors use a quadruped robot for H2R handoffs, however, the quadruped is only used to navigate to the object giver. The handoff is then completed in a fixed stationary position.

Walking handoffs are a normal part of our everyday lives and it is important for robot's in human-robot teams to achieve a high level of accuracy and reliability for this task. Compared to stationary handoffs, walking handoffs provide us with a new challenge when developing handoff detection models. Walking handoffs generate large amounts of noise in the force readings at the end effector. This problem is accentuated by our use of a quadruped robot, where each step creates a large spike in force values. There exist other learning based handoff detection methods [24] but these models only test stationary handoffs which limit their generalization to real world handoff scenarios. By applying our handoff classification models to walking handoffs, we will be able to test the models resilience to increased ambient force reading noise.

II. CONTRIBUTIONS

The main contributions of this paper are as follows:

1) Introduction of a neuromorphic spiking neural networks to robot-human handoffs which outperforms the current thresholding techniques.
2) Benchmarking of the neuromorphic model against other learning based techniques (k-nearest neighbors classifier)
3) Demonstrating how learning based models can be successfully applied to both stationary and walking handoffs.

III. METHOD

A. Robot Platform

To test the effectiveness of various R2H handoff algorithms, we used the quadruped Boston Dynamics Spot Robot [10] equipped with a Spot Arm that we have named Bight. With the addition of the Spot Arm, Bight can manipulate objects, gesture, and engage in a broader range of tasks.

In order to conduct R2H handoffs, Bight needs to sense the forces exerted at the end effector of its arm. This data is made available through the Boston Dynamics Spot SDK using the motor impedance feedback command [10]. These force readings consist of linear forces in the x, y, and z direction, along with torque about the x, y, and z axis. Using the Spot Arm, force readings can be captured every 6 ms. Previous research has suggested that people greatly prefer handoffs that occur < 474 ms [8]; thus we capture 60 time steps worth of force readings, resulting in a 360 ms prediction window.

B. Types of Handoffs

Since Bight carries the arm on its back, we are able to conduct both stationary and walking handoffs. Stationary handoffs are common in robot-human teams and have been the focus of research for the various R2H models. However, to the best of our knowledge, there are no existing models that perform walking handoffs. Walking handoffs occur frequently in everyday situations. For example, it is not uncommon to hand someone their phone while walking or a nurse to hand a doctor medicine or a clipboard. Walking handoffs are a crucial element of everyday life and, in order for human-robot teams to reach parity with human-human teams, walking handoff must be addressed.

1) Stationary Handoffs: For stationary handoff experiments we instructed Bight to fetch a tool and initiate a handoff towards a human receiver. The handoff movement consisted of Bight extending its arm towards the receiver at approximately waist height (1 meter). Once the arm completed its intended trajectory, an LED light on the arm turned on to indicate that the robot was ready to transfer the handover object (approximately 1 second after arm extension). At this point the arm began recording force data exerted on the end effector. The receiver then grasped the object to begin the object transfer process. During the handoff stage, force data was continuously fed to one of the three handoff detection algorithms. If the algorithm determined a handoff had occurred based on the input force readings, the algorithm would signal to the arm to open the gripper and release the tool. Figure 1 displays the forces at the end effector of the Spot Arm during a stationary handoff.

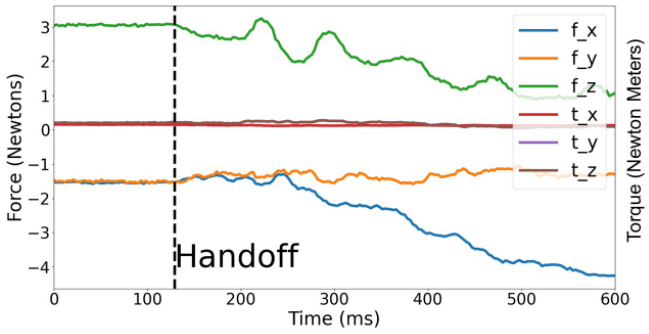

Fig. 1: Stationary handoff force data. The dotted black line indicates where the handoff occurs. Note: f_x, f_y, and f_z are in Newtons whereas t_x, t_y, and t_z are in Newton-Meters.

