DRAFT

Proceedings of the ASME 2021 17th International Manufacturing Science and Engineering Conference
MSEC2022
June 27-July 1, 2022, West Lafayette, Indiana, USA

MSEC2022-83907

EVALUATION OF DATA-DRIVEN MODELS IN HUMAN-ROBOT LOAD-SHARING

Vinh Nguyen
National Institute of Standards and Technology
Gaithersburg, MD

Jeremy Marvel
National Institute of Standards and Technology
Gaithersburg, MD

ABSTRACT

Human-robot load-sharing is a potential application for human-robot collaborative systems in production environments. However, knowledge of the appropriate data-driven models for this application type is limited due to a lack of physical real-world data and validation metrics. This paper describes and demonstrates a load-sharing testbed for evaluating data-driven models in a human-robot load-sharing application. Specifically, the testbed consists of a single operator and single robot relocating a payload to a desired destination. In this work, the operator initially communicates to the robot using audio feedback to initiate and alter robotic motion commands. During the payload relocation, human, payload, and robot state data are recorded. The measurements are then used to train three data-driven models (neural network, naïve Bayes, and random forest). The data-driven models are then used to transmit movement commands to the robot during human-robot load-sharing without the use of audio feedback, thus improving robustness and eliminating audio signal processing time. Evaluation of the three data-driven models shows that the random forest model was demonstrated to be the most accurate model followed by naïve Bayes and then the neural network. Hence, the results of this study provide novel insight into the types of data-driven models that can be used in load-sharing applications in addition to development of a real-world testbed.

Keywords: Human-Robot Interaction, Manufacturing, Load-Sharing, Collaborative Robot, Wearable Electronics

1. INTRODUCTION

Human-robot collaborative systems are seeing increasing adoption in manufacturing applications due to their flexibility and efficiency. In particular, load-sharing applications are studied by researchers in fields including production [1] and health [2]. In this work, human-robot load-sharing refers to a system where at least one human and one independent robot coordinate to relocate a payload. In this application of human-robot collaboration, the robot must have both perception and cognitive capabilities to ensure accurate and safe handling of the payload.

Exclusive monitoring of the robot state, such as the joint angles and current sensors, has been used for the robot to learn the optimal configuration for assisted lifting [3]. However, human-robot load-sharing applications generally tend to require the robot to sense the payload or state of the human. As such, multiple prior sensing technologies have been developed to augment robots for this task. For instance, DelPreto and Rus [4] used surface electromyography in a manufacturing-based load-sharing system to sense the operator intent based on muscle signals in the human arm. In addition, van der Spaa et al. [5] used data from an entire motion capture suit to train a graph search algorithm to determine optimal robot motion. Another use of motion capture for robot perception in human-robot load-sharing involved tracking the robot hand, human hand, and payload pose [6]. However, in a realistic production environment, the robot, operator, and payload information must all be measured by a variety of sensor types to improve robustness against sensor interference that is known to occur in production environments [7]. In addition, cognitive models must be developed based on sensor data for determining the appropriate robot commands to ensure safe and accurate execution of human-robot load-sharing.

Though teleoperated human-robot load-sharing applications have been demonstrated [8], automated robotic motions that require cognitive models are more desirable due to reduced manual variation and lag. Therefore, human-robot load-sharing is a use-case of interest for researchers to develop robotic cognitive models of human behavior, because the application requires human-robot coordination. While analytical models are known to be used in human-robot collaboration applications such as task allocation [9], data-driven models are known to be more reliable in quantifying stochastic phenomena including

operator intent and sensor noise. Many of the previously mentioned prior literature also developed data-driven models that leveraged their demonstrated sensing capabilities. For instance, Sheng et al. [6] developed a reinforcement learning approach to lift the payload, though only vertical lifting was conducted. In addition, Roveda et al. [10] used a reinforcement learning algorithm based on neural networks to conduct impedance control in human-robot load-sharing. However, reinforcement learning requires multiple trials, which may require incorrect motions that induce hazards in a coordinated task, such as load-sharing, for training. While supervised learning approaches such as neural networks [4] and Gaussian Process Regression [3] have been demonstrated, there is a lack of comparison between the plethora of data-driven models that can be applied towards human-robot load-sharing. Therefore, researchers and manufacturers have limited knowledge on the appropriate data-driven models that can be used in human-robot load-sharing.

This research aims to provide an evaluation of supervised learning data-driven models in human-robot load-sharing. In this work, a single human operator and a single robot coordinate to relocate a payload. To initially train the data-driven models in this work, voice commands by the operator are used to direct the robot motion to relocate the payload while human, payload, and robot state data are recorded. The data are then used to train three individual data-driven models (neural network, naïve Bayes, and random forest) to command robot motions based on measurements without the use of voice commands. Figure 1 shows a flowchart of training and implementation methodology. The results of the performance of the models are then evaluated followed by discussion and conclusion.

