EMG- based Grasping force estimation for robot skill transfer learning

Waddah Ali
Faulty of control systems and robotics.
ITMO University
Saint Petersburg, Russia
waddahkh.ali94@gmail.com

Sergey Kolyubin
Faulty of control systems and robotics.
ITMO University
Saint Petersburg, Russia
s.kolyubin@itmo.ru

***Abstract*—In this report, we are discussing a new machine learning architecture Multi-layer preceptron - random forest regressors pipeline (MLP-RF model) that stacks two ML regressors of different kind to estimate the generated gripping forces from recorded surface electromyographic activity signals (EMG) during gripping task. The dataset we adopted to evaluate our approach consist of sEMG-Force data profile of 24 sEMG channels placed as 3 bracelets patterns each of 8 electrodes around the forearm muscles while the forces were measured by an ergonomic hand dynamometer was developed. For each finger, the dynamometer features 2 tensiometer sensor points and analogue linearization circuit. The sEMG signals were then filtered and preprocessed to formulate the data frame that will be used to train the proposed ML model. The proposed ML model is a pipeline of stacking 2 different nature ML models, a random forest regressor model (RF regressor) and a multiple layer perceptron artificial neural network (MLP regressor). The models were stacked together, and the outputs were penalized by a Ridge regressor to get the best estimation of both models. The model was evaluated by different metrics, mean squared error and coefficient of determination or r^2 score to improve the model prediction performance. We tuned the most significant hyper parameters of each of the MLP-RF model components using random search algorithm followed by grid search algorithm. Finally, we evaluated our MLP-RF model performance by comparing the prediction results with the state-of-art Recurrent Neural Network (RNN) model and the results shows that the MLP-RF outperforms the state-of-art model.**

***Index Terms*—sEMG signals, Multi-layer perceptron Regressor (MLP), Random Forest regressor (RF), Recurrent Neural Network (RNN), Robot Grasping Forces, skill transfer learning**

I. Introduction

The need to solve manipulation tasks in multiple applications in complex environments raised the relevance of skill transfer learning techniques to teach robotic arms from human demonstration. different approaches have been raised to play an essential role for controlling the robotic manipulators by human demonstrations in order to reduce the time consuming and traditional programming complexity and to outperform the limitations of the classical control methods for robotics to be scalable and generalized for different applications in complex environments. the most related approaches were focusing on behavioral cloning (BC) [1], generative adversarial imitation learning (GAIL) [2] and the inverse reinforcement learning (IRL) [3]. the high computational complexity and high dimensional task spaces for the lateral two approaches affected the efficiency of applying those methodologies on complex tasks.

In behavioral cloning, a neural network (NN) is suitable for modeling non-linear data and is able to account for differences between different conditions. Over the past decade, several NN-based EMG pattern recognition methods have been introduced. For example, in [4], a NN background propagation (BP) is used to perform pattern recognition with frequency responses. In [5], researchers were able to isolate four movements of the forearm (flexion, extension, pronation and supination) using a combination of BPNN and Hopfield NN. In [6], [7] and others, similar work was done. However, the commonly used BPNNs in the above studies do not provide higher learnability and performance, and a large amount of training data is required as well as a large number of training iterations.

In this paper, we are proposing a new behavioral cloning based approach to develop an algorithm that allows building a policy-based machine learning model that takes the raw recorded human arm bio-signals for muscles activities as a state and outputs the appropriate grasping forces to be applied to let the robot imitate the human demonstrated grasping skill. The goal of this study is to build the first essential block of transferring the demonstrated human grasping and manipulation skills for different complex tasks under complex environments to teach a robotic manipulator human skills.

The paper is organised as follows, section II represent the experimental setup of the study, we then implement the Data preprocessing needed before training the proposed model, and explain how features were selected to train the proposed model. Section III represent how we build our proposed model in details while results and discussions under implementing our approach and comparing in to the state of the art RNN approach are illustrated in section IV. Section V holds the final conclusion.