2) Walking Handoffs: For the walking experiments, Bight began by picking up the tool and walking towards a human receiver with the object extended. The human receiver was situated just off to the side so that they could grab the object as Bight walked by. The arm continuously read force readings into the detection algorithm while Bight walked along a forward trajectory. If the algorithm determined a handoff had occurred based on the force readings, the algorithm would signal to the arm to open the gripper and release the tool. Figure 2 displays the forces at the end effector of the Spot Arm during a walking handoff. Unlike the stationary handoff depicted in Figure 1, there is no clear visual indication when the handoff occurs. This is due to the fact that the forces generated by a handoff are minuscule compared to the forces exerted on the end effector from locomotion. This presents a substantial challenge, specifically for thresholding techniques, to correctly classify walking handoff events. Pictures depicting the stationary and walking handoff experiments are displayed in Figure 3.

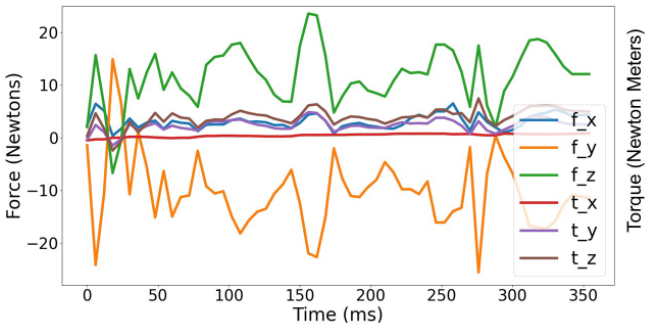

Fig. 2: Walking handoff force data. Note: f_x, f_y, and f_z are in Newtons whereas t_x, t_y, and t_z are in Newton-Meters.

Fig. 3: Stationary and walking handoff experiments.

C. Algorithms

For both walking and stationary experiments we tested three different handoff detection algorithms. These consisted of a k-nearest neighbor (KNN) classifier using dynamic time warping (DTW) from the TSLearn library [27], a neuromorphic learning model using the SLAYER framework [23], and a force thresholding algorithm inspired by previous work in the R2H literature [6] [25]. The KNN and neuromorphic approaches represent learning based models while the threshold algorithm relies on handcrafted user parameters. By comparing these various algorithms for both walking and stationary scenarios we intend to determine the method which performs the best in terms of various quantitative performance metrics.

1) SLAYER: Neuromorphic computing aims to replicate how the brain stores, and processes data to more efficiently implement various machine learning models. Some neuromorphic models use spiking neural networks which have been shown to perform well on time series data such as event cameras [23]. A traditional deep neural network uses real number inputs, weights, biases, and activation functions to propagate information through the network. A spiking neural network differs in that it encodes real number input values into 1 bit temporal spiking events. These spiking events are fed into a neuromorphic spiking neuron. When spikes enter the neuron, its membrane potential increases. If it increases above the threshold of excitation, the neuron outputs a spike to the next layer. Since spikes enter at irregular timesteps, a neuron's behavior is dictated by the density and frequency

of the input spikes. Each neuron is modeled after a human neuron and have various hyperparameters which determine how and when they propagate spiking activity throughout the network. Since force readings from the robot arm are read as time series continuous events, these force readings can be easily integrated into a spiking neural network.

The big advantage of neuromorphic computing is improved power efficiency [30]. Neuromorphic hardware can both process and store data together, where as classical computing separates these processes, resulting in extra fetching and storing steps. While neuromorphic SNN's offer improvements to power efficiency, they do not perform as well as state of the art artificial neural networks for complex classification tasks [30]. For this reason, it is important to keep in mind the complexity of the classification problem when utilizing an SNN.