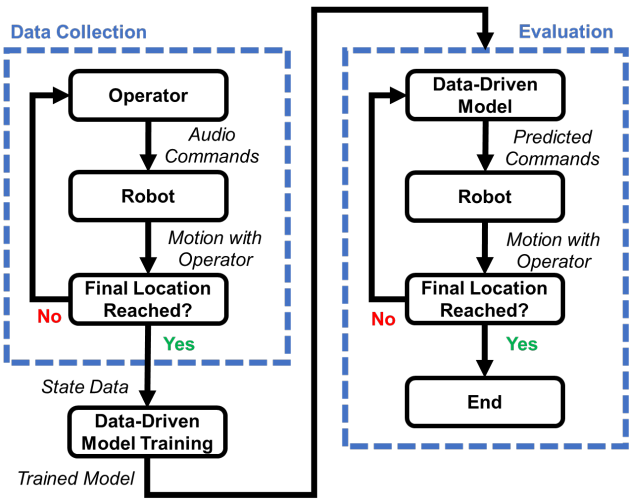

FIGURE 1: FLOWCHART OF TRAINING AND EVALUATION METHODOLOGY

2. TESTBED

Figure 2 shows the experimental testbed used in this work. At the beginning of the experiments, the robot was automatically driven to the start position shown in Fig. 2 and the data collection was started. Then, the operator and the robot were tasked with raising a payload (+Z direction), moving the payload horizontally (+Y direction), and then lowering the payload (-Z direction) onto the end position supports identified in Fig. 2. In this work, the payload was a 200 mm x 700 mm x 1.60 mm aluminum plate (~0.61 kg). For safety purposes, bumpers were mounted around the plate and a soft-stop foot switch was wired to the robot controller.

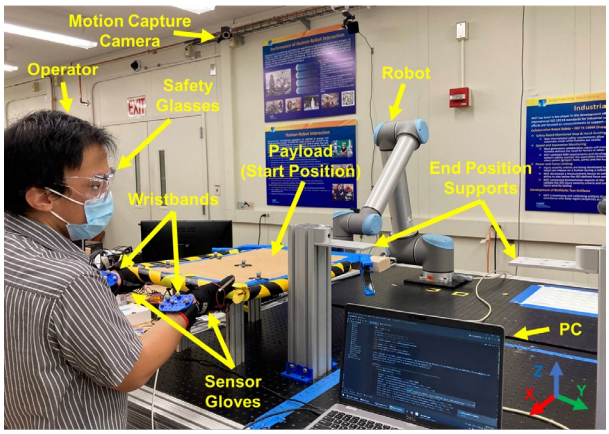

FIGURE 2: EXPERIMENTAL SETUP FOR LOAD-SHARING TESTBED

2.1 Audio Commands for Training

Initially, data for supervised learning were collected by repeatedly conducting the load-sharing task with the operator controlling the robot with voice commands. Figure 3 shows the available voice commands the operator can give to move the robot in a specified direction at 20 mm/sec. In addition, the voice command "Stop" would stop any robot motion. To conduct the voice control, a Python script ran on a personal computer (PC) that used a Bluetooth receiver to read audio signals. The Google Cloud Text-to-Speech Application Programming Interface (API) was used to convert the raw audio signal to text within 2 seconds. The text was then analyzed by the Python script to determine the direction of motion for the robot.

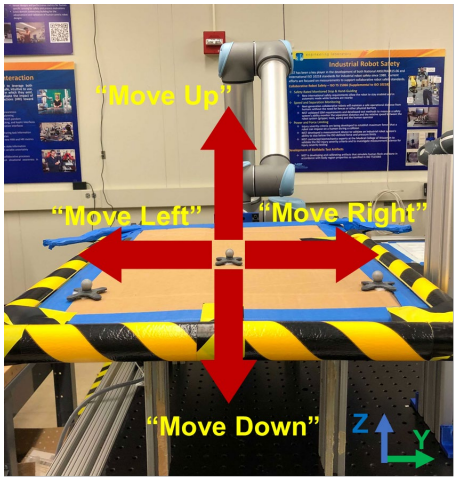

FIGURE 3: VOICE COMMANDS AND THE CORRESPONDING ROBOT MOTION DIRECTION

During the training phase, the operator used voice commands to move the robot in a specified direction, and the operator would move in tandem with the robot while holding the payload. Thus, robot, human motion, and payload data would be recorded throughout the entire move, which would then in turn be used to train the data-driven models in this work.

2.2 Robot and Operator

In this experiment, a Universal Robot UR10 collaborative robot arm with a CB3 control was used to support the payload. The robot was configured to accept velocity commands from the external computer using the User Datagram Protocol (UDP) at a rate of 10 Hz. For safety, the payload was manually clamped to the robot's end effector instead of using a robotic gripper. During the load-sharing process, recorded data from the robot included Cartesian position and velocity.

In this paper, a single operator was used since the focus of this work was the evaluation of data-driven models in a controlled human-robot team. Note that the operator was provided no instruction on the ideal path to move the robot in addition to no instruction regarding body posture. Thus, this research studies variability within a particular human subject but does not generalize the results to a population. In addition, the operator's intuition was used to determine factors including the appropriate vertical direction height before moving in the horizontal direction.

