2024 21st International Conference on Ubiquitous Robots (UR)
June 24 - 27, 2024, New York University, USA

Investigating the Generalizability of Assistive Robots Models over Various Tasks

Hamid Osooli, Christopher Coco, Johnathan Spanos, Amin Majdi and Reza Azadeh

***Abstract*— In the domain of assistive robotics, the significance of effective modeling is well acknowledged. Prior research has primarily focused on enhancing model accuracy or involved the collection of extensive, often impractical amounts of data. While improving individual model accuracy is beneficial, it necessitates constant remodeling for each new task and user interaction. In this paper, we investigate the generalizability of different modeling methods. We focus on constructing the dynamic model of an assistive exoskeleton using six data-driven regression algorithms. Six tasks are considered in our experiments, including horizontal, vertical, diagonal from left leg to the right eye and the opposite, as well as eating and pushing. We constructed thirty-six unique models applying different regression methods to data gathered from each task. Each trained model's performance was evaluated in a cross-validation scenario, utilizing five folds for each dataset. These trained models are then tested on the other tasks that the model is not trained with. Finally the models in our study are assessed in terms of generalizability. Results show the superior generalizability of the task model performed along the horizontal plane, and decision tree based algorithms.**

I. Introduction

The importance of modeling in the realm of assistive robotics is a well-recognized aspect that is crucial for effective control and user interaction. Assistive robots, particularly those in constant interaction with human users, present unique challenges in modeling. Unlike conventional rigid body systems governed by standard physics, these robots involve complex interactions between the user and the robot. This complexity necessitates the adoption of data-driven modeling techniques, which are frequently used in robotics research [1], [2], [3], [4].

One of the primary challenges in these systems is the variability of the user actions. For a model to be task generalizable, it ideally needs exposure to a comprehensive range of movement trajectories. Previous studies, such as [3], demonstrated the need for an extensive set of motion trajectories, by training robots on a vast array of random movement trajectories, enhancing their ability to perform unforeseen tasks. However, this approach is impractical in real-world scenarios, especially when considering the vast data requirements and the unique challenges posed by assistive robots used by individuals with disabilities.

To enhance the accuracy of the trained model, Zhang et al. [4] proposed an optimization problem, where model accuracy is a constraint. While novel methods to increase accuracy are valuable, they do not inherently lead to a generalizable model that minimizes the need for additional data collection. This highlights the need for an in-depth study into generalizability that provides sufficient data for developing a generalizable model.

This paper introduces a comprehensive study on the generalizability of the tasks models. We focus on upper-limb exoskeletons, specifically using the MyoPro 2 Motion-G, a 2 DOF exoskeleton, equipped with two APDM opal IMU sensors. Our study encompasses six algorithms namely Locally Weighted Projection Regression (LWPR), K-Nearest Neighbours (KNN), Support Vector Regression (SVR), eXtreme Gradient Boosting (XGBoost), Multi Layer Perceptron (MLP), and Gaussian Process Regression (GPR). The performance of each algorithm is evaluated on six different tasks (Horizontal (H), Vertical (V), diagonal from Left leg to Right eye (LR), diagonal from Right leg to Left eye (RL), Eating (E), and Pushing (P)). We use the R-squared score to assess the effectiveness and generalizability of each task model and algorithm.

Results show that the task models performed along the horizontal plane, and decision tree based algorithms are superior in terms of the generalizability. These findings are practical for developing strategies that can enhance the effectiveness and adaptability of models across diverse scenarios.

II. Related Work

The generalizability of the machine learning models helps on reducing the need for repetitive training and data collection. This concept has been addressed in other fields that study the generalizability of deep learning models in visual tasks [5]. However, generalizability remains a largely

Authors are with the Persistent Autonomy and Robot Learning (PeARL) Lab, University of Massachusetts Lowell, Lowell, MA 01854, USA {hamid_osooli, reza_azadeh}@uml.edu, {christopher_coco, johnathan_spanos, amin_majdi}@student.uml.edu

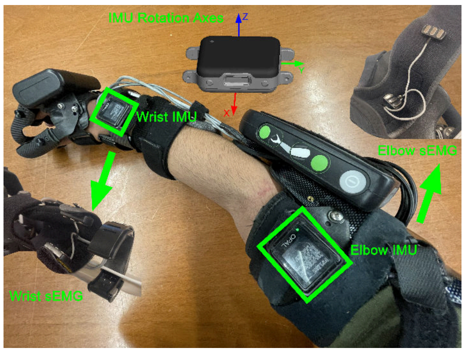

Fig. 1. Myopro prosthetic augmented with IMU sensors for data collection. Figure annotation highlights the placement of sEMG sensors and the rotational axes of APDM Opal IMU sensors.