2) Model Architecture: For our neuromorphic model we used Intel's SLAYER network which is built on top of their Lava framework [23]. For the network we used 4 layers, consisting of an input layer, two dense layers, and a final affine layer for classification. The input layer encodes the 6 force features into a spike train so that the spiking neural network can process the data. This is done using SLAYER's CUBA layer, a second-order variant of the classical LIF neuron model with improved neural dynamics. The encoder operates by adding the input values constantly over time to a CUBA neuron, which, in turn, generates spikes [14].

$$\text{Input Encoding Layer: } \mathbb{R}^{60\times 6} \rightarrow \{0,1\}^t \tag{1}$$

After the input layer encodes the force data, the spike train is passed through two dense layers. These layers combine a dense feed-forward neural network with CUBA dynamics. These dense layers have 32 neuron inputs and outputs [14].

$$\text{Dense Spiking Layer: } \{0,1\}^t \rightarrow \{0,1\}^t \tag{2}$$

For the output we take the membrane potential of the neuron in the affine layer which generates a real-value number. The affine layer acts as a decoder to translate the final spikes into a real number confidence prediction [14].

$$\text{Affine Decoder Layer: } \{0,1\}^t \rightarrow \mathbb{R} \tag{3}$$

3) Parameter Tuning: In order to compute the spikes in the network, a SNN employs a neuron model to determine how and when to fire. Table I displays the parameters used for the leaky integrate and fire neuron model we used for our network. These parameters can be tuned in order to change the behavior of the network to fit the desired goal.

The final parameter for the SLAYER model is the model confidence threshold. When the SLAYER model evaluates force readings, it outputs a confidence score between 0 and 1. A user must specify a confidence score threshold to determine when a handoff event has or has not occurred. For our tests we used a confidence score of .7 for stationary handoffs and .9 for walking handoffs. The reason why the walking confidence was much higher was due to the increased noise during the handoff. In order to not trigger false positives we increased the level of model confidence to account for this factor. A confidence of .9 works quite well for stationary handoffs, though not tested here.

TABLE I: Neuron Model Parameters

Threshold	0.1
Current Decay	0.9
Voltage Decay	0.9
Tau Grad	1
Scale Grad	1
Scale	$1 \ll 6$

D. KNN

In order to accurately benchmark the performance of the SLAYER model, we compared it to another learning based method, the k-nearest neighbors classifier. The KNN classifier is a supervised learning model which predicts the input's class based upon the classes of the K closest points in the training set. While KNN classifiers traditionally use euclidean distance as the distance metric, we are working with time series data. As shown by Yesilli et al. [31] dynamic time warping works exceptionally well as a KNN distance metric for classifying time series data. For our experiment, we trained a KNN classifier from the TSLearn framework [27] with a K size of 5 and the distance metric set to DTW.

1) Parameter Tuning: For a KNN the two parameters which a user can tune are the K value and the distance metric. The K value is the closest K neighbors to the query point. The majority class from the resulting K points is the predicted class for the model output. If this value is too low then the model could make incorrect prediction's due to noise or outlier points. If the value is too large then it would be more resilient to noise, but have slower computation time. These factors should be taken into account when choosing a value for K. If accuracy is the goal, techniques like cross-validation can be used on a validation training dataset to find the optimal value for K [26]. Along with a value for K, we can also specify what we want to use for a distance metric. Common distance metrics for time series KNN classifiers are dynamic time warping [28], soft dynamic time warping [9], canonical time warping [34], and symbolic aggregate approximation [32].

E. Thresholding

Traditionally, robot-human handoffs utilized simple threshold based approaches to detect when a handoff occurs. Such algorithms work by setting a user specified threshold (λ). If the forces exerted at the end effector (σ) are less than or equal to the threshold, $\sigma <= \lambda$, the robot arm does not recognize a grab event and continues to hold onto the handover object. However, if the forces at the end effector exceed the threshold, $\sigma > \lambda$, then the robot arm releases the object. This is a simple heuristic that works well under controlled environments, but it can run into a number of difficulties. If the threshold is set too low, then the robot

will prematurely drop the handover object. If the threshold is set too high, then the robot may not release the handover object, even if the receiver exerts a lot of force in the handoff.

In order to test the threshold method and compare it to the SLAYER and KNN learning based approaches, we set a user configurable threshold value for linear forces in the x direction. Once the magnitude of 5 consecutive forces readings from the robot arm end effector surpassed the threshold, the robot arm released the handover object.