2.3 Position Tracking

To track the position of the payload and operator wearables, an 8-camera motion capture system (Vicon Bonita 10) at 100 Hz was used. The camera system communicated to an external PC using Power-over-Ethernet. The external PC then transmitted tracking data of the desired objects to the main PC using the UDP protocol. Figure 4 shows the objects that were tracked in this work. Specifically, retroreflective motion tracker markers were mounted onto the payload center point, operator wristbands, and safety glasses. For each object, a rigid body was assigned in the motion capture software to the marker arrangement, providing a full 6 degrees-of-freedom pose. Therefore, the Cartesian position and orientation of the objects were tracked and then used for training and control. Note that the operator wristbands had to be positioned such that the markers would face towards the cameras and therefore the testbed with its current camera distribution cannot accommodate the operator flipping their wrist orientation. Alternatively, an Inertial Measurement Unit was explored as an option to obtain hand position information, but this method was determined to be infeasible due to the need for repeated calibration in addition to lag time caused by the required Extended Kalman Filter [11].

2.4 Glove Sensors

For localized hand measurements, operator gloves (Klein Tools 40229) were augmented with electronic sensors as shown in Figure 5. For finger bend measurements, flexible sensors that change resistivity with respect to the degree of bending (Spectra Symbol FS-L-0055-253-ST) were sewn onto the back of the glove. In addition, thin pressure sensors that change resistivity with respect to applied pressure (Interlink Electronics FSR 400) were sewn onto the glove fingertips to measure the contact force. Both the bend and pressure sensors were wired to voltage dividers that were then measured by the Analog-to-Digital Converter of a Teensy 4.0. The Teensy 4.0 then transmitted data to the PC using serial communication at 100 Hz. Note that the gloves also allow transmission using wireless Bluetooth via a Microchip RN-42 Module, though this feature was not used in this work.

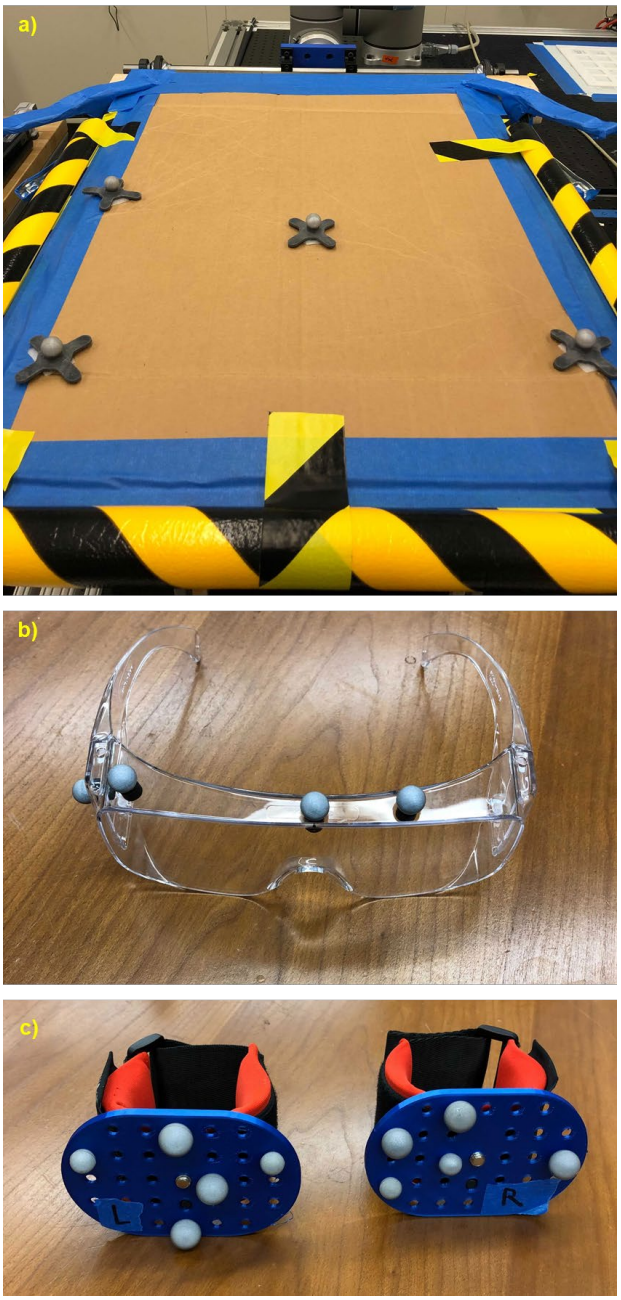

FIGURE 4: a) PAYLOAD, b) SAFETY GLASSES, AND c) WRISTBANDS TRACKED BY THE MOTION CAPTURE SYSTEM

FIGURE 5: a) FRONT AND b) BACK VIEW OF AUGMENTED OPERATOR GLOVES

3. DATA-DRIVEN MODELS

After collecting the data, three individual data-driven classifier models were trained to predict the correct robot motion command based on the sensor measurements. Hence, using these data-driven models, audio feedback would no longer be required for load-sharing, which therefore improves processing speed and reliability. Specifically, neural networks, naïve Bayes, and random forest algorithms were studied. This section briefly summarizes the classifiers in addition to specific parameters used in this work.