II. Experimental Setup

putEMG-Force datasets were used to train and evaluate our proposed model [8]. Were recorded by Biomedical Engineering and Bio-cybernetics Team – Poznan university of technology, Poland. Databases of surface electromyographic

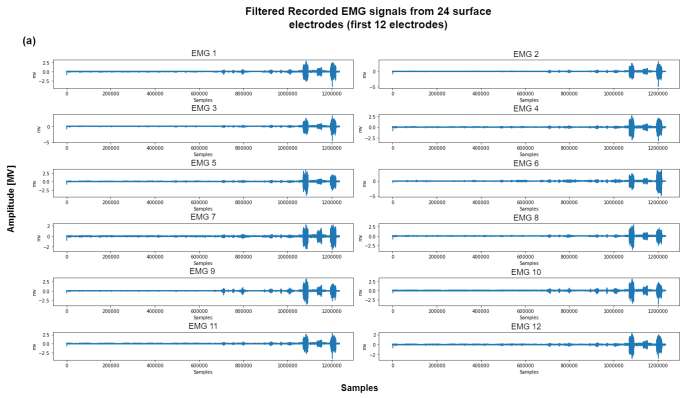

Fig. 1. an example of the recorded EMG signals from 12 sEMG electrodes after filtering. The overall 24 filtered recorded EMG signals are the input data to train the proposed ML model in this article.

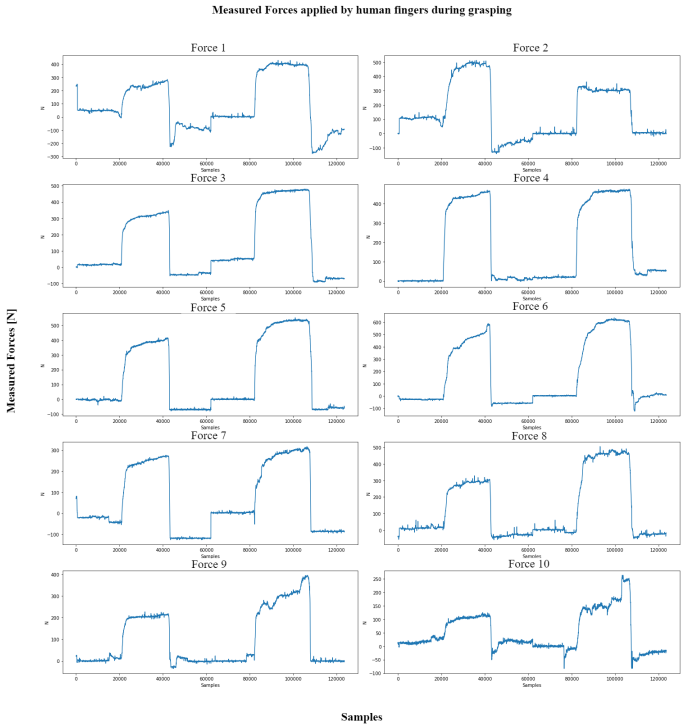

Fig. 2. Measured grasping forces from human fingers. These measures are the output (target) for training the proposed ML model to predict.

activity recorded from forearm. Datasets allows for development of algorithms for gesture recognition and grasp force recognition. Experiment was conducted on 44 participants, with two repetitions separated by minimum of one week. sEMG was recorded using a 24-electrode matrix.

the sEMG-Force data profile consists of 24 filtered EMG signals columns as input (Fig. 1), and 10 recorded Grasping forces as output to train the proposed ML model (Fig. 2).

A. Data Preprocessing

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. The recorded raw EMG-Force dataset contains noises, missing values, because of noticeable artifacts like displacement of sEMG electrodes during recording, the noise that were caught by those electrodes...etc [8], i.e., it can not be directly used for the proposed machine learning models.

Data preprocessing is required tasks for cleaning the data and making it suitable for the ML model which also increases the accuracy and efficiency of the ML model.

First of all, we had to calculate the voltage values from recorded sEMG channels as in Eq. (1) which were stored as ADC raw values where N is an ADC value.

$$x = N \times \frac{5}{2^{12}} \times \frac{1000}{200}[mv] \tag{1}$$

The calculated voltage values were then filtered using Multi-notch filter (frequencies = $30, 49.99, 90, 60, 150Hz$), Butter Worth Band-pass filter ($lp = 20Hz, hp = 700Hz$) as suggested by [8].

the filtered data were cleaned from $Nans$, infinite values and outliers to guarantee the consistency and the stability of the model prediction performance.

the data were rescaled using Standard scaler to bring every feature in the same footing without any upfront importance and to make the gradient descent converge much faster in case of MLP model.