1) Parameter Tuning: As discussed in the prior sections, the thresholding model works by choosing a user specified threshold λ. This value needs to be handpicked and tuned for the specific environment and type of handoff being executed. It can be quite time consuming when choosing a good value for λ and must be constantly changed to fit the situation. For our testing we used 1 threshold for stationary handoffs (0.0 Newtons). and 3 different values for walking handoffs (5, 10, and 16 Newtons).

F. Model Training

In order to fit/train the KNN and SLAYER models we needed to collect labeled training data. For both models we collected 120 stationary force samples. Out of the 120 samples 60 were positive handoff force readings while the other 60 were negative handoff readings. For the 60 negative handoff reading, 30 were "no event" samples where the arm was not touched and only the ambient forces at the arm end effector were captured. The remaining 30 samples consisted of "noise events" where the arm was bumped from different angles to simulate collisions with outside objects. Handoff events were given a label 1 and negative handoff events were given the label 0. We collected another 120 force samples for the walking models using the same handoff, "no event", and "noise event" breakdown.

Since the SLAYER model required a training loop we trained the SLAYER network for 20 epochs using the AdamW optimizer with a MSE loss function.

IV. Experiments

A. Experiment Design

In order to evaluate our models for both stationary and walking handoffs we conducted a number of experiments. The first was to attempt 10 handoffs for each model. This would allow us to test true positives, false positives, and false negatives for the various models. Next we performed 10 noise tests. During the noise test, a handoff was initiated and the robot arm was bumped from multiple angles to simulate noise which a robot may experience in real life scenarios. This data collection allowed us to determine the number of true negatives and false positives for the models. Finally, we ran 10 "no event" tests, where a handoff was initiated but there was no handoff for 15 seconds. No external forces were exerted on the robot for this test. Similar to the noise test, the "no event" test determined the number of true negatives and false positives. These three different types of tests were repeated for the walking handoff evaluation. For the walking handoff tests, a human receiver stood off to the side while Bight passed with an object extended for a handoff. The receiver would then grab the object as bight passed to complete the handoff. A single operator was used throughout all training and testing.

B. Evaluation Criteria

1) Quantitative Analysis: In order to evaluate the effectiveness of the proposed methods we compared various quantitative performance metrics. For quantitative metrics we used accuracy, precision, recall, f1 score, and tool handover time [21] to benchmark the models. If we let TP be the number of true positives, TN the number of true negatives, FP the number of false positives, and FN the number of false negatives. we can calculate precision, recall, f1 score, and accuracy as [7]:

$$\text{Precision (P)} = \frac{TP}{TP + FP}$$

$$\text{Recall (R)} = \frac{TP}{TP + FN}$$

$$\text{F1 Score} = \frac{2 * P * R}{P + R}$$

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

Along with precision, recall, f1, and accuracy, we used tool handover time to benchmark the speed at which a model could evaluate a handover. Since it was critical to achieve a sub 474 ms handover time [8], execution time demonstrates a model's ability to match natural and desired handover speed.

C. Stationary Handoffs

As shown in Table II the SLAYER model outperforms all other methods in accuracy, precision, recall, and f1 score. The thresholding model had the fastest average execution time, but had the lowest accuracy. Since the SLAYER model was only 4 ms slower than thresholding and performed the best across all other metrics, it proved to be the most effective model for stationary handoffs. Along with reliability metrics, all models executed within the 474 ms acceptable handoff time [8]. A box plot depicting the execution times can be seen in Figure 4.