3.1 Neural Network

Neural network models have been used in multiple manufacturing applications including tool condition monitoring [12] and part quality inspection [13]. Neural networks consist of multiple neurons with activation functions [14]. These neurons are segregated into an input layer, an output layer, and at least one hidden layer between the input and output layers. The neuron of a subsequent layer can be represented as the activation function of the sum of the previous layer multiplied by calibrated weights. Thus, the final output of a neural network is represented as the activation function of its prior layer multiplied by its weights. Hence, a neural network model can model nonlinear systems due to application of weights and activation functions.

To determine the weight values, this work uses a limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm [15] to minimize a cross entropy loss function. In addition, a single 20 neuron hidden layer was used in this work. The size of the hidden layer was determined by gradually increasing the number of neurons until the classification accuracy no longer showed significant improvement.

3.2 Naïve Bayes

Naïve Bayes models leverage Bayes' theorem under the assumption that every pair of features are conditionally independent given the value of the output [16]. Naïve Bayes models have been demonstrated in fault detection in semiconductor manufacturing [17] and anomaly detection in Industrial Internet of Things systems [18]. In naïve Bayes classifiers, the classification rule can be expressed as

$$\hat{y} = \underset{y}{arg\max}(P(y)\prod_{i=1}^{n} P(x_i|y)) \quad (1)$$

where y is the output and x is the feature. In this work, $P(y)$ is determined using Maximum A Posteriori estimation, while $P(x_i|y)$ is assumed to be Gaussian in this work:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}}\exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right) \quad (2)$$

where μ_y and σ_y are calculated by maximizing the likelihood function. Therefore, since the formulation is straightforward with minimal numerical search algorithms, the training time of the naïve Bayes classifier is expected to be the fastest out of the models studied in this work.

3.3 Random Forest

Random forest models leverage the use of multiple decision trees to create an ensemble classification [19]. Random forest models have been used in manufacturing applications including robot diagnostics [20] and job-shop scheduling [21]. Each decision tree is trained with the bagging method using a random subset of data with replacement for training. During implementation, each of the decision trees (100 in this paper) outputs a classification. The ensemble classification is then determined by a majority vote of the decision trees. Therefore, by using subsets for training each decision tree, errors induced by overfitting can be reduced.

In addition to data bagging, the random forest was also configured to bag a random subset of input features (maximum of 7 features at a time) [22]. This feature bagging reduces correlation among trees caused by strong predictors. By reducing this correlation, the prediction accuracy is less prone to overfitting in addition to ensuring that all features are analyzed properly by the training algorithm.

4. RESULTS

This section describes predominant trends in the load-sharing training data. Then, the models are evaluated by offline cross-validation followed by physical implementation of the trained models without audio feedback.

4.1 Training Data

Seven load-sharing trials with the same participant were conducted to obtain measurements for training the data-driven models. Figure 6 shows representative time series results of the motion tracker data. As expected, the robot's internal position data are relatively smooth because its pose measurements are based on internal encoder data. However, the payload, glasses, and wrist measurements that are based on motion capture marker data are shown to be noisier. For instance, the X, Y, and Z safety glass positions are reported as 0 mm when data are lost by the motion capture cameras due to occlusion or failed camera triangulation. However, this is prone to occur in motion capture applications in a real-world environment, so the data were not filtered in this work to ensure representative measurements. A possible alternative would be to apply a median filter or first-order hold in the event of data loss, at the cost of losing general data fidelity. In addition, the rotations about the X, Y, and Z in the motion tracker data show that rigid bodies represented by motion capture markers are prone to "flipping" due to a combination of noise and symmetry in the motion capture patterns. This is prone to occur in the wrists since the motion capture markers are mounted on a two-dimensional (2D) plane, as shown in Fig. 4c. However, similar to data loss, this can occur in real-world applications and therefore the data were not processed to allow for this effect. Possible compensation methods include 3D mounting of the motion capture markers at the cost of comfort and flexibility in addition to implementing process-based prediction models.