B. Feature Selection.

Feature selection is intended to reduce the number of input variables (the EMG signals) to those that are believed to be most useful to a model in order to predict the target variable. In case of our proposed model where we represent a pipeline of stacking 2 different regressors, Random Forest and Mlutilayer perceptron.

Random forest regressor includes ensembles of decision trees which depend on penalized regression models that select the features which are higher related to predict the output [9].

For the MLP regressor, we adopted a wrapper feature selection algorithm [11] as suggested by [10], as they obtained a significant improvement in the overall results with respect to learning with the whole set of variables in most of the data sets tested.

III. BUILDING THE PROPOSED MODEL PIPELINE

Our approach proposes stacking two different regressors, multi-layer perceptron (MLP) and random forest (RF) regressors, via a meta-regressor, ridge regressor. The individual regressors were trained separately based on the complete training set; then, the meta-regressor was fitted based on the outputs (meta-features) of the individual regressors in the ensemble. This approach adopts stacking regression technique to give improved prediction accuracy [12].

A. Reasons behind the proposed model architecture

The reason behind the proposed approach is to add a noval approach to the bench-marking where the generated gripping

forces can be predicted and estimated from recording the sEMG human muscle activity signals.

The choice of the random forest model was according to the advantages behind adopting ensemble models:

- One of the biggest advantages of random forest is its versatility. It can be used for both regression and classification tasks, and it's also easy to view the relative importance it assigns to the input features.
- Random forest is also a very handy algorithm because the default hyper-parameters it uses often produce a good prediction result. Understanding the hyper-parameters is pretty straightforward, and there's also not that many of them.
- The ability to handle the non-linearity of the sEMG signals as independent predictors of the dependent force output.
- One of the biggest problems in machine learning is overfitting, but most of the time this won't happen thanks to the random forest classifier. If there are enough trees in the forest, the classifier won't over fit the model.

The choice of the multiple layer perceptron neural network (MLP) is because of the following advantages:

- Applicability to complex nonlinear problems which makes it appropriate for our problem statement
- High performance with large input data, in our case, we need a large amount of input data recorder under different scenarios to guarantee the generalizability of the proposed approach.
- Rapidity, where it provides a quick prediction after training, which is critically important to execute the task in real time for a robot.
- Consistency, where the same accuracy can be achieved and guaranteed using smaller amount of data.
- Scalability to different training and prediction scenarios, different shapes of datasets.

The idea behind an ensemble of models (stacking MLP with RF regressor in our approach), is to maximize our models' predictions from multiple machine learning models by assigning weights according to their performance. To guarantee giving the better performing model more say in our final prediction in an ensemble we use stacking algorithm that learns how to best combine each of the models in an ensemble to come up with the best performance.

- An ordinary machine learning model only tries to map input towards output by generating a relationship function.
- Stacking acts on one level above the ordinary by learning the relationship between the prediction result of each of the ensembled models on out-of-sample predictions and the actual value.

In most of the papers discussing stacked models, the meta-model used is often just a simple model such as Linear Regression for regression tasks and Logistic Regression for classification tasks. One reason why more complex meta-models are often not chosen is because there is a much higher chance that the meta-model may over fit to the predictions from the base models [13].

For our problem, Ridge Regression [14] works much better than Linear Regression. This is because the base model's predictions are strongly correlated, as they are all trying to predict the same relationship. Hence, a Linear Regression fit may cause the final prediction to be highly sensitive to changes in the data. Therefore, higher variance leads to bad generalization.

Ridge Regression comes with regularization parameters and hence is able to deal with the correlation between each base model's predictions much better than Linear Regression. This has been shown empirically to be true; however, a general proof has yet been devised in any papers.

Going back to the MLP and RF regressors, the architecture was detected according to tuning hyper-parameters of both models using random search algorithm [15] followed by grid search algorithm [16].

The proposed model structure is illustrated in Fig. 3.