TABLE II: Stationary Handoff Results

	Accuracy	Precision	Recall	F1	Avg Time (ms)
SLAYER	**0.97**	**0.91**	1.0	**0.953**	59.8
KNN	0.9	0.77	1.0	0.87	314.36
Threshold	0.833	0.64	1.0	0.78	**56.067**

D. Walking Handoffs

As shown in Table III the SLAYER model has excellent performance in every available metric. The SLAYER model is also substantially faster than the 474 ms goal suggested by [8] as shown in Figure 5. The KNN model also performed quite well: it had excellent accuracy and was also faster than 474 ms. Not surprisingly, all the threshold models had concerns. The low threshold model dropped the tool 100%

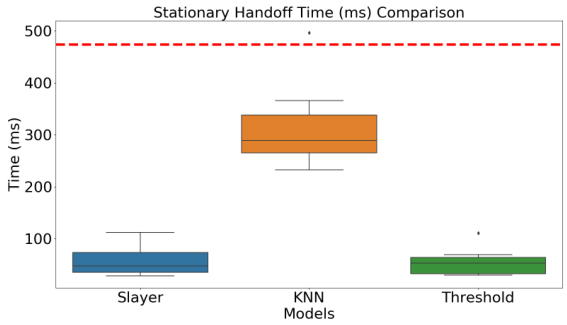

Fig. 4: Stationary handoff execution time. The dashed red line depicts the upper bounds for an acceptable handoff time (474 ms).

of the time as the robot was approaching the person. The high threshold had the opposite problem: holding onto the tool for too long, requiring a substantial grab before the robot released it. The medium threshold was somewhere in between, dropping it more often than it should, but releasing the tool when grabbed. We note that we tried many different thresholds for low, medium, and high, and we could not find a threshold that did not drop the tool or not release it while the robot was walking.

TABLE III: Walking Handoff Results

	Accuracy	Precision	Recall	F1	Avg Time (ms)
SLAYER	**1.0**	**1.0**	1.0	**1.0**	**85.46**
KNN	0.9	0.77	1.0	0.87	396.94
Threshold Low	0	0	0	0	N/A
Threshold Mid	0.2	0.2	1.0	0.33	224.88
Threshold High	0.833	0.667	1.0	0.8	809

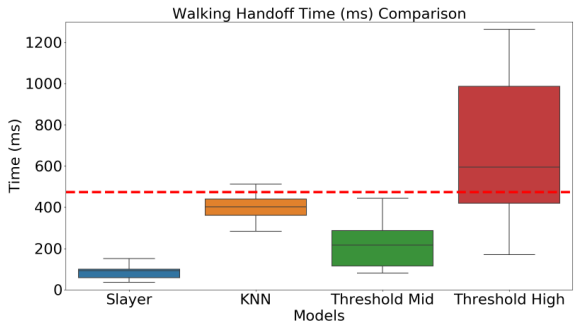

Fig. 5: Walking handoff execution time. The dashed red line depicts the upper bounds for an acceptable handoff time (474 ms).

V. Discussion

When looking at the quantitative results from both the stationary and walking handoffs, we can see that the learning methods outperform thresholding. While thresholding had the lowest execution time for the stationary handoffs, it achieved these results at the cost of accuracy and reliability. The stationary thresholding method produced more false positives during the noise tests due to the noise triggering the force release condition. Since the learning based methods were trained to differentiate between outside forces and handoff forces, they did a much better job at accurately classifying these events. Videos demonstrating these cases can be found in the supplemental video section.

While the thresholding model performed acceptably for stationary handoffs, it performed very poorly for walking. When there was noise from walking, it was very difficult to choose a good handcrafted value for the threshold. Even with a good value, the accuracy was very poor. If the threshold was too high then the accuracy improved but the handoff took too long to execute. The reason for this was the handoff required a very long and hard pull to trigger the threshold. This was not only awkward but a very poor handoff experience. If the value was too low then the threshold would trigger immediately and the robot would drop the tool despite there being no handoff. An intermediate value threshold performs slightly better than the low threshold, but still suffers in a number of ways. It will often have false positive handoff predictions and does not conduct handoffs as quickly as the SLAYER method. Videos demonstrating these cases can be found in the supplemental video section.

For the evaluation of the models we implemented the same pull direction for both walking and stationary handoffs. In the future we would like to generalize our model to account for all realistic pull directions which may arise in a handoff scenario. Similarly, we only evaluated the models on one tool. For future iterations we would like to generalize the handoff model to account for different tool weights and inertial forces due to differing dimensions. Finally, we were not able to power profile the SLAYER model on neuromorphic hardware. With neuromorphic hardware we would be able to benchmark the power savings that the neuromorphic SNN provides compared to the other models running on GPUs.