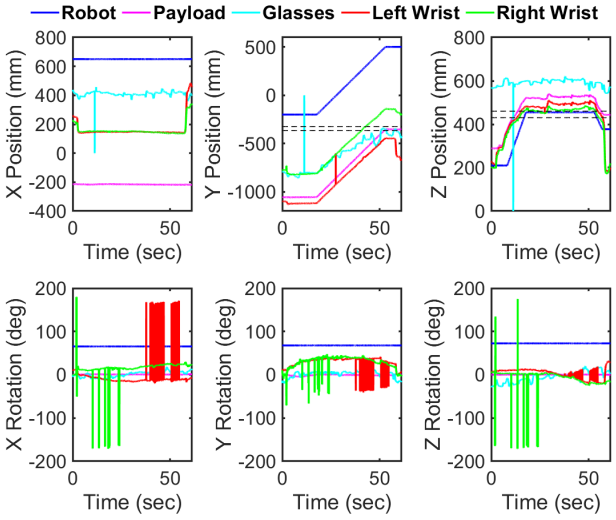

FIGURE 6: REPRESENTATIVE TIME SERIES FOR ROBOT AND TRACKER MEASUREMENTS WITH DASHED LINE REPRESENTING TARGET LOCATION OF PAYLOAD ± 20 MM

Figure 7 shows glove data for the same test shown in Fig. 6. Interestingly, the flex sensors are shown to start and saturate at different levels. For instance, the flex sensors for the index and middle fingers of the left glove vary significantly from the right glove. Therefore, after the operator wears the gloves, the sensors may need calibration. However, this approach is time-consuming in real-world production environments, especially if the wearables are repeatedly removed. Therefore, this work does not conduct sensor calibration and hence also considers variation in the glove sensor measurements when training its data-driven models. In addition, the pressure sensor measurements (bottom of Fig. 7) also exhibit variation between the right and left gloves. However, the pressure sensors can be used as important features to identify the start and stop of the load-sharing process.

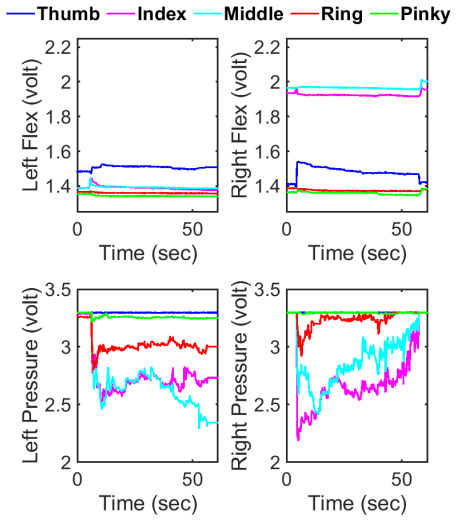

FIGURE 7: REPRESENTATIVE TIME SERIES FOR GLOVE SENSOR MEASUREMENTS

4.2 Evaluation Metrics

To evaluate the performance of the three data-driven models, a series of metrics were defined based on the design of the testbed and the nature of the training datasets. These metrics are roughly categorized as "accuracy" and "efficiency."

Accuracy in this work is an assessment of the correctness of the algorithms' capacity to identify the directional motions of the human operator. Given the time series nature of this application, accuracy is the correctness of the estimate of the operator's intended motion, $\tilde{m}_n$, versus the actual intended motion, m_n, at a given time step, n, over the entire length of the test, N:

$$accuracy = \frac{\sum_{n=0}^{N} q_n}{N}, \text{where } q_n = \begin{cases} 1 & \text{if } \tilde{m}_n = m_n \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

In all instances, incorrect estimates of operator motions introduce strain at the grasp points and may cause internal stresses and forces to occur within the payload. In addition, estimates that are opposite of the operator's motions (e.g., the operator moves the workpiece upward, but the robot attempts to move it downward) can result in injury. In addition, confusion matrices were presented along with accuracy metrics to illustrate the nature and severity of the errors.

Similarly, given the manufacturing application domain, the efficiency of the learning algorithm is measured as a function of the timing of the algorithm's performance. The algorithms that are slower to predict are considered less efficient than those that

take less time. However, efficiency is distinct from the accuracy measure, as it is possible for a given approach to be both fast and incorrect. There are three components of the time factor that are critical for assessing the efficiency of a given algorithm:

1. Training time – How much time and effort are required to advance the performance of an algorithm to the point where it can make estimates of the operator's motions with some level of certainty?
2. Prediction time – How much time is required for an algorithm to take the sensor inputs from the operator's actions and then produce a motion estimate?
3. Correction time – In the event that an initial motion estimate is incorrect, how much time lapses before the correct motion is identified?

In this work, the results from only the first two categories are presented. However, it should be noted that longer correction times tend to directly correlate to unnecessary increases in task times and may also result in task performance degradation, damage to equipment, and operator injury.