B. Random Forest Regressor

Random Forest Regression [17] is a supervised learning algorithm that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model.

A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees (see Fig. 3 Random Forest Regressor).

To get the best performance of the Random forest regressor, we tuned the most significant hyper parameters using random search algorithm [18] followed by grid search algorithm [19].

The main idea of the aforementioned search algorithms as optimization problem to find the optimal set of model hyper-parameters by randomly iterating over the the predefined set of values for each hyper- parameter, train the model with these values and evaluate the approximation results, finally setting the hyper-parameters' values with the best approximation results. The tuned model hyper parameters are:

- $bootstrap$: Whether bootstrap samples are used when building trees. If False, the whole dataset is used to build each tree.
- $Maxdepth$: The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than $min_samples_split$ samples.
- $maxfeatures$: The number of features to consider when looking for the best split.
- $minsamplessplits$: The minimum number of samples required to split an internal node
- $Minsamplesleaf$: The minimum number of samples required to be at a leaf node.
- $Nestimators$: The number of trees in the forest.

C. Multi-layer perceptron neural networks

A multi-layer perceptron (MLP) [20] is a fully connected class of feed-forward artificial neural network (ANN). The

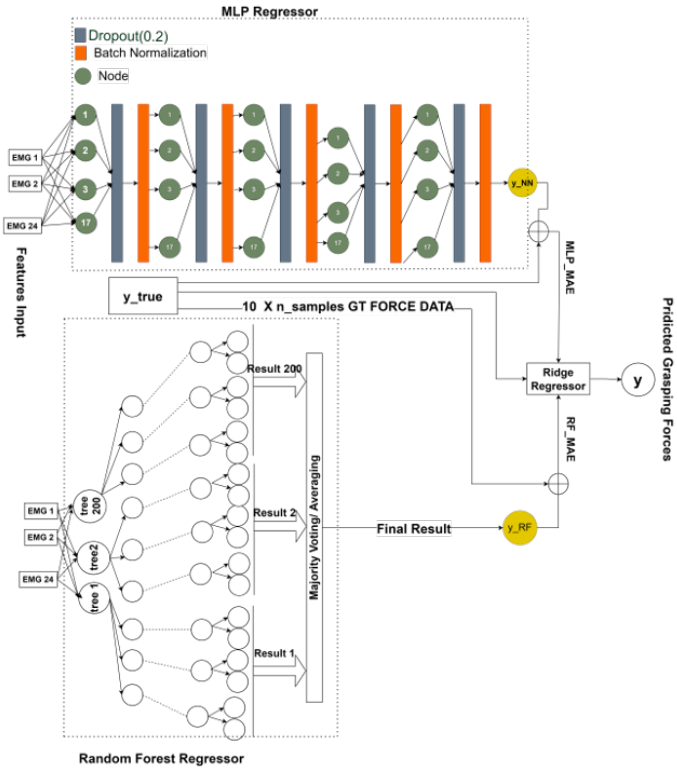

Fig. 3. MLP-RF Model Structure

term MLP is used ambiguously, sometimes loosely to mean any feed-forward ANN, sometimes strictly to refer to networks composed of multiple layers of perceptron (with threshold activation). Multi-layer perceptrons are sometimes colloquially referred to as "vanilla" neural networks, especially when they have a single hidden layer.

1) Activation Function: In recent developments of deep learning the rectifier linear unit (ReLU) is more frequently used as one of the possible ways to overcome the numerical problems related to the sigmoids, for that reason we chose to implement ReLU as an activation function in the dense layers of our model.

2) MLP model architecture: The MLP model has two main hyper-parameters that control the architecture or topology of the network: the number of layers and the number of nodes in each hidden layer.

The MLP regressor was structured as follows:

- Input Layer: The size of the input layer equals to the number of sEMG channels, 24 units.
- Hidden Layers: we have 4 hidden dense layers each of size defined by Eq. (2) , each of them is connected to a batch normalization and dropout layer, respectably.
- Output Layer: To produce the output variables, we have out put layer of size 10, the number of grasping forces to be predicted by the model.

$$N_{hidden} = \frac{N_{inputsamples}}{Factor \times (N_{inputfeatures} + 1)}, \qquad (2)$$

where, $Factor$ is an integer factor, N_{hidden}, the number of nodes in the hidden layer, $N_{iputsamples}$, the number of input data samples and $N_{inputfeatures}$, the number of input features, i.e., the number of sEMG channels.