VI. Conclusions

In this paper we compared the effectiveness of three different models for performing stationary and walking handoffs. We found that the two learning based models (KNN and SLAYER) greatly outperformed thresholding in accuracy. While there were some execution time advantages with thresholding for stationary handoffs, they were minimal. The learning based approaches really excelled when utilized for walking handoffs. Since the KNN, and SLAYER models learned what forces constituted a handoff, they proved to be much more resilient to the noise exerted on the end effector during walking. Our findings demonstrate the advantages of learning based approaches compared to the traditionally used thresholding techniques for robot to human handoffs. We also show that neuromorphic spiking neural networks excel at fast accurate classification tasks on time series data. Our findings suggest that spiking neural networks can also be applied to other areas of HRI which use time series sensor data.

REFERENCES

[1] Henny Admoni, Anca Dragan, Siddhartha S. Srinivasa, and Brian Scassellati. Deliberate delays during robot-to-human handovers improve compliance with gaze communication. In *2014 9th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 49–56, 2014.

[2] Jacopo Aleotti, Vincenzo Micelli, and Stefano Caselli. Comfortable robot to human object hand-over. In *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*, pages 771–776, 2012.

[3] Jonathan Bohren, Radu Bogdan Rusu, E. Gil Jones, Eitan Marder-Eppstein, Caroline Pantofaru, Melonee Wise, Lorenz Mösenlechner, Wim Meeussen, and Stefan Holzer. Towards autonomous robotic butlers: Lessons learned with the pr2. In *2011 IEEE International Conference on Robotics and Automation*, pages 5568–5575, 2011.

[4] Maya Cakmak, Siddhartha S. Srinivasa, Min Kyung Lee, Jodi Forlizzi, and Sara Kiesler. Human preferences for robot-human hand-over configurations. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1986–1993, 2011.

[5] Wesley P. Chan, Iori Kumagai, Shunichi Nozawa, Yohei Kakiuchi, Kei Okada, and Masayuki Inaba. Implementation of a robot-human object handover controller on a compliant underactuated hand using joint position error measurements for grip force and load force estimations. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1190–1195, 2014.

[6] Young Sang Choi, Tiffany Chen, Advait Jain, Cressel Anderson, Jonathan D. Glass, and Charles C. Kemp. Hand it over or set it down: A user study of object delivery with an assistive mobile manipulator. In *RO-MAN 2009 - The 18th IEEE International Symposium on Robot and Human Interactive Communication*, pages 736–743, 2009.

[7] Peter Christen, David J. Hand, and Nishadi Kirielle. A review of the f-measure: Its history, properties, criticism, and alternatives. *ACM Comput. Surv.*, 56(3), oct 2023.

[8] Marco Controzzi, Harmeet Singh, Francesca Cini, Torquato Cecchini, Alan Wing, and Christian Cipriani. Humans adjust their grip force when passing an object according to the observed speed of the partner's reaching out movement. *Experimental Brain Research*, 236(12):3363–3377, 2018.

[9] Marco Cuturi and Mathieu Blondel. Soft-dtw: a differentiable loss function for time-series, 2018.

[10] Boston Dynamics.

[11] A. Gómez Eguíluz, I. Rañó, S. A. Coleman, and T. M. McGinnity. Reliable object handover through tactile force sensing and effort control in the shadow robot hand. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 372–377, 2017.

[12] Elena Corina Grigore, Kerstin Eder, Anthony G. Pipe, Chris Melhuish, and Ute Leonards. Joint action understanding improves robot-to-human object handover. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4622–4629, 2013.

[13] Zhao Han and Holly Yanco. The effects of proactive release behaviors during human-robot handovers. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 440–448, 2019.

[14] Alexander Henkes. Xor: A minimalistic regression tutorial for lava-dl.

[15] Chien-Ming Huang, Maya Cakmak, and Bilge Mutlu. Adaptive coordination strategies for human-robot handovers. 07 2015.