4.3 Cross-Validation

5-fold cross-validation was conducted to evaluate the data-driven models before conducting physical evaluation. Table 1 shows the model accuracy results of the cross-validation. The random forest model is shown to exhibit the highest accuracy at 97.31%, while the neural network is shown to have the lowest accuracy at 72.12%. This is because the features in this work vary in their correlation with the predicted motion output and their level of noise. For instance, the X position data of all the measurements shown in Fig. 8 is relatively constant compared to the Y position data because the payload only moves in the Y direction. Therefore, the X position data would not be as correlated with predicting the operator voice commands as the Y position data. Because the random forest conducts feature bagging when training its ensemble decision trees, the random forest model is able to implicitly isolate the relevant features. Though a neural network is expected to conduct the same approach by activating only the relevant neurons after the corresponding weights are minimized, the amount of data required in this work appears to be insufficient, which results in the neural network's performance deteriorating. The naïve Bayes algorithm also identifies the relevant features by explicitly defining $P(x_i|y)$, as shown by its 93.27% accuracy. Note that a possible compensation method is to manually isolate the relevant features for training, though that approach is inefficient when using large amounts of data. In addition, some features might change their correlation with the predictions or noise characteristics over time, such as when components degrade.

TABLE 1: ACCURACY OF DATA-DRIVEN CLASSIFIERS AS A RESULT OF CROSS-VALIDATION

Neural Network	Naïve Bayes	Random Forest
72.12%	93.27%	97.31%

Figure 8 shows the confusion matrices for each of the models. Note that the operator did not use the command "Move Left" since the payload did not overshoot in the Y direction. Figure 8 shows that most of the robot motion in this work is "Move Right" since this is the largest distance the robot travels. Figure 8 also shows that the neural network appears to skew towards the "Move Right" and "Stop" in its prediction errors. In addition, most errors in the naïve Bayes model appears to be in predicting "Stop" and "Move Right" when the robot should be lifting the payload and lowering the payload, respectively. Finally, the main prediction error for the random forest is predicting "Stop" when the robot should be lifting the payload.

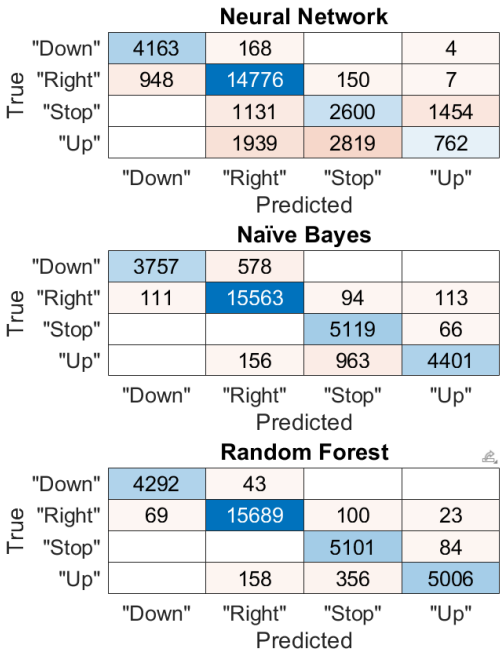

FIGURE 8: CONFUSION MATRICES FOR THE DATA-DRIVEN CLASSIFIERS

4.4 Trained Model Validation

After conducting cross-validation, the classifiers were trained with all available data for physical load-sharing experiments. Table 2 shows the training and prediction time (averaged over 1000 samples) on a Windows i7 PC. The neural network and the naïve Bayes models were shown to have the longest and shortest required training times, respectively. This is because the neural network back-propagation is known to be intensive while the optimization of the hyperparameters of the naïve Bayes model requires less iterations. Interestingly, the neural network has the fastest prediction time while the random forest has the slowest. This is because the actual implementation of the trained weights by the neural network is simply matrix multiplication with the use of activation functions while the random forest requires the calculations produced by multiple decision trees. Reducing the number of trees can decrease the prediction time at the cost of reducing accuracy. Finally, the naïve Bayes prediction time is significantly faster than the random forest since it uses analytical calculations in its

prediction. However, note that the prediction times are negligible compared to the 100 ms cycle time for robot commands.

TABLE 2: TRAINING AND PREDICTION TIMES FOR THE DATA-DRIVEN CLASSIFIERS

	Neural Network	Naïve Bayes	Random Forest
Training	25,210 ms	32 ms	5,930 ms
Prediction	0.075 ms	0.17 ms	6.4 ms

To evaluate the data-driven classifiers in physical load-sharing experiments, three trials were conducted for each model to predict and command coordinated robot motion with the operator without the use of audio feedback. Note that the neural network was not used in the physical load-sharing evaluation due to its poor prediction results as shown in Table 1 and Fig. 8 to avoid potential accidents. Hence, this work shows that the validation approaches shown in Tables 1 and 2, and in Fig. 8 can be used to conduct safety evaluations for these models before conducting real human-robot experiments.

Figure 9 shows time-series positions of the payload for the naïve Bayes classifier. Interestingly, even though the naïve Bayes exhibited a 93.27% prediction accuracy, the payload was unable to reach its final location. This is because the Y position in Fig. 9 shows that the robot and operator move past the desired workpiece location. Hence, the robot does not lower the payload onto its final Z destination. The naïve Bayes confusion matrix in Fig. 8 shows that the classifier tends to incorrectly predict "Move Right" when the payload should stop moving to the right and start lowering the payload. One method to compensate for this error is to conduct training experiments where the operator purposefully moves past the destination and says "Move Left" to readjust the payload back to the desired Y direction.