In addition to the essential dense layers in the MLP regressor, we added two types of special layers, Batch Normalization layers to accelerate the training process and Dropouts layers to avoid the over-fitting and under-fitting problems. More details will be explained later in results and discussion section during model evaluation illustrations.

IV. Results and Discussions

The MLP-RF model was trained on the proposed dataset by splitting them into train and test splits (80% for training and 20% for testing), the specified batch size during MLP training was set to 500 epochs with an early stopping callback condition in case of the training process does not improve the model prediction into 3 consecutive epochs. The specified batch size was 256 samples to accelerate the training process. The specified learning rate for the optimizer was set to 0.001 by convention.

The model was evaluated using K-Fold Cross Validation algorithm [21] to guarantee that the results are stable and accurate. the model prediction performance was evaluated by slicing the data frames into 10 equally sized folds. In each iteration, an arbitrary fold was chosen to validate the prediction results where the 9 other folds for tuning the model weights.

The problem of over-fitting and under-fitting that is very common in machine learning models in general required to improve the model prediction make a variance- bias trade-off. In case of the RF model, we used random search algorithm for tuning the mentioned model hyper parameters in section III-B where the resulted r^2 score metrics shows improvement of performance after finding the appropriate values for the tuned hyper parameters as illustrated in Table. I.

TABLE I
Tuned hyper-parameters for Random Forest Regressor

Random Forest Regressor tuned Hyper parameters	
parameters	***values***
bootstrap	False: all data is used to train each tree
max depth	None: the depth of each tree until reaching the min sample splits
max features	'sqrt': the root square of the number of selected features
min samples splits	2
min samples leaf	1
N estimators	200

Table. II illustrates the resulted mean and std values for r^2 score of model prediction performance after K-Fold Cross Validation using the tuned hyper parameters compared to the default hyper parameters before tuning.

TABLE II
Tuned hyper-parameters for Random Forest Regressor

r^2 score for RF prediction before and after hyper parameters tuning				
r^2 score	**Tuned hyper parameters**		**Default hyper parameters**	
	Training	validation	Training	Validation
	0.95(0.10)	0.75(0.05)	1.00(0.0) (over-fitting)	0.4(0.1)

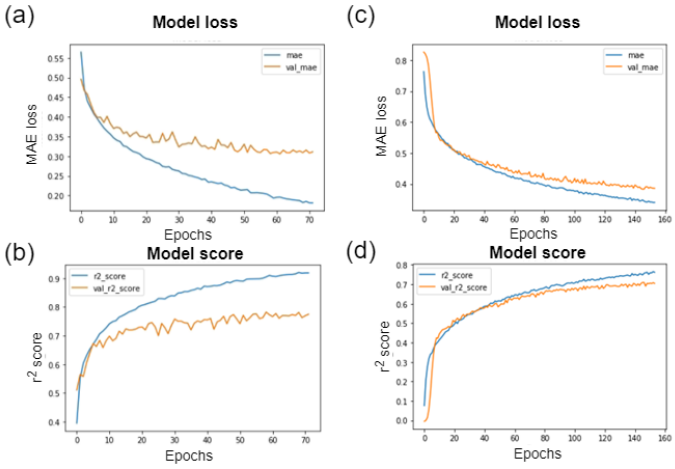

Fig. 4. model prediction performance evaluation where the red curve illustrates the validation result compared to the blue one which illustrates the training result: (a) Model loss before adding dropouts layers; (b) Model r^2 score before adding Dropout layers; (c) Model loss after adding dropouts; (d) Model r^2 score after adding Dropouts