2024 21st International Conference on Ubiquitous Robots (UR) | 979-8-3503-6107-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/UR61395.2024.10597514

Authorized licensed use limited to: UNIV OF MASS-LOWELL. Downloaded on February 09,2025 at 21:23:03 UTC from IEEE Xplore. Restrictions apply.

unexplored factor in modeling the interaction between the users and the prosthetic robots.

Siu et al., [6] introduced a non-adaptive controller that integrates pressure and sEMG data gathered during training to construct a K-nearest neighbors (KNN) classifier. This classifier was built on fourteen signal features derived from each of the six sEMG sensors. The sEMG features were normalized through mean subtraction and division by standard deviation. Additionally, an adaptive controller was employed to update the sEMG mapping using the KNN classifier. The effectiveness of the proposed model was assessed through a tabletop book-shelving task. The controller exhibited adaptability to user-specific physiological changes, such as fatigue. The authors then proposed a Learning from Demonstration (LfD) [7], [8] approach in which the user demonstrates the task for the robot, enabling it to learn and be trained. Although practical in many robotic experiments, LfD approach may be less effective, especially in cases where the prosthesis is being used for an impaired limb.

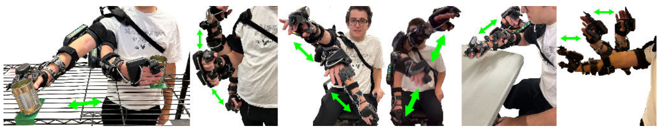
Fig. 2. Overview of the diverse tasks employed for data acquisition from test subjects, including Horizontal (H), Vertical (V), diagonal from Left leg to Right eye (LR), diagonal from Right leg to Left eye (RL), Eating (E), and Pushing (P).

To achieve task generalizable capability, the model necessitates exposure to a comprehensive set of diverse trajectories. Kwiatkowski and Lipson [3] addressed this requirement by training the robot on a dataset comprising 1000 randomly generated trajectories, enhancing its adaptability to unforeseen tasks. Zhang et al. [4] approached the modeling task as an optimization problem, treating model accuracy as a constraint. Since pre-covering the complete state-space with data is impractical [9], many of the works in this area proposed online model learning [10], [11], [12], as a way to gather more data for modeling.

While collection of huge amounts of motion trajectory data is feasible in various scenarios, it poses a challenge for assistive robots, particularly exoskeletons. These devices often help users with limited mobility, such as individuals with disabilities, who may not engage in extensive movements suitable for data collection. Therefore, the strategic design of efficient tasks that results in sufficient amount of data for training a generalized model becomes important in such contexts.

III. Problem Formulation

A comprehensive model capturing the interaction between the user and the exoskeleton necessitates an encompassing description of the device states across its elbow and wrist degrees of freedom. The system's formulation, encapsulating robot states, inputs, and user inputs can be formulated as:

$$x_{t+1} = x_t + f(x_t, u_t, v_t) + \eta_t, \quad \eta_t \sim \mathcal{N}(0, \Sigma_\eta), \tag{1}$$

where f defines the unknown dynamic evolution of the interaction between the user and the exoskeleton over time t, $x \in \mathbb{R}^L$ is the state vector (elbow/wrist angles and angular velocities), and $v \in \mathbb{R}^N$ is the user's action incorporating biceps/triceps sEMG measurements for the elbow and the wrist, $u \in \mathbb{R}^M$ is the robots action vector including the thresholds considered on the difference between the sEMG signals of the biceps and triceps or opening and closing of the hand. This sEMG threshold triggers the robot's assistance when surpassed. The uncertainties from IMUs are modeled by an additive white Gaussian noise [13]. Potential noises from the exoskeleton, however, were compensated by the built-in mechanisms, allowing for their exclusion from our noise consideration framework.

IV. Dynamic Model Learning via Regression

Modeling the dynamics of exoskeleton robots is more challenging due to the presence of human actions and the interaction between the device and the human. To accurately model f, in this paper, we utilize six different regression methods, namely Locally Weighted Projection Regression (LWPR), K-Nearest Neighbours (KNN), Support Vector Regression (SVR), eXtreme Gradient Boosting (XGBoost), Multi Layer Perceptron (MLP), and Gaussian Process Regression (GPR).