[16] Andras Kupcsik, David Hsu, and Wee Sun Lee. Learning dynamic robot-to-human object handover from human feedback, 2016.

[17] José R. Medina, Felix Duvallet, Murali Karnam, and Aude Billard. A human-inspired controller for fluid human-robot handovers. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 324–331, 2016.

[18] Muhammad Akmal Bin Mohammed Zaffir and Takahiro Wada. Presentation of robot-intended handover position using vibrotactile interface during robot-to-human handover task. In *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '24, page 492–500, New York, NY, USA, 2024. Association for Computing Machinery.

[19] A. Jung Moon, Daniel M. Troniak, Brian Gleeson, Matthew K.X.J. Pan, Minhua Zheng, Benjamin A. Blumer, Karon MacLean, and Elizabeth A.t Crof. Meet me where i'm gazing: How shared attention gaze affects human-robot handover timing. In *2014 9th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 334–341, 2014.

[20] Valerio Ortenzi, Francesca Cini, Tommaso Pardi, Naresh Marturi, Rustam Stolkin, Peter Corke, and Marco Controzzi. The grasp strategy of a robot passer influences performance and quality of the robot-human object handover. *Frontiers in Robotics and AI*, 7, 2020.

[21] Valerio Ortenzi, Akansel Cosgun, Tommaso Pardi, Wesley Chan, Elizabeth Croft, and Dana Kulic. Object handovers: a review for robotics, 2022.

[22] Sina Parastegari, Ehsan Noohi, Bahareh Abbasi, and Miloš Žefran. A fail-safe object handover controller. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2003–2008, 2016.

[23] Sumit Bam Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time, 2018.

[24] Antonis Sidiropoulos, Efi Psomopoulou, and Zoe Doulgeri. A human inspired handover policy using gaussian mixture models and haptic cues. *Autonomous Robots*, 43(6):1327–1342, 2019.

[25] Emrah Akin Sisbot, Luis F. Marin-Urias, Xavier Broquère, Daniel Sidobre, and Rachid Alami. Synthesizing robot motions adapted to human presence. *International Journal of Social Robotics*, 2(3):329–343, 2010.

[26] Samya Tajmouati, Bouazza El Wahbi, Adel Bedoui, Abdallah Abarda, and Mohamed Dakkoun. Applying k-nearest neighbors to time series forecasting : two new approaches, 2021.

[27] Romain Tavenard, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Rußwurm, Kushal Kolar, and Eli Woods. Tslearn, a machine learning toolkit for time series data. *Journal of Machine Learning Research*, 21(118):1–6, 2020.

[28] Tuan Minh Tran, Xuan-May Thi Le, Hien T. Nguyen, and Van-Nam Huynh. A novel non-parametric method for time series classification based on k-nearest neighbors and dynamic time warping barycenter averaging. *Engineering Applications of Artificial Intelligence*, 78:173–185, 2019.

[29] Andreea Tulbure, Firas Abi-Farraj, and Marco Hutter. Fast perception for human-robot handovers with legged manipulators. In *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '24, page 734–742, New York, NY, USA, 2024. Association for Computing Machinery.

[30] Kashu Yamazaki, Viet-Khoa Vo-Ho, Darshan Bulsara, and Ngan Le. Spiking neural networks and their applications: A review. *Brain Sciences*, 12(7), 2022.

[31] Melih C. Yesilli, Firas A. Khasawneh, and Andreas Otto. Chatter detection in turning using machine learning and similarity measures of time series via dynamic time warping. *Journal of Manufacturing Processes*, 77:190–206, 2022.

[32] Yufeng Yu, Yuelong Zhu, Dingsheng Wan, Qun Zhao, and Huan Liu. A novel trend symbolic aggregate approximation for time series. *CoRR*, abs/1905.00421, 2019.

[33] Shichao Zhang, Xuelong Li, Ming Zong, Xiaofeng Zhu, and Ruili Wang. Efficient knn classification with different numbers of nearest neighbors. *IEEE Transactions on Neural Networks and Learning Systems*, 29(5):1774–1785, 2018.

[34] Feng Zhou and Fernando Torre. Canonical time warping for alignment of human behavior. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009.