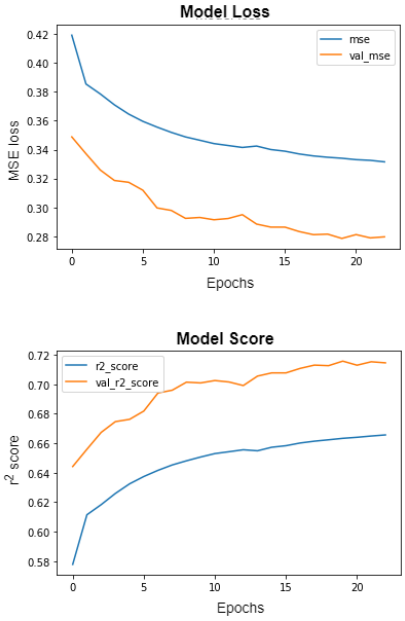

Fig. 5. RNN model prediction evaluation; loss and score

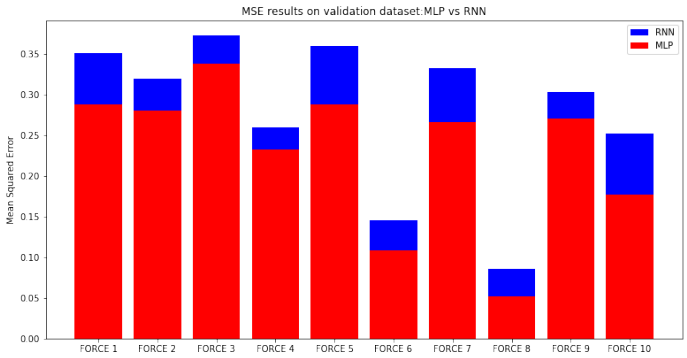

Fig. 6. Comparing validation results of our model and the RNN state of the art model

In case of MLP model, the preliminary results showed over-fitting of the model because of its high complexity compared to the size of dataset that was used for training, for that reason the model architecture described in section III-C was modified by adding special layers called Dropouts, in order to modulate the model complexity according to the size of input data to avoid the over-fitting problem. The drop rate was set to 0.2, i.e., for each batch of the dataset, the layer arbitrarily drops out 20% of the nodes and keeps 80% of them to reduce the model complexity over the dataset batch.

The second necessary modification is adding Batch normalization layer after each hidden layer to automatically standardize the inputs to a hidden dense layer from the previous one. Batch normalization was implemented, as it has the effect of dramatically accelerating the training process of a neural network, and in some cases improves the performance of the model via a modest regularization effect.

Fig. 4 illustrates the evaluation of model prediction after adding Dropout layer to improve the model prediction performance over the over-fitting and under-fitting problems.

It is clearly from Fig. 4 that the improvement of the model prediction performance Where both training and validation on unseen dataset gives approximately similar prediction performance (high r^2 score and low mean absolute error). Even of the outperforming of the model prediction on training scenario in the 1^{st} case over the second one i.e., after adding the dropouts, ($r^2_{trainingwithoutdropouts} = 90\% > r^2_{trainingwithdropouts} = 83\%$), the goal is to reduce the gap of prediction performance between training and validation scenarios to consider the model as a stable, consistent and accurate over different unseen datasets.

Finally , we compare our proposed model to the state-of-the-art Recurrent model [22] on the represented dataset to represent the efficiency of using it in grasping force prediction from sEMG signals. Fig. 7 and Fig. 8 illustrate the model prediction performance of our model and the RNN model respectively.

We see from Fig. 7 that the results are more likely noisy which leads us to clarify an important point of this work; as mentioned before, in each sample step our model predicts the value of 10 grasping force measurements (2 for each finger) , for that reason we notice that the resulted prediction performance is more likely noisy, i.e., in each sample step, the model tries to predict the most likely correct value for each of the 10 output forces. Taking the mean of the ten predicted values from the model, we get the final most likely correct predicted value and the result will be in the middle which is 90% close to the real measured value.