Each model incorporates an input vector made of the states x_t, robot's actions u_t and the user's actions v_t, represented as the vector $\tilde{x}_t = [x_t, u_t, v_t] \in \mathbb{R}^{L+M+N}$. For the training targets, we use the difference between the current and future state vectors: $\Delta x_t = x_{t+1} - x_t \in \mathbb{R}^L$. The target dataset is defined as $T^L_{1:t} = \{f(\tilde{x}_1), \dots, f(\tilde{x}_t)\}$, and a new input point at which the model is queried is shown with $\tilde{x}_*$.

A. Locally Weighted Projection Regression (LWPR)

LWPR [14] is an algorithm that achieves nonlinear function approximation in high dimensional spaces with redundant and irrelevant input dimensions having little to no impact on the output. It considers R locally linear models for approximation of the function as: $f = \mathbb{E}\{\bar{f}_k|\tilde{x}_*\} = \sum_{k=1}^{R} \bar{f}_k p(k|\tilde{x}_*)$. From the Bayes' theorem, when we query a new input point $\tilde{x}_*$, the probability of the model k can be expressed as:

$$p(k|\tilde{x}_*) = \frac{p(k|\tilde{x}_*)}{p(\tilde{x}_*)} = \frac{p(k|\tilde{x}_*)}{\sum_{k=1}^{R} p(k|\tilde{x}_*)} = \frac{w_k}{\sum_{k=1}^{R} w_k}. \tag{2}$$

Hence,

$$f(\tilde{x}_*) \sim \frac{\sum_{k=1}^{R} w_k \bar{f}_k(\tilde{x}_*)}{\sum_{k=1}^{R} w_k}, \tag{3}$$

we have $\bar{f}_k = \bar{x}_k^\top \hat{\theta}_k$ and $\bar{x}_k = [(\tilde{x}_* - c_k)^\top, 1]^\top$, in which $\hat{\theta}_K$ consists of the estimated parameters of the model, and c_k is the center of the k-th linear model. w_k is the weight that determines whether a data point $\tilde{x}_*$ is into the region of validity of the model k. This is similar to a receptive field, and is defined by a Gaussian kernel

Authorized licensed use limited to: UNIV OF MASS-LOWELL. Downloaded on February 09,2025 at 21:23:03 UTC from IEEE Xplore. Restrictions apply.

$$w_k = \exp(-\frac{1}{2}(\tilde{x}_* - c_k)^\top D_k(\tilde{x}_* - c_k)), \quad (4)$$

where D_k is the distance matrix and should be positive definite. In the learning procedure, the shape of the D_k and the $\hat{\theta}_k$ parameters of the local models are adjusted to minimize the error between the predicted values and the observed targets [11]. We initialize D_k as $D_k = rI_{L+M+N}$, where r is a constant value tuned based on the model performance and I is the identity matrix with the same size as the inputs.

B. K-Nearest Neighbours (KNN)

The KNN algorithm [15] is based on the distance-weighted nearest neighbor estimation, where k most similar values of the input data $\tilde{x}$ are used for the prediction of the diameter distribution of $f(\tilde{x})$. The similarity of the data is measured by their distance as

$$d_{ij} = \sum_{l=1}^{L} c_l \|(\tilde{x}_{il} - \tilde{x}_{jl})\|, \quad (5)$$

where $\tilde{x}_l$ is the input, and c_l is the coefficient for the input. Then the distances d are sorted based on the weight calculated by

$$w_{ij} = \frac{(\frac{1}{1+d_{ij}})^p}{\sum_{i=1}^{k}(\frac{1}{1+d_{ij}})^p}, \forall\, i \neq j, \quad (6)$$

where k is the number of the nearest neighbors used, and p is the weighting parameter of distance. The weighting parameter p determines how fast should the weights of the nearest neighbors decrease when the distance d_{ij} increases, and weights should sum to one.

C. Support Vector Regression (SVR)

SVR [16] is an algorithm that belongs to the family of support vector machines. In SVR, the regression function $f(\tilde{x})$ is estimated by the hyper plane h:

$$f(\tilde{x}) = h^\top \tilde{x}_* + b \text{ with } h \in \mathbb{R}^{L+M+N}, b \in \mathbb{R}. \quad (7)$$