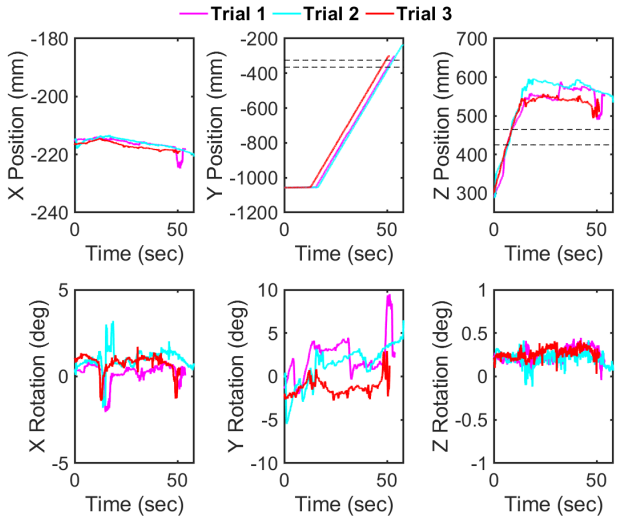

FIGURE 9: REPRESENTATIVE PAYLOAD POSITION RESULTS FOR NAÏVE BAYES CLASSIFIER WITH DASHED LINE REPRESENTING TARGET LOCATION OF PAYLOAD ± 20 MM

Figure 10 shows results for the random forest classifier. Figure 10 shows that the random forest classifier is able to accurately predict the intended operator commands and coordinate moving the payload to its final destination. This is because Table 1 shows that the random forest exhibits 97.31% accuracy, and its largest prediction error is predicting "Stop" when the robot should be moving up. This is most likely due to errors in transitioning between the initial zero velocity state and the lifting state. In addition, Figs. 9 and 10 show that the models experience variation in the payload trajectory. For instance, Fig. 10 shows that the random forest lifts the payload higher in the Z direction for Trial 1 vs. Trials 2 and 3. This could be because the data-driven models are also mimicking operator variation in this work.

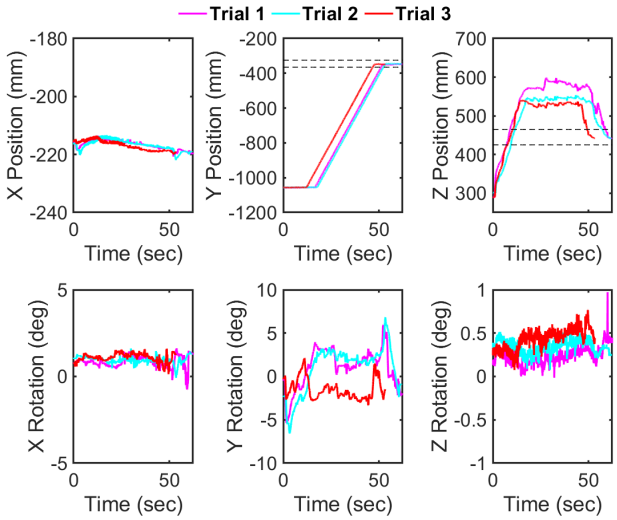

FIGURE 10: REPRESENTATIVE RESULTS FOR RANDOM FOREST CLASSIFIER WITH DASHED LINE REPRESENTING TARGET LOCATION OF PAYLOAD ± 20 MM

5. CONCLUSION

This paper presents an evaluation of three data-driven decision models in a human-robot load-sharing application. Robot, operator, and payload data were collected to train neural network, naïve Bayes, and random forest models by using a load-sharing testbed consisting of a motion tracker, wearable sensors, and voice commands. The random forest model was shown to exhibit the highest prediction accuracy and was able to command robot motion to accurately coordinate with the operator without the use of voice commands. In addition, the naïve Bayes and the random forest models were not used previously in human-robot load-sharing applications demonstrated in prior literature. Hence, this work can further the advancement of human-robot collaboration in manufacturing applications due to its novel insights into a human-robot coordinated task.

Though this work evaluates variation within an operator, human subjects research is desired as future work. In addition, developing methods to identify the most impactful features on predictions quickly and quantitatively are a topic of interest. Also, an ongoing topic of future work is to develop more data-

driven classifiers, such as an ensemble neural network, to improve robustness and accuracy. Finally, timing analysis between voice commands and human movements are also of interest regarding future work.

ACKNOWLEDGEMENTS

This work was funded by NIST and the National Research Council Research Associateship Program.

DISCLAIMER

Certain commercial equipment, instruments, or materials are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by NIST, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

REFERENCES

[1] Anvaripour, M., Khoshnam, M., Menon, C., and Saif, M., 2019, "Safe Human Robot Cooperation in Task Performed on the Shared Load," *2019 International Conference On Robotics And Automation (ICRA)*, IEEE, pp. 3761–3767.