Recurrent neural networks (RNN) are the state of the art algorithm for sequential data and are used by Apple's Siri and and Google's voice search. It is the first algorithm that remembers its input, due to an internal memory, which

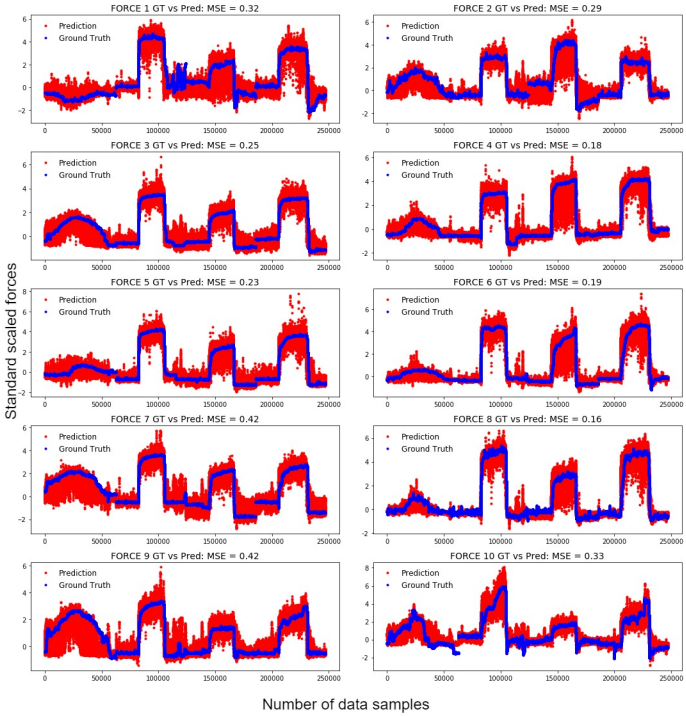

Fig. 7. MLP model prediction performance results for the 10 grasping forces; Ground truth data (GT) is in blue curve, while the predictions (Pred) are in red scatters; MSE: the mean squared error for each force prediction.

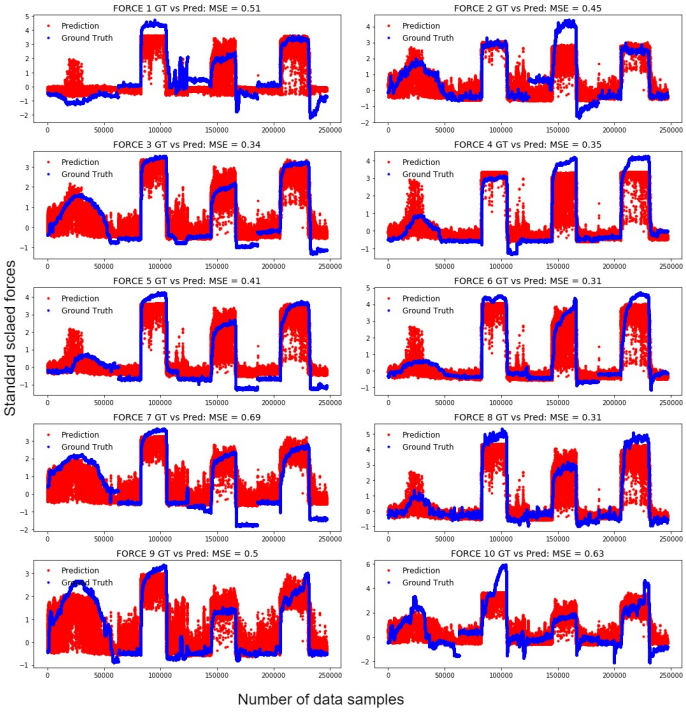

Fig. 8. RNN model prediction performance results for the 10 grasping forces; Ground truth data (GT) is in blue curve, while the predictions (Pred) are in red scatters; MSE: the mean squared error for each force prediction.

makes it perfectly suited for machine learning problems that involve sequential data. The RNN internal memory comes from Long-Short Term Memory layers (LSTM) [23] that make information cycle through a loop. When it makes a decision, it considers the current input and also what it has learned from the inputs it received previously.

Comparing the evaluation metrics; r^2 score (Fig. 5 and Fig. 4-b), and the mean squared error illustrated in Fig. 6 for both models, we find that our proposed model outperforms the state-of-the-art RNN model for predicting the grasping forces from recorded sEMG signals and can be adopted for further development of designing a skill transfer policy to teach a robot the human skills by demonstrating the applied forces and trajectories.