Exploiting the structural risk minimization [17], the generalization accuracy of the SVR is optimized on the empirical error and flatness of the $f(\tilde{x})$ which is the result of the small values for h. Therefore the SVR aims to include the dataset patterns inside an ϵ-tube while minimizing the $\|h\|^2$. We can formulate this as an optimization problem:

$$\begin{aligned}
&\text{minimize}\frac{1}{2}\|h\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*), \\
&\text{s.t. } f(\tilde{x}_i) - h^\top \tilde{x}_i - b \leq \epsilon + \xi_i, \\
&h^\top \tilde{x}_i + b - f(\tilde{x}_i) \leq \epsilon + \xi_i^*, \\
&\xi_i, \xi_i^* \geq 0, \quad i = 1, \ldots, l, \qquad (8)
\end{aligned}$$

where C, ϵ, and ξ, ξ^* are the cost for the trade-off between the empirical error and the flatness of the $f(\tilde{x})$, the ϵ-tube size, and slack variables. Adding the Lagrangian multipliers α and α^* transforms the quadratic programming problem into a dual optimization. Furthermore, SVR is capable of non-linear function approximation by employing a kernel function $k_f(\tilde{x}_i, \tilde{x}_j)$. Thus the SVR estimates f as:

$$f(\tilde{x}) \sim \sum_{i=1}^{s}(\alpha - \alpha^*)k_f(\tilde{x}_i, \tilde{x}_j) + b, \quad (9)$$

where s is the number of the support vectors, and b is a constant [18]. In this paper we use a composite kernel as $k_f(\tilde{x}_i, \tilde{x}_j) = k_{\text{constant}} + k_{\text{matern}} + k_{\text{white}}$ where $k_{\text{constant}} = 1^2$, $k_{\text{white}} = \Lambda^2$, and $k_{\text{matern}}(\tilde{x}_i, \tilde{x}_j)$. Λ is the noise level for the white kernel. We consider $l = 2, \nu = 1.5$, and $\Lambda = 1$.

D. eXtreme Gradient Boosting (XGBoost)

XGBoost [19] is an algorithm that uses a decision tree as its base classifier for the target dataset $D^L_{1:t}$, that contains L observations. In the typical Gradient Boosting (GB) algorithms, we use B additive functions for G times boosting of the gradient, to predict the output. Consider $f_k(\tilde{x})$ as the prediction for the k-th instance at the b-th boost

$$f_k(\tilde{x}) \sim \sum_{b=1}^{B} f_b(\tilde{x}_k). \quad (10)$$

GB minimizes a loss function:

$$O_b = \sum_{k=1}^{L} e(f_k(\tilde{x}), \hat{f}_k(\tilde{x})), \quad (11)$$

where $e(f_k(\tilde{x}), \hat{f}_k(\tilde{x}))$ is the measurement of the difference between the prediction $f_k(\tilde{x})$ and its real value $\hat{f}_k(\tilde{x})$.

If we add a regularization term $\Omega(f_b)$ to (11), the result will be the loss function of XGBoost:

$$\begin{aligned}
O_b &= \sum_{k=1}^{L} e(f_k(\tilde{x}), \hat{f}_k(\tilde{x})) + \sum_{b=1}^{B}\Omega(f_b) \\
&= \sum_{k=1}^{L} e(f_k(\tilde{x}), \hat{f}_k(\tilde{x})) + \gamma\tau + 0.5\lambda\|\omega\|^2. \qquad (12)
\end{aligned}$$

The regularization term $\Omega(f_b)$ penalizes the complexity of the model, and can be expressed as $\gamma\tau + 0.5\lambda\|\omega\|^2$. Where τ represents the number of leaves in the tree filled with data, and γ is the minimum loss reduction threshold for further partition. If the loss reduction is less than γ, XGBoost will stop. λ is a fixed coefficient, and $\|\omega\|^2$ is the Euclidean norm of the leaf weight [20].

E. Multi Layer Perceptron (MLP)

MLP [21] is a class of artificial neural network that consists of multiple layers of neurons, each layer fully connected to the next one. Usually the structure includes an input layer, one or more hidden layers, and an output layer. In MLP, each hidden layer g (where $1 \leq g \leq G$) transforms the output of the previous layer $\tilde{x}^{g-1}$ using a weight matrix ϖ^g and a bias vector β^g. The transformation is a linear combination followed by a nonlinear activation σ:

Authorized licensed use limited to: UNIV OF MASS-LOWELL. Downloaded on February 09,2025 at 21:23:03 UTC from IEEE Xplore. Restrictions apply.

$$\hat{f}(\tilde{x})^g = \sigma(\varpi^g \hat{f}(\tilde{x})^{g-1} + \beta^g). \tag{13}$$

In this paper, we used ReLU as the activation function σ. The final layer G produces the output. We also use L-BFGS [22] to update the parameters using the gradient of the R-squared as the loss function.