[2] Kim, W., Lee, J., Peternel, L., Tsagarakis, N., and Ajoudani, A., 2017, "Anticipatory Robot Assistance for the Prevention of Human Static Joint Overloading in Human–Robot Collaboration," *IEEE Robotics and Automation Letters*, 3(1), pp. 68–75.

[3] Berger, E., Vogt, D., Haji-Ghassemi, N., Jung, B., and Amor, H. ben, 2013, "Inferring Guidance Information in Cooperative Human-Robot Tasks," *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, IEEE, pp. 124–129.

[4] DelPreto, J., and Rus, D., 2019, "Sharing the Load: Human-Robot Team Lifting Using Muscle Activity," *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 7906–7912.

[5] van der Spaa, L., Gienger, M., Bates, T., and Kober, J., 2020, "Predicting and Optimizing Ergonomics in Physical Human-Robot Cooperation Tasks," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 1799–1805.

[6] Sheng, W., Thobbi, A., and Gu, Y., 2014, "An Integrated Framework for Human–Robot Collaborative Manipulation," *IEEE Transactions on Cybernetics*, 45(10), pp. 2030–2041.

[7] Tan, J. T. C., and Arai, T., 2011, "Triple Stereo Vision System for Safety Monitoring of Human-Robot Collaboration in Cellular Manufacturing," *2011 IEEE International Symposium on Assembly and Manufacturing (ISAM)*, IEEE, pp. 1–6.

[8] Lawitzky, M., Mörtl, A., and Hirche, S., 2010, "Load Sharing in Human-Robot Cooperative Manipulation," *19th International Symposium in Robot and Human Interactive Communication*, IEEE, pp. 185–191.

[9] Tsarouchi, P., Michalos, G., Makris, S., Athanasatos, T., Dimoulas, K., and Chryssolouris, G., 2017, "On a Human–Robot Workplace Design and Task Allocation System," *International Journal of Computer Integrated Manufacturing*, 30(12), pp. 1272–1279.

[10] Roveda, L., Maskani, J., Franceschi, P., Abdi, A., Braghin, F., Tosatti, L. M., and Pedrocchi, N., 2020, "Model-Based Reinforcement Learning Variable Impedance Control for Human-Robot Collaboration," *Journal of Intelligent & Robotic Systems*, 100(2), pp. 417–433.

[11] Nenna, V., Pidlisecky, A., and Knight, R., 2011, "Application of an Extended Kalman Filter Approach to Inversion of Time-lapse Electrical Resistivity Imaging Data for Monitoring Recharge," *Water Resources Research*, 47(10), pp. W10525.

[12] Dornfeld, D. A., and DeVries, M. F., 1990, "Neural Network Sensor Fusion for Tool Condition Monitoring," *CIRP Annals*, 39(1), pp. 101–105.

[13] Zhang, B., Jaiswal, P., Rai, R., Guerrier, P., and Baggs, G., 2019, "Convolutional Neural Network-Based Inspection of Metal Additive Manufacturing Parts," *Rapid Prototyping Journal*, 25(3), pp. 530-540.

[14] Anthony, M., and Bartlett, P. L., 2009, "Neural Network Learning: Theoretical Foundations," Cambridge University Press.

[15] Liu, D. C., and Nocedal, J., 1989, "On the Limited Memory BFGS Method for Large Scale Optimization," *Mathematical Programming*, 45(1), pp. 503–528.

[16] Rish, I., 2001, "An Empirical Study of the Naive Bayes Classifier," *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, pp. 41–46.

[17] Fan, S.-K. S., Hsu, C.-Y., Tsai, D.-M., He, F., and Cheng, C.-C., 2020, "Data-Driven Approach for Fault Detection and Diagnostic in Semiconductor Manufacturing," *IEEE Transactions on Automation Science and Engineering*, 17(4), pp. 1925–1936.

[18] Wu, D., Jiang, Z., Xie, X., Wei, X., Yu, W., and Li, R., 2019, "LSTM Learning with Bayesian and Gaussian Processing for Anomaly Detection in Industrial IoT," *IEEE Transactions on Industrial Informatics*, 16(8), pp. 5244–5253.

[19] Rodriguez-Galiano, V. F., Ghimire, B., Rogan, J., Chica-Olmo, M., and Rigol-Sanchez, J. P., 2012, "An Assessment of the Effectiveness of a Random Forest Classifier for Land-Cover Classification," *ISPRS Journal of Photogrammetry and Remote Sensing*, 67, pp. 93–104.

[20] Wescoat, E., Krugh, M., and Mears, L., 2021, "Random Forest Regression for Predicting an Anomalous Condition on a UR10 Cobot End-Effector from Purposeful Failure Data," *Procedia Manufacturing*, 53, pp. 644–655.

[21] Jun, S., Lee, S., and Chun, H., 2019, "Learning Dispatching Rules Using Random Forest in Flexible Job Shop Scheduling Problems," *International Journal of Production Research*, 57(10), pp. 3290–3310.

[22] Ho, T. K., 1998, "The Random Subspace Method for Constructing Decision Forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8), pp. 832–844.