V. Conclusion

In this study, we discussed an implementation of a proposed stacking regressor of 2 different in nature nonlinear regressors Random forest regressor and multi-layer perceptron neural network regressor to estimate the generated grasping forces from human arm muscle activities records. the estimated grasping forces can be then transferred to a robotic manipulator to reproduce the same human grasping skills. the results showed the out performance of our proposed model over the RNN state-of-the-art model that treats the biological muscle activities signals as a time series signals and have a higher complexity, memory and computation cost over the proposed MLP-RF model. the results conclude the efficiency of adopting our model in related application.

References

[1] Torabi, Faraz, Garrett Warnell, and Peter Stone. "Behavioral cloning from observation." arXiv preprint arXiv:1805.01954 (2018).

[2] Ho, Jonathan, and Stefano Ermon. "Generative adversarial imitation learning." Advances in neural information processing systems 29 (2016).

[3] Hadfield-Menell, Dylan, et al. "Cooperative inverse reinforcement learning." Advances in neural information processing systems 29 (2016).

[4] Uchida, Noriyoshi, et al. "EMG pattern recognition by neural networks for multi fingers control." 1992 14th Annual International Conference of the IEEE Engineering in Medicine and Biology Society. Vol. 3. IEEE, 1992.

[5] Kelly, M.F., Parker, P.A., and Scott, R.N. (1990). The Application of Neural Networks to Myoelectric Signal Analysis: A Preliminary Study. IEEE Trans. on Biomedical Engineering, 37(3), 221–230.

[6] Koike, Y. and Kawato, M. (1994). Estimation of Arm Posture in 3D-Space from Surface EMG Signals Using a Neural Network Model. Transactions Institute of Electronics, Information and Communication Engineers, J77-D(4), 368–375 (in Japanese).

[7] Huang, H.P. and Chen, C.Y. (1999). Development of a Myoelectric Discrimination System for a Multi-Degree Prosthetic Hand. In Proc. of the 1999 IEEE International Conf. on Robotics and Automation (pp. 2392–2397). Detroit, USA.

[8] Kaczmarek, P.; Mańkowski, T.; Tomczyński, J. putEMG—A Surface Electromyography Hand Gesture Recognition Dataset. Sensors 2019, 19, 3548.

[9] Kursa, Miron Rudnicki, Witold. (2011). The All Relevant Feature Selection using Random Forest.

[10] Romero, Enrique Sopena, Josep. (2008). Performing Feature Selection With Multilayer Perceptrons. IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council. 19. 431-41. 10.1109/TNN.2007.909535.

[11] G. H. John, R. Kohavi, and K. Pfleger, "Irrelevant features and thesubset selection problem," in Proc. 11th Int. Conf. Mach. Learn., 1994,pp. 121–129.

[12] Breiman, L. Stacked Regressions. Machine Learning 24, 49–64 (1996). https://doi.org/10.1023/A:1018046112532

[13] Lim, Y., Stacked Ensembles — Improving Model Performance on a Higher Level. Towards Data Science, 2021.
[14] Hilt, Donald E.; Seegrist, Donald W. (1977). Ridge, a computer program for calculating ridge regression estimates
[15] Al-Muhammed, Muhammed Jassem, and Raed Abu Zitar. "Probability-directed random search algorithm for unconstrained optimization problem." Applied Soft Computing 71 (2018): 165-182.
[16] William Menke, in Geophysical Data Analysis: Discrete Inverse Theory (Third Edition), 2012
[17] Ho, Tin Kam (1995). Random Decision Forests. Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282
[18] Zabinsky, Zelda B. "Random search algorithms." Department of Industrial and Systems Engineering, University of Washington, USA (2009).
[19] Liashchynskyi, Petro, and Pavlo Liashchynskyi. "Grid search, random search, genetic algorithm: a big comparison for NAS." arXiv preprint arXiv:1912.06059 (2019).
[20] Hastie, T.T., Robert. Friedman, Jerome., The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York, NY, 2009.
[21] Stone, M (1977). "An Asymptotic Equivalence of Choice of Model by Cross-Validation and Akaike's Criterion". Journal of the Royal Statistical Society, Series B (Methodological). 39 (1): 44–47.
[22] Medsker, Larry R., and L. C. Jain. "Recurrent neural networks." Design and Applications 5 (2001): 64-67.
[23] Graves, Alex. "Long short-term memory." Supervised sequence labelling with recurrent neural networks (2012): 37-45.