F. Gaussian Process Regression (GPR)

GPR [23] is an algorithm that for each dimension $z = 1, \dots, L$ of the difference vector Δx_t, estimates f as

$$f(\tilde{x}) \sim \mathcal{GP}(\mu_f(\tilde{x}), k_f(\tilde{x}, \tilde{x}')). \tag{14}$$

For the target dataset $D^L_{1:t}$, the trained GPR model can be queried at a new input point $\tilde{x}_*$:

$$p(f(\tilde{x}_*)|D^L_{1:t}, \tilde{x}_*) = \mathcal{N}(\mu_f(\tilde{x}_*), \sigma^2_f(\tilde{x}_*)). \tag{15}$$

Unlike other regression methods, GPR does not provide a prediction set. Instead, it provides two lists of means and variances for each prediction. The mean and variance predictions are calculated by a kernel vector $\mathbf{k}_f = k(D^L_{1:t}, \tilde{x}_*)$, and a kernel matrix K_f, with entries of $K^{ij}_f = k_f(\tilde{x}_i, \tilde{x}_j)$ as

$$\begin{aligned} \mu_f(\tilde{x}_*) &= \mathbf{k}_f^\top K_f^{-1} D^L_{1:t} \\ \sigma^2_f(\tilde{x}_*) &= k_f(\tilde{x}_*, \tilde{x}_*) - \mathbf{k}_f^\top K_f^{-1} \mathbf{k}_f. \end{aligned} \tag{16}$$

where k_f is the composite kernel used in SVR. To leverage the same evaluation used for other models, we incorporate the mean values of the GPR output.

V. Experimental Setup

In our experiments, we use the MyoPro, a lightweight two degrees of freedom upper limb exoskeleton [24]. This wearable robot utilizes four surface electromyography (sEMG) sensors. Sensor placement includes two on the upper arm and two on the forearm. The device allows for user-specific threshold adjustments at these sensor locations, which differentiate between the muscular activities of the biceps and triceps for arm movements, and those related to the hand's opening and closing gestures. Activation of the device's motor occurs upon exceeding these predefined thresholds, thereby facilitating user assistance. While the system provides data concerning the velocity of the integrated motors, it lacks the capability to offer information pertaining to the rotational movements of the hand.

To collect data for modeling the robot, we add two APDM opal IMU sensors to the arm and wrist locations to measure the rotations of the hand (see Fig. 1). Having access to the APDM opal IMU gyroscopes, we use Unscented Kalman filter [25] for calculation of the angles from angular velocity readings.

We conducted data collection for six distinct tasks: Horizontal (H), Vertical (V), diagonal from Left leg to the Right eye (LR), diagonal from Right leg to the Left eye (RL), Eating (E), and Pushing (P) shown in Fig. 2. These tasks are chosen due to their diverse features. Horizontal (H) involves a movement along the horizontal plane while Vertical (V) is a movement along the vertical plane. On the other hand, diagonal from Left leg to Right eye (LR) and diagonal from Right leg to Left eye (RL) are movements that cross the body. The last two tasks replicate the daily activities of the user where Eating (E) involves a range of motions towards the mouth, and Pushing (P) is a movement in the outward direction.

For the horizontal task, participants moved an empty can between two predefined points on a table. The vertical task involved rotating the arm around the axis originating from the shoulder. We also introduced two diagonal tasks to complement horizontal and vertical tasks, requiring participants to move their hand from their leg to the front of their eye—either from the left leg towards the right eye or the opposite. The last two tasks are eating and pushing. In the eating task, participants moved their wrist from the table towards their mouth, while in the pushing task, they performed a forward arm movement, closing and opening the arm starting from the chest.

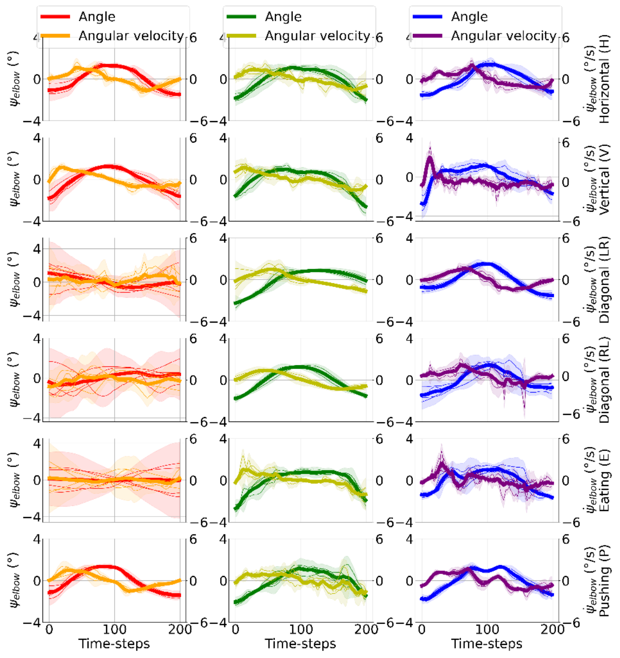

Fig. 3. Mean and standard deviation of the first two input features (elbow angle and angular velocity) calculated with 99% confidence interval from four trials for each of the six tasks, with data collected from three test subjects distinguished by different colors from left to the right.

VI. Data Processing and Validation

As illustrated in Fig. 3, four separate trials for each task is separated from the data. We then averaged over the number of trials (four) to construct our dataset. To have a similar interval length across different tasks dataset, we employed one-dimensional linear interpolation. We linearly rescaled the inputs to have zero mean and unit variance on the training set. Although it is possible to similarly rescale the output data, we chose not to do this because our tests showed that it did not significantly enhance our results.

Authorized licensed use limited to: UNIV OF MASS-LOWELL. Downloaded on February 09,2025 at 21:23:03 UTC from IEEE Xplore. Restrictions apply.

All of the six experiments were repeated by three users (among the authors), and the data was used for modeling by six different regression methods discussed in Sec IV.

We divided the dataset into five distinct subsets for cross-validation. The performance of the models were evaluated by averaging the outcomes of the R-squared score, expressed in percentage terms, across the cross-validation subsets. This metric was chosen over other evaluation criteria due to its universal applicability across all models and methods used in our study.

VII. Results & Discussion

Our main goal is to determine the capability of a model, trained on a specific task (Fig. 2) to generalize to other tasks. We evaluate the generalizability of the obtained models when trained on a specific task and tested on other tasks. The complication is due to the fact that the trained model must be capable of approximating twelve different features as mentioned in Sec. III. For instance as it is shown in Fig. 3 while the elbow angle follows a roughly similar pattern in different tasks, the angular velocity varies significantly from one task to the other.

We use a graph representation to demonstrate the results. The nodes indicate tasks, while the edges show level of generalizability of the model when tested on the task in the destination node. As shown in Fig. 4, each model was trained on the home task, *node* and tested on the destination task *node*. The brightness of the *edges* indicates the R-squared score for that specific training averaged over the models trained for three subjects. Thus a brighter or darker edge means lower or higher generalizability, respectively. It also allows us to assign a score based on the average number of input/output edge weights to the graphs, and order the models from the highest to the lowest. Based on their corresponding scores in Fig. 4, we notice that in terms of generalizability the selected regression algorithms can be sorted from best to worst as: XGBoost (84.93%), GPR (82.31%), KNN (76.79%), LWPR (69.31%), SVR (63.31%), and MLP (55.65%).

The difference in performances could have various reasons. For instance, the performance of LWPR is highly dependent on the choice of the hyper-parameter r, which defines the initial distance between local models. To have a fair comparison, we tuned r once the model was being trained, and kept it fixed when testing on other task datasets.

In addition to our finding about the generalizability of the trained models, by considering an R-squared value of 80% as the threshold for acceptable performance, and counting the output edges of each node, we can assess the extent to which each task model is generalizable within each algorithm. Fig. 5 shows the results for this evaluation. We notice that on a model trained by LWPR algorithm the H task model has the highest generalizability in comparison to the set of (RL, E, and P) task models that are 16.39%, 17.98%, and 16.98% less generalizable respectively. In this algorithm the V and the LR task models do not generalize to any other model with an over 80% R-squared. When using KNN, the E task model is the most generalizable and the set of (H, LR, RL, and P) are 17.18%, 16.79%, 17.02%, and 16.32% less generalizable and the V task is not generalizable over the threshold. In SVR only the model trained on the task P is 16.11% generalizable and other task models generalize below the 80% threshold.

In the XGBoost that was ranked first in terms of generalizability, the H task on average generalizes by 86.90% to the other tasks and V, LR, RL, E, and P task models are 32.65%, 18.69%, 17.05%, 15.91%, 17.33% less generalizable than it. When ranking the models on their level of generalizability, we noticed that in MLP that is the least generalizable, all of the task models are nearly generalizable by 36 37%. Except

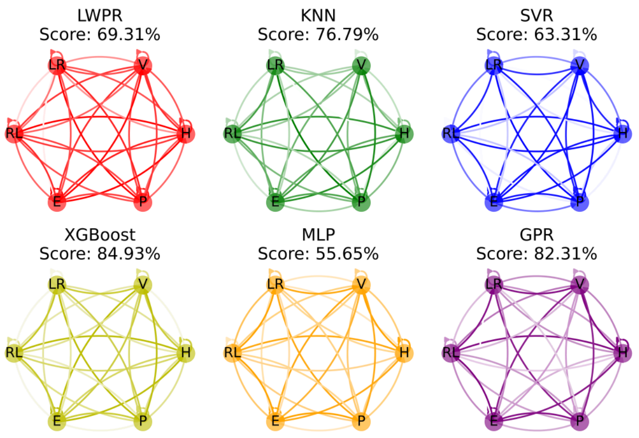

Fig. 4. Performance evaluation of each model, where training is conducted on the home node and testing is performed on the destination node data. Edge color intensity inversely correlates with the models' ability to generalize; a brighter edge shows lower generalizability.

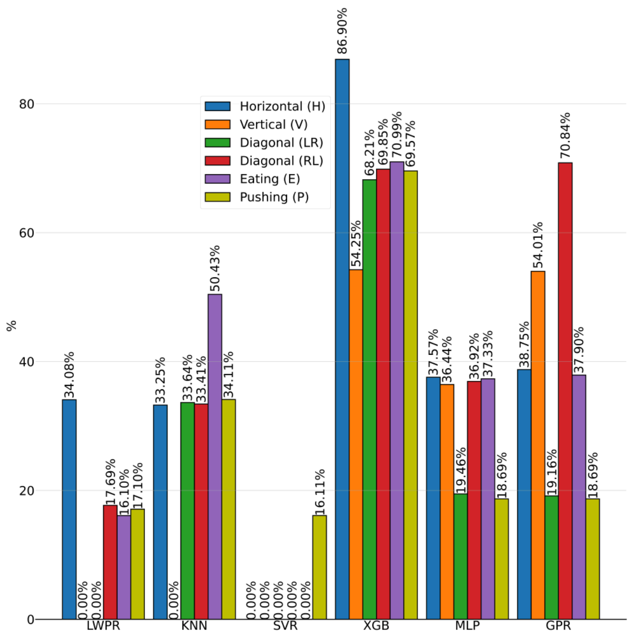

Fig. 5. The generalizability of different task data sets within each algorithm, using a R-squared value of 80% as the threshold for acceptable performance.

Authorized licensed use limited to: UNIV OF MASS-LOWELL. Downloaded on February 09,2025 at 21:23:03 UTC from IEEE Xplore. Restrictions apply.

for the LR and the P that are 18% less generalizable than the others. Our second place in the ranking list was for GPR that has a different generalizability from others. When using GPR, the RL is the most generalizable task model and V, H, E, LR, and P are respectively less generalizable models by 16.83%, 32.09%, 32.94%, 51.68%, 52.15%.

We then evaluated the algorithm-agnostic generalizability of the task models by summing their levels of generalizability in each algorithm and averaging over the total number of tasks. Fig. 6 shows that we can order the task models in our study for descending generalizability as H, RL, E, P, V, and LR.

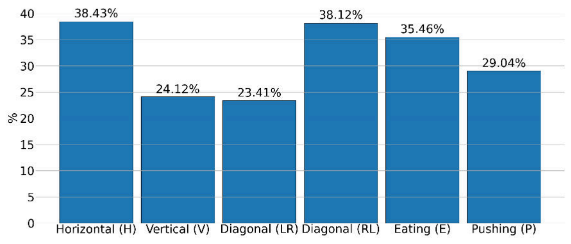

Fig. 6. The average generalizability for the task datasets in our study, using an R-squared value of 80 as the threshold for acceptable performance.

The average training times for LWPR, KNN, SVR, XGBoost, MLP, and GPR in our study averaged over 5 folds of cross validation sets, six tasks and three subjects, were 0.003, 0.034, 0.005, 0.147, 0.247, and 0.243 seconds, respectively. This information can offer a more holistic view of the model performance when integrated with the generalizability.

VIII. Conclusion

In this paper, we focused on task generalizability in the data-driven modeling of assistive robots, particularly focusing on upper-limb exoskeletons. Our study employed six regression modeling techniques, among which the dynamic model constructed using the XGBoost algorithm showed superior generalizability capabilities. We collected the data from six different tasks, with an emphasis on cross-validating models trained on each one. The main finding from our experiments was the superiority of the model built for the H task in terms of generalizability. This insight provides context for refining the selection of model learning techniques and choosing appropriate tasks with specific features for model training. This study provides potential opportunities to construct more generalizable models that could lead to improved performance in a diverse array of tasks, ultimately benefiting the users of such assistive technologies.

Acknowledgment

This work was supported by the National Science Foundation (CMMI-2110214).

References

[1] T. Wu and J. Movellan, "Semi-parametric gaussian process for robot system identification," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 725–731.

[2] S. Riedel and F. Stulp, "Comparing semi-parametric model learning algorithms for dynamic model estimation in robotics," *stat*, vol. 1050, p. 27, 2019.

[3] R. Kwiatkowski and H. Lipson, "Task-agnostic self-modeling machines," *Science Robotics*, vol. 4, no. 26, p. eaau9354, 2019.

[4] C. Zhang, A. Khan, S. Paternain, and A. Ribeiro, "Sufficiently accurate model learning," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 10 991–10 997.

[5] K. Alhamoud, H. A. A. K. Hammoud, M. Alfarra, and B. Ghanem, "Generalizability of adversarial robustness under distribution shifts," *Transactions on Machine Learning Research*, 2023.

[6] H. C. Siu, A. M. Arenas, T. Sun, and L. A. Stirling, "Implementation of a surface electromyography-based upper extremity exoskeleton controller using learning from demonstration," *Sensors*, vol. 18, no. 2, p. 467, 2018.

[7] B. Hertel, M. Pelland, and S. R. Ahmadzadeh, "Robot learning from demonstration using elastic maps," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 7407–7413.

[8] H. Ravichandar, S. R. Ahmadzadeh, M. A. Rana, and S. Chernova, "Skill acquisition via automated multi-coordinate cost balancing," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7776–7782.

[9] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey," *Cognitive processing*, vol. 12, pp. 319–340, 2011.

[10] H. Cao, Y. Yin, D. Du, L. Lin, W. Gu, and Z. Yang, "Neural-network inverse dynamic online learning control on physical exoskeleton," in *Neural Information Processing: 13th International Conference (ICONIP)*. Springer, 2006, pp. 702–710.

[11] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Model learning with local gaussian process regression," *Advanced Robotics*, vol. 23, no. 15, pp. 2015–2034, 2009.

[12] R. F. Reinhart and J. J. Steil, "Reaching movement generation with a recurrent neural network based on learning inverse kinematics for the humanoid robot icub," in *9th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2009, pp. 323–330.

[13] K. Nirmal, A. Sreejith, J. Mathew, M. Sarpotdar, A. Suresh, A. Prakash, M. Safonova, and J. Murthy, "Noise modeling and analysis of an imu-based attitude sensor: improvement of performance by filtering and sensor fusion," in *Advances in optical and mechanical technologies for telescopes and instrumentation II*, vol. 9912. SPIE, 2016, pp. 2138–2147.

[14] S. Vijayakumar, A. D'Souza, and S. Schaal, "Lwpr: A scalable method for incremental online learning in high dimensions," *Journal of Machine Learning Research*, pp. 623–626, 2008.

[15] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[16] M. Awad, R. Khanna, M. Awad, and R. Khanna, "Support vector regression," *Efficient learning machines: Theories, concepts, and applications for engineers and system designers*, pp. 67–80, 2015.

[17] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 1999.

[18] D. Kim, H.-j. Lee, and S. Cho, "Response modeling with support vector regression," *Expert Systems with Applications*, vol. 34, no. 2, pp. 1102–1108, 2008.

[19] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[20] Y. Wang and S. Ni, "A xgboost risk model via feature selection and bayesian hyper-parameter optimization," *International Journal of Database Management Systems (IJDMS)*, vol. 11, no. 1, 2019.

[21] F. Murtagh, "Multilayer perceptrons for classification and regression," *Neurocomputing*, vol. 2, no. 5-6, pp. 183–197, 1991.

[22] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1-3, pp. 503–528, 1989.

[23] C. Williams and C. Rasmussen, "Gaussian processes for regression," *Advances in neural information processing systems*, vol. 8, 1995.

[24] M. Myomo Inc., Cambridge, "Myomo – The MyoPro Brace for Stroke Survivor's Paralyzed Arm." [Online]. Available: https://myomo.com

[25] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium*. IEEE, 2000, pp. 153–158.

Authorized licensed use limited to: UNIV OF MASS-LOWELL. Downloaded on February 09,2025 at 21:23:03 UTC from IEEE Xplore. Restrictions